



**Universiteit Leiden**

**ICT in Business and the Public Sector**

**TOWARDS A MULTI-DIMENSIONAL MATURITY  
MODEL: ASSESSING DEVOPS SUCCESS**

Does DevOps work? An analysis of how DevOps changes the digital user's  
experience

Name: Anne Plancius

Student-no: s0928569

Date: 10/12/2019

1st supervisor: Dr C. J. (Christoph) Stettina

2nd supervisor: T. (Tyron) Offerman MSc

MASTER'S THESIS

Leiden Institute of Advanced Computer Science (LIACS)

Leiden University

Niels Bohrweg 1

2333 CA Leiden

The Netherlands

## Acknowledgements

First of all, I would like to thank my thesis advisor, Dr C. J. (Christoph) Stettina of the Centre for Innovation at Leiden University in the Netherlands. Dr Stettina was patient and continually motivated me to finish: He gave his full support throughout the drafting of the questionnaire and kept quietly pushing me to complete my work.

I would also like to thank the practitioners from the different companies who were involved in the interviews. Without their honest and open conversations during the interviews, I would not have been able to assemble the case studies, and so the validation of the theories discussed herein would not have been possible.

Another acknowledgement should go to Tyron Offerman, MSc, from the Leiden Institute of Advanced Computer Sciences at Leiden University as the second reader of this thesis. He provided invaluable new insights that improved the quality of this thesis.

Last but not least, I want to express my gratitude to my family—Irene, Jens, and Ymke. Irene, thanks for all the moments you stepped in to support our kids; Jens, thank you for listening to a father who was constantly speaking about DevOps. Ymke, I must express my gratitude for pushing me to the limit by constantly asking: “Isn’t your thesis ready yet?”

This thesis would still be in my mind and not on paper if it weren’t for the support of everyone who helped me so much.

Anne Plancius

## Abstract

This paper brings together my working life (as a presales consultant for various companies selling software that supports developers and operators) and my life as a student. Throughout my working career and during my studies, I have expressed an interest in DevOps, one of the topics of greatest interest to most of my customers/clients. In that context, the question often arises: “Does DevOps deliver value?”. This thesis focuses on answering that question and, during the process of seeking whether DevOps delivered tangible results, addressed a number of related subtopics:

- How can DevOps be defined? And how can DevOps be practically implemented?
- Can one company be “more” DevOps-oriented than another? Does a maturity model exist for assessing DevOps implementation?
- Are there business indicators demonstrating that a company that is more DevOps-oriented experiences concrete business benefits?

In order to find answers to these questions, a number of case studies were undertaken, consisting of multiple interviews at companies within the same industry and the use of external data in order to measure success.

After analyses of the data were performed, the conclusion was reached that the current practical application of DevOps is heavily focused on automation and does not fully implement the principles of DevOps. Another outcome was the realization that the practical implementation of DevOps (as mirrored in these case studies) did not yield measurable business benefits.

A critical conclusion that was reached was that a maturity model for DevOps, one enabling comparisons to be made among companies, currently does not exist and remains to be formulated. In light of the fact that transitioning to DevOps is such a complex process, even the creation of a maturity model that facilitates comparisons cannot do justice to such an intricate and involved topic. Further research on what enhances full and effective implementation of DevOps needs to be carried out in the future.

## Contents

1.	Introduction .....	8
1.1	Why was this research initiated? .....	8
	What is the problem being investigated? .....	8
1.2	Is there an answer to questions about DevOps? .....	9
1.3	Why is the development of a maturity model so important? .....	9
1.4	Research question.....	10
2.	The scope of this thesis.....	11
3.	Related work.....	12
3.1	Introduction .....	12
3.2	Practitioner literature .....	24
3.3	Maturity models.....	36
4.	Research method.....	43
4.1	Research strategy and design .....	43
5.	Data collection .....	50
5.1	Company A.....	50
5.2	Company B.....	56
5.3	Company C.....	62
5.4	Company D.....	67
6.	Analysis .....	74
6.1	Definitions for DevOps.....	74
6.2	DevOps maturity .....	76
6.3	Success .....	81
7.	Discussion.....	91
7.1	Definitions of DevOps vs. the realities of actual implementation: Release engineering vs. DevOps.....	91
7.2	A DevOps Maturity Model: Towards a focus area maturity model.....	94
7.3	DevOps Success: Release frequency as the critical success indicator .....	98
8.	Conclusions .....	100
8.1	Release engineering and app ratings .....	100
8.2	Does DevOps matter? .....	100
8.3	Current DevOps maturity models are inadequate.....	101
9.	Comments and recommendations for future research.....	102

9.1	Limitations to the validity of the research .....	102
9.2	Future work.....	103
10.	References .....	104

---

List of figures

---

Figure 1: Software lifecycle, by Barry Boehm .....	21
Figure 2: Maturity Model for Continuous Delivery, by David Farley and Jez Humble .....	35
Figure 3: DevOps maturity levels, as defined by Samer I. Mohamed .....	39
Figure 4: Micro Focus DevOps maturity .....	41
Figure 5: Box of bricks of research .....	43
Figure 6: The research pyramid .....	44
Figure 7: Company: IOS app rating .....	55
Figure 8: Company A: Google Play Store app rating .....	55
Figure 9: Company B: IOS App rating .....	61
Figure 10: Company B: Google Play Store app rating .....	61
Figure 11: Company C, IOS App Rating .....	66
Figure 12: Company C: Google Play Store app rating .....	67
Figure 13: Company D: IOS app rating .....	72
Figure 14: Company D, Google Play Store App Rating .....	73
Figure 15: Micro Focus DevOps Maturity Model .....	78
Figure 16: Company A, Apple App Store release frequency .....	82
Figure 17: Company B, Apple App Store release frequency .....	83
Figure 18: Company C: IOS app rating .....	87
Figure 19: Company C: IOS app release frequency .....	88
Figure 20: Company B, IOS App Release Frequency .....	88
Figure 21: Company : IOS app rating .....	89
Figure 22: Company: IOS app release frequency .....	89
Figure 23: Company A, IOS App Rating .....	90

---

*List of tables*

---

Table 1: The scope of this research .....	11
Table 2: Definitions of DevOps .....	15
Table 3: Overview of practitioner literature .....	23
Table 4: The Phoenix Project, by Kim, Behr, and Spafford .....	25
Table 5: DevOps: A Software Architect’s Perspective, by Bass, Weber, and Shu .....	27
Table 6: The DevOps Handbook, by Kim, Humble, Dubois, and Willis .....	28
Table 7: Continuous Delivery and DevOps: A Quickstart Guide, by Swartout.....	30
Table 8: Effective DevOps, by Davis .....	32
Table 9: Continuous Delivery, by Farley and Humble .....	34
Table 10: DevOps deontic constraints (as defined by Mc Carthy and others) .....	38
Table 11: Research strategies by Saunders and others .....	45
Table 12: Data collection overview.....	50
Table 13: Company A: NPS scores.....	54
Table 14: Company A: Banking monitor trust index.....	54
Table 15: Company:, Banking Monitor Trust Index .....	60
Table 16: Company C: Internal measurement of mobile app ratings.....	65
Table 17: Company C: NPS Scores .....	65
Table 18: Company C: Banking Index Trust Monitor .....	66
Table 19: Company D: NPS scores .....	71
Table 20: Company D: Banking Index Trust Monitor .....	72
Table 21: DevOps maturity, according to Mohamed.....	78
Table 22: DevOps maturity comparison .....	81
Table 23: Internal release frequency .....	82
Table 24: Banking Monitor Overview .....	84
Table 25: Collaboration and communication overview.....	84
Table 26: NPS overview .....	85
Table 27: Trust Monitor overview .....	85
Table 28: Quality statements.....	86
Table 29: Company C: Internal app rating .....	87
Table 30: An overview of the practices in theory and in the cases .....	92
Table 31: DevOps: A focus area maturity model .....	97
Table 32: External Success Metrics.....	98

# 1. Introduction

## 1.1 Why was this research initiated?

When I was working at BMC Software in 2003, Nicolas Carr, the acclaimed writer of *The Shallows* (2010) and Richmond Visiting Professor at Williams College in the United States, rocked my world when he wrote: “IT does not matter”<sup>1</sup>. I went on to debate this statement with others. Of course, my idea was and still is that IT *does* matter. In 2011, my views were boosted by Marc Andreessen (2011), who stated that: “Software is eating the world”, meaning that IT (software) budgets were crowding out spending on hardware. The ensuing debate supported my view that IT does matter and, because it does, developments in that field need to be investigated and analyzed. No one questions the fact that the use of IT is pervasive and omnipresent: We all use our mobile phones to do banking, arrange transportation, read books, and/or listen to streaming audio.

In the heat of this discussion and as a consultant at different companies where I worked to support IT operations, a new movement arose: It was called DevOps, which was a set of practices and a mindset that contributed to shorten the systems development lifecycle, thus enabling the continuous delivery of high-quality software to take place.

While working with customers and prospective clients, I started to discuss this new movement called DevOps. Of course, in the beginning, we shared ideas about what DevOps is, or what we thought the terminology meant. Once there was agreement, we shared implementation practices. But even before I began researching this thesis and investigating this topic, I realized that crucial discussions about people implementing DevOps were not taking place: In short, the “why” of implementing DevOps and whether this system brought business benefits were never raised and answered. This thesis attempts to answer these critical questions.

## What is the problem being investigated?

Due to market circumstances, the work of the institutes like the DevOps Agile Skills Association (DASA), and the efforts of the DevOps Institute, companies are currently being triggered to implement DevOps: That said, what are the business benefits that they can expect from it?

Ultimately, a second crucial question needs to be asked: Can some companies implement DevOps more extensively than others and, if so, how and why does that happen? What is it that those companies need to focus on in order to implement DevOps more completely?

Personally, I was interested in knowing whether there were business benefits to be gained from implementing DevOps. When Marc Andreessen (2011) made his famous statement that software was eating the world, did he imply that a better method can be created, one that will help to build and run software and do it more effectively? Would such a system have an impact on the business world and could this be measured and quantified?

---

<sup>1</sup> <https://hbr.org/2003/05/it-doesnt-matter>

## 1.2 Is there an answer to questions about DevOps?

What is needed to ensure that companies successfully implement DevOps is a common understanding of the principles underlying it; even more importantly, what is required is the development of a maturity model for DevOps, enabling it to be assessed. This maturity model will give companies guidance on what to improve, similarly to what took place in relation to the Capability Maturity Model (Paulk, 1995).

It would be useful during any analysis to check the claim made by Sharma (2015) and Chen (2015) by comparing, on the basis of the maturity model, the performance of companies operating in a similar market. Analyzing the claims they make and the claims of Marc Andreessen (2015) would lead to identifying a set of companies that are delivering applications to Google Play and to the App Store, thus enabling a measurement of their ratings and the release sequence of their applications to be made. That will not deliver information about the time to value, but it will confirm that the user likes the mobile applications provided by the companies. It will also show the frequency of releases. Apart from other external data, NPS scores and the banking index can show if implementing DevOps is having a business impact.

In order to verify my assumptions and my hypotheses, I have chosen to look at the banking industry in the Netherlands, which is one of the sectors that is being “eaten up” by software, a reality confirmed by Peter Jacobs who, as the Chief Information Officer (CIO) of ING Bank in the Netherlands (until 2013), stated: “ING is an IT company with a banking licence”. This quote proves that IT plays a critical role in banking. If that is the case, then the implementation of DevOps should have a huge impact on the effectiveness of the bank that has adopted this set of practices.

## 1.3 Why is the development of a maturity model so important?

A maturity model will help companies to establish a focus on what to implement in their trajectory towards DevOps. In addition, a maturity model will give them a mechanism enabling them to compare themselves against their peers in terms of their maturity levels, leading to a common understanding of DevOps. The comparison of the mobile application ratings and the release frequency should demonstrate if the claims of Sharma (2015) and Chen (2015) are accurate and verifiable.

If I can deliver a definition of DevOps and if I can create a mechanism to compare which practices a set of companies are using, then I will have delivered an answer to the first question.

In addition, the results of my research will enable companies to engage in comparisons in accordance with DevOps standards and will link their application of DevOps to other company assets. In addition, I will be providing, as a result of my research, detailed information about the assertions made by Sharma and Chen. A thorough analysis should prove that their claims are either right or that they are wrong.

## 1.4 Research question

The central question to be answered by the research is:

“On what basis can companies be compared in terms of their success in implementing DevOps?”

This question was answered on the basis of several supporting/subsidiary questions, the answers to which will contribute to a fuller understanding of the main question. Some of these questions are:

1. What is DevOps? A common definition of DevOps will ensure that there is agreement as to how its success can be described and verified.
2. How can the potential maturity stages be defined as DevOps develops inside an organization? If the application of DevOps is to be compared among companies, it is crucial that the parameters for assessment are consistent, precisely defined, and understood so that the extent of DevOps implementation can be measured.
3. Precisely what are the indicators measuring success in terms of DevOps implementation inside an organization? In other words, once the maturity of DevOps inside a company has been measured, the next crucial step is to define the criteria proving that the application has been successful.

All of the above has resulted in the exploration of the following topics in this thesis:

- a definition for DevOps, one which shows its important elements and which provides a common ground for understanding;
- a maturity model for DevOps, a tool which provides the means of carrying out comparisons among companies in relation to DevOps implementation and the resulting levels of performance; and
- a method to measure the impact of DevOps on a company and to identify the important key performance indicators (KPIs) that show the business impact of DevOps, thus providing a justification as to whether DevOps should be implemented or not.

## 2. The scope of this thesis

The aim of this thesis is to deliver not only a definition for DevOps, but a maturity model for it as well. The following table provides a better insight into the scope of the research.

Topic	In Scope	Not in Scope
DevOps	<ul style="list-style-type: none"><li>• Definitions</li><li>• Management practices</li></ul>	<ul style="list-style-type: none"><li>• Tool support</li></ul>
Maturity models	<ul style="list-style-type: none"><li>• Focus and process areas</li></ul>	<ul style="list-style-type: none"><li>• Prescriptions for implementation</li></ul>
Success	<ul style="list-style-type: none"><li>• Success of ratings in banking applications</li><li>• Delivery frequency</li></ul>	<ul style="list-style-type: none"><li>• Definitions of KPIs for success</li><li>• Comparisons of financial statements</li></ul>

*Table 1: The scope of this research*

The table above makes the scope of the research clear. This is crucial because debates about DevOps often concentrate on continuous delivery, an approach that examines automation and tooling. Such a focus could potentially distort the results, since this researcher works for a vendor that sells tools for continuous delivery and this fact could, albeit inadvertently, bias the findings and the analysis. Therefore, a focus on continuous delivery is beyond the scope of this thesis.

When reading McChrystal (2015), I came to the conclusion that, in a complex world, everything centres around adaptability, so a prescriptive model about how to implement DevOps would come into conflict with the important principle of adaptability; this would render discussions about DevOps useless, because every implementation requires a different approach. In short, every application of DevOps should be unique.

While success can be measured in many different ways, according to the article “Exploiting the software advantage”, by Freeform Dynamics Ltd (2015), the main driving force for digital initiatives is improved customer satisfaction, something that can be measured by the ratings that are reflected in the different app stores. Information from Sanjeev Sharma and Bernie Coyne (2015) indicates that DevOps is being implemented in order for organizations to achieve a faster time to value, a factor that can be checked on the basis of the delivery frequency figures found in the different app stores.

Of course, it can be argued that the most important metric for a company is ultimately the financial position of the company, but this position is dependent on much more than the methodology by which the company manages its IT department. This topic is, therefore, extensively discussed in this thesis.

## 3. Related work

### 3.1 Introduction

This section elaborates on the origins, definitions, and main concepts (as covered by scientific and practitioner literature) of DevOps.

#### 3.1.1 The history of DevOps

At the IT Revolution website<sup>2</sup>, John Willis, one of the early practitioners of DevOps, explains that, in his view, three major threads led to the development of DevOps.

The first major thread, one involving Agile infrastructure (described here), also featured in a movie by Damon Edwards, which can be found on the internet<sup>3</sup> and is called “The (Short) History of DevOps”. This film reveals that DevOps development grew out of the Agile Conference in Toronto, Canada, in 2008, and began with a conversation, apparently in a hallway, between Andrew Shafer and Patrick Dubois, who went on to form the “Agile Systems Administration Group”. Andrew Shafer had posted that he was going to be giving a session on Agile infrastructure, and that the only person that showed any interest was Patrick Dubois. Because of the apparent lack of interest, Andrew did not appear at his own session. But Patrick was motivated by the subject matter and pursued his own enquiries. In a conversation, they agreed to arrange a meeting that eventually became, in Ghent, Belgium, the foundation for DevOps, an initiative that focused on the infrastructure of Agile.

The second thread, the velocity thread, was based on the presentation that John Allspaw, delivered on the O’Reilly Velocity 2009 conference, in San Jose, California, entitled: “10+ Deploys Per Day: Dev and Ops Cooperation at Flickr”<sup>4</sup>. The presentation conveyed the importance of integrating development and operations. Patrick Dubois was not physically present, but was virtually present at the event. The presentation depicted what Patrick Dubois was trying to achieve in Belgium during a migration of a data centre for the Belgian government. This inspired the start of #DevOpsDays on Twitter, and later to a conference in Ghent, Belgium, in 2009.

The last thread, the Lean startup thread, was based on a book called *The Four Steps to the Epiphany* (2005), written by Steve Blanks, a book that, according to his own blog, had a huge impact on Eric Ries, author of *The Lean Startup* (2011) and an active blogger on the Startup Lessons Learned blog<sup>5</sup>. Eric Ries was also an advisor to Wealthfront Inc, an automated investment service located in Palo Alto, California, which became a poster child for DevOps.

In the scientific literature, Floris Erich, Chintan Amrit, and Maya Daneva, in their article “Report: DevOps Literature Review” (2014) ) and Breno Bernard Nicolau de França, Helvio Jeronimo Junior, and Guilherme Horta Travassos, in their article, “Characterizing DevOps by Hearing Multiple Voices”, state

---

<sup>2</sup> <https://itrevolution.com/the-convergence-of-devops>

<sup>3</sup> <https://www.youtube.com/watch?v=o7-luYS0iSE>

<sup>4</sup> <https://conferences.oreilly.com/velocity/velocity2009>

<sup>5</sup> <http://www.startuplessonslearned.com/2008/11/what-is-customer-development.html>

that the term “DevOps” was introduced in 2009. Rappaport (2014) expresses the view that the term was coined at the O’Reilly Velocity Conference, which connects it with the velocity thread. All of the accounts on the internet seem to say that the starting point for DevOps consisted of a hashtag on Twitter: #DevOpsDays.

The first main event profiling DevOps was “devopsdays”, organized in Ghent, Belgium (Willis (2010) and Rapaport (2014), an event which changed organizations and the way people worked; this event, one of the first of many to come around the world, focused on analyzing Agile infrastructure, and all on the basis of the expertise and experience of Patrick Dubois. At the time, Dubois was in charge of facilitating a major data centre migration for the Belgian Government and, therefore, was fully aware of the needs of both the development and operations sectors. In that context, he realized that there were communications problems between these two areas. Although people working in development were already introducing Agile development methodologies, those in charge of operations could not cope with these new ways of working—a challenge that Dubois wished to address.

On another website from the company Chief, John Willis describes what DevOps means to him, stating: “DevOps is not a plan, it is a reaction”<sup>6</sup>. Willis mentions in this blog that Patrick Dubois had noticed a trend arising from Agile-based web operating companies. These companies used their Agile development methodologies to run their operations, a fact which led to a flurry of articles and to a small bar-type conference called “#DevOpsDays in Ghent”. These developments clearly linked DevOps to the Agile infrastructure thread.

Breno Bernard Nicolau de França, Helvio Jeronimo Junior and Guilherme Horta Travassos (2016, p.56) in their article “Characterizing DevOps by Hearing Multiple Voices “ define DevOps as “neologism that characterizes a movement of ICT professionals who are adopting a different attitude in relation to software delivery by promoting collaboration between software systems development and its operations functions”: Their approach is based on a set of principles and practices such as culture, automation, measurement, and sharing.

### 3.1.2 Definitions of DevOps

Table 2 provides varying definitions of DevOps by experts and practitioners in the field:

---

<sup>6</sup> <https://blog.chef.io/what-devops-means-to-me/>

Source	Definition Text	Emphasis on Concepts			
Scientific Sources					
		Culture	Automation	Measurement	Sharing
Andrej Dyck; Ralf Penners; and Horst Lichter: "Towards Definitions for Release Engineering and DevOps"	"DevOps is an organizational approach that stresses empathy and cross-functional collaboration within and between teams – especially development and IT operations – in software development organizations, in order to operate resilient systems and accelerate delivery of change".	X			X
Floris Erich; Chintan Amrit; and Maya Daneva: "Cooperation Between Software Development and Operations: A Literature Review"	"DevOps is a collection of principles that try to improve cooperation between development and operations in organizations which separate these two functions".	X			X
Ramtin Jabbari; Nauman bin Ali; Kai Petersen; and Binish Tanveer: "What is DevOps? A Systematic Mapping Study on Definitions and Practices"	"DevOps is a development methodology aimed at bridging the gap between development and operations, emphasizing communication and collaboration, continuous integration, quality assurance, and delivery with automated deployment utilizing a set of development practices."	X	X	X	X
Tuuka Peuraniemi, Review: "DevOps, Value-Driven Principles, Methodologies and Tools"	"DevOps combines operations and development while extending agile methodologies and principles outside development. DevOps has four different aspects: culture, automation, measurement, and sharing. Culture meaning people over processes and tools, automation for quick feedback, measurement for quality control, and finally sharing tools and processes with others".	X	X	X	X
Breno Bernard Nicolau de França, Helvio Jeronimo Junior and Guilherme Horta Travassos: "Characterizing DevOps by Hearing Multiple Voices"	"DevOps is a neologism representing a movement of ICT professionals addressing a different attitude regarding software delivery through the collaboration between software system development and operations functions, based on a set of principles and practices such culture, automation, measurement and sharing".	X	X	X	X

Practitioner sources					
Gartner: Gartner IT Glossary <a href="https://www.gartner.com/en/information-technology/glossary/devops">https://www.gartner.com/en/information-technology/glossary/devops</a>	<p>“DevOps represents a change in IT culture, focusing on rapid IT service delivery through the adoption of agile, lean practices in the context of a system-oriented approach. DevOps emphasizes people (and culture), and seeks to improve collaboration between operations and development teams. DevOps implementations utilize technology — especially automation tools that can leverage an increasingly programmable and dynamic infrastructure from a life cycle perspective”.</p>	X	X		
"Micro Focus: What is DevOps?" <a href="https://www.microfocus.com/en-us/what-is/devops">https://www.microfocus.com/en-us/what-is/devops</a>	<p>“DevOps is best defined as a philosophy or ideology. Many of the underlying principles and language of the DevOps philosophy are grounded in a combination of Agile software development plus Kaizen, Lean Manufacturing, and Six Sigma methodologies”.</p> <p>“The common goal of DevOps is to remove friction, risk, and other constraints to enable faster, more successful application production rollouts, as often and as rapidly as the business requires”.</p>	X		X	

Table 2: Definitions of DevOps

### 3.1.2.1 Scientific definitions of DevOps

Although this section gives an overview of the scientific definitions of DevOps, DevOps is a movement started by practitioners, and therefore it is appropriate to begin the discussion with John Willis, who had already, in his article, “What DevOps Means to Me” (2010), provided a framework (CAMS), an acronym that describes a model for DevOps and that provides an explanation of how DevOps is implemented by many teams:

- Culture: People and process come first.
- Automation: The place where he (Willis) places tooling, a mechanism that “stiches together an automation fabric”.
- Measurement: “If you can’t measure, you can’t improve”.
- Sharing: The “loopback” in the CAMS cycle: A culture where people share ideas and problems is crucial.

This framework for CI/CD integration was picked up both by Jez Humble, a lead practitioner in the field of automation and described in his book *Continuous Delivery*, and by Joanne Molesky, in their article “Why Enterprise Must Adopt DevOps to Enable Continuous Delivery” in the *Cutter IT Journal* of August 2011. This particular edition of the *Cutter IT Journal* is completely focused on DevOps. In this article, the authors not only explain the CAMS framework, but they add Lean to the mix, together with discussions on the concept of the value stream.

This article was referenced by Floris Erich, Chintan Amrit, and Maya Daneva (2014) in their review of the practitioner literature contained in the “DevOps Literature Review” (2014), and in their article, “Cooperation Between Software Development and Operations: A Literature Review” (2014).

Tuuka Peuraniemi (2015) defines DevOps as a combination of development and system operations, one which extends Agile methodologies and principles in a manner that transcends development. In addition, he enumerates four different aspects that are intrinsic to DevOps: culture, automation, measurement, and sharing. Expanding on the meaning of these aspects and on their importance, Peuraniemi states that “culture” involves focusing on people over processes; “automation” is crucial for quick feedback; “measurement” is critical for quality control; and “sharing” means that knowledge about tools and processes are dispersed among others (2015), which is a clear reference to Willis’ framework. In addition, Peuraniemi (2015) references one of the articles of Erich et al. (2014).

Mitesh Soni (2015) states that DevOps culture extends the Agile methodology in a way that rapidly creates applications and delivers them across the business environment in an automated manner, thus improving performance and quality assurance. Speaking about collaboration, Soni describes DevOps as an emerging culture in which development, testing, and operations teams collaborate to deliver outcomes in a continuous and effective manner.

In their article, “Management Challenges for DevOps Adoption within UK SMEs”, Stephen Jones, Joost Noppen, and Fiona Lettice (2016) describe DevOps as an approach that emphasizes a culture of collaboration through the harmonization of software development and IT operations. This article is based on a case study, and so extends the theoretical framework that Erich et al. (2014) have built to include actual practices.

In their report on the “First International Workshop on Emerging Trends in DevOps and Infrastructure”, Rakesh Rana and Miroslaw Staron (2016) elaborate on Jones’, Noppen’s, and Lettice’s explanations of the four principle aspects characterizing DevOps, by providing the following definitions:

1. “Culture” means establishing a collaborative environment, meaning that developers and operators are both responsible for change, maintenance, and stability. Development and operations sectors in a business/organization should be working in unison to deliver high-quality software to end-users.
2. “Automation” requires that full automation is implemented in order to speed up delivery and feedback and thus enhance building, testing, and deployment.
3. “Measurement”: Success and improvement can only be demonstrated if measurements of progress are taken. The following are some of the elements that need to be assessed: varying aspects of the project; business metrics; technical information; and customer satisfaction.

4. “Sharing”: A process that brings development and operations closer together; this entails collaboration in terms of infrastructure, tools, and knowledge.

Erich et al. (2014) in their article, “Cooperation Between Software Development and Operations: A Literature Review”, also describe DevOps as a set of practices and principles that are related to culture, automation, measurement and sharing (CAMS). But they suggest an extension to the CAMS framework so that it can include the concepts of services, quality assurance, structures, and standards. However, the latter concepts are not accepted by other researchers.

De Franca et al (2016) further elaborate on the principles underpinning DevOps by expanding on the following aspects related to the four factors discussed above:

- social aspects: Involve facets related to culture, e.g. collective performance evaluation; an environment of trust; openness to change; and respect among team members;
- automation: an automated collection of metrics, automation of repetitive tasks, and automated delivery of infrastructure;
- quality assurance: Ensures high standards in terms of both development and operations processes as products;
- leanness: DevOps requires a Lean process, as it has as its goal the assurance of a continuous flow, enabling software to be developed and delivered regularly, in small incremental changes;
- sharing: Information and knowledge are disseminated among individuals to promote the exchange of personal learning and project information; and
- measurement: Measuring processes is key, in that assessment instantiates developments and thus results in the smooth functioning of software development and operations lifecycles.

Lucy Ellen Lwakatare, Pasi Kuvaja, and Markku Oivo (2015), in their article, “Dimensions of DevOps”, describe the following dimensions of DevOps and explain the following patterns:

- collaboration: rethinking the roles of teams in development and operations activities and how they can be effectively reoriented;
- automation: continuous deployment of infrastructure and functionality, which in marketing terms means continuous change;
- monitoring: instrumenting application and aggregating monitored data into insights; and
- measurement: collecting the data to measure the performance of development.

A crucial insight in the last article is that the authors deliver the additional perspective of monitoring, which should be, in the mind of the researcher, a normal part of operations and an important part of successful delivery of IT services.

In a second article, Lucy Ellen Lwakatare, Pasi Kuvaja, and Markku Oivo (2016) underscore the importance of conducting “[a]n exploratory study of DevOps, extending the dimensions with practices”, thus emphasizing useful practices that have been used in the questionnaire.

It can be seen that, on the basis of the articles and references mentioned above, the principle aspects of DevOps are culture, automation, measurement (including monitoring), and sharing. The results of the

research carried out would seem to lead to the conclusion that the definition by Dyck et al (2015, p.3) is, perhaps, the best one at this point in time: “DevOps is an organizational approach that stresses empathy and cross-functional collaboration within and between teams – especially development and IT operations – in software development organizations, in order to operate resilient systems and accelerate delivery of change”.

### 3.1.2.2 Practitioners

The previous section mentioned a list of practitioners who were important to the DevOps movement. However, besides John Willis, Patrick Dubois, and Jez Humble, other people significantly influenced the movement as well. Amongst those are authors like Gene Kim, who also founded IT revolution and who wrote, together with Kevin Behr, and George Spafford, a book entitled *The Phoenix Project*, which is seen as a novel about DevOps<sup>7</sup>.

Paragraph 3.2 summarizes the main literature from different practitioners, and then compares those books with the CAMS framework, as introduced in paragraph 3.1.1.1. Apart from a detailed comparison of the literature, it is important to provide a definition of DevOps, as it is seen and practised by practitioners, since this would highlight the differing perspective between science and practice.

What follows below is a list of DevOps definitions that emerge from the available literature; these definitions are combined with information from the DevOps Agile Skills Association, which is an independent organization whose aim is the development of DevOps training and certification.

L. Bass, Weber I., and Zhu, I. (2015, p.4) define DevOps as “a set of practices that are intended to reduce the time between introducing a change to a system and the change being transitioned into normal production, and in a manner that ensures the highest quality”. To accomplish this, Bass (2015, pp. 5-6) emphasizes that the following steps should be taken in relation to DevOps practices:

- Treat operations personnel as first-class citizens from the point of view of requirements. Operators are people that maintain an organization’s systems, so they will have specific requirements and knowledge about those systems. An organization needs to make sure that those requirements are part of a business’s development.
- Make development personal more responsible for incident handling and break the barriers between development and operations. Focus on making sure that operators are part of the development process and ensure that developers are part of the support processes.
- Enforce the deployment process, the traceability, and consistency used by all, by including by Dev and Ops personnel.
- Use continuous deployment.

---

<sup>7</sup> <https://itrevolution.com/faculty/gene-kim/>

- Develop infrastructure codes and create scripts that can configure the infrastructure that is needed for this service.

The definition of DevOps by Bass (2015) is comparable to the definition of DevOps from Dyck and others (2015). On the other hand, Jennifer Davis and Katherine Daniels (2016, p.11), in their book *Effective DevOps*, define DevOps as “a cultural movement that changes how individuals think about their work, because this set of practices enables the diversity of the work to be valued, supports intentional processes that accelerate the rate by which the business realize value, and measures the effect of social and technical changes”.

Similarly to Willis (2010, p.29), their book is written in reaction to the previous literature on DevOps and sees a trend in DevOps that focuses on the outcomes, rather than on people and process. They also focus on the culture and processes because that will encourage iteration and improvement in how and why we do things.

Although Davis’ and Daniels’ definition (2016) is similar to Bass’s (2015) and to Dyck’s and others (2015), their focus is different, which will be shown below (in paragraph 3.2.5).

Apart from the literature that emerged, the DevOps momentum also reached training and certification institutes. One organization that focused solely on DevOps was the DevOps Digital Skill Association (DASA). DASA defines DevOps as follows:

DevOps, a philosophy, culture, and movement that arose from an urgent need for better alignment, collaboration, and empathy between IT Development and IT Operations teams or departments, is now increasingly used to denote precisely the aforementioned key ingredients that constitute the New IT wave.

For us, enterprise-wide DevOps stands for rethinking traditional IT practices and capabilities, including a product, process, and people perspective. DevOps is the ultimate search for flow in the delivery of IT services<sup>8</sup>.

This definition focuses largely on the culture of an organization, but neglects to mention automation, which is troubling because this organization is responsible for certifying people in DevOps. It is clear that the definitions given by DASA do not correlate to the DevOps principles that DASA itself has provided, which are:

- “a focus on customer-centric action;
- creation with the end in mind;
- establishing end-to-end responsibility;
- the creation of cross-functional autonomous teams;
- promoting continuous improvement; and
- automation of everything that can be automated.<sup>9</sup>”

These principles definitely overlap with aspects of the CAMS framework. Both speak about automation, culture, and measurement, but this is not shown in the definitions provided by DASA.

---

<sup>8</sup> <http://www.devopsagileskills.org/>

<sup>9</sup> <http://www.devopsagileskills.org/>

### 3.1.3 Main concepts covered

Two major improvements have heavily influenced DevOps.

The first is the development of Agile software, a development mentioned by several authors in their scientific work. The second is the formulation of a short description of the software delivery lifecycle, a change that became part of the questionnaire because this had a major impact on separating development and operations into individual teams with different key performance indicators (KPIs).

Soni (2015) and Peuraniemi (2015), in explaining the pillar of culture, state that DevOps has extended Agile methodologies from development into operations.

Peuraniemi (2015, p. 45) even paraphrases one of the values out the Agile manifesto and relates it to software development, pointing out that “individuals and interactions [should be emphasized] over processes and tools”.

As the research was being carried out, it became obvious that companies were beginning with the implementation with Agile methodologies for software development and then were extending this strategy to operations. Thus, it is clear that a very brief explanation of the Agile manifesto will assist in understanding the process of software development, even though this is not the core part of this thesis.

Agile principles for software development were released in 2001 with the release of the Agile Manifesto<sup>10</sup>. The manifesto was written by 17 authors, who were very experienced in software development. The aim of the Manifesto was to reveal better ways of developing software, by following the principle of “Doing it and helping others [to] do it”, that is, by promoting certain values over others.

The main focus of Agile is to continuously deliver software that is valued by the customer and, on the assumption that the requirements of the customer will change, to create a constant feedback loop for the customer during the software delivery lifecycle. This will create a better way to develop software on the basis of emphasizing the values on the left over the values on the right:

“Individuals and interactions	<b>Over</b>	Processes and tools”
“Working software	<b>Over</b>	Comprehensive documentation”
“Customer collaboration	<b>Over</b>	Contract negotiation”
“Responding to change	<b>Over</b>	Following a plan”

As Willis (2010) stated, “DevOps is not a plan, but a reaction”, and in the same blog he stated that the early practitioners of DevOps were inspired by Agile web-based companies, that is, they were running operations in an Agile fashion and in the same manner that they were running development teams. This

---

<sup>10</sup> <https://agilemanifesto.org/>

fact contrasts with those large enterprises that had apparently implemented Agile development processes, but whose operations were still running in a more traditional mode.

To understand the new Agile way of working, it is important to know what traditional limitations Agile methods were responding to. It is also crucial to bear in mind that the traditional way of working was inspired by the description of the software lifecycle created by Barry Boehm in his article “Software Engineering” (1976):

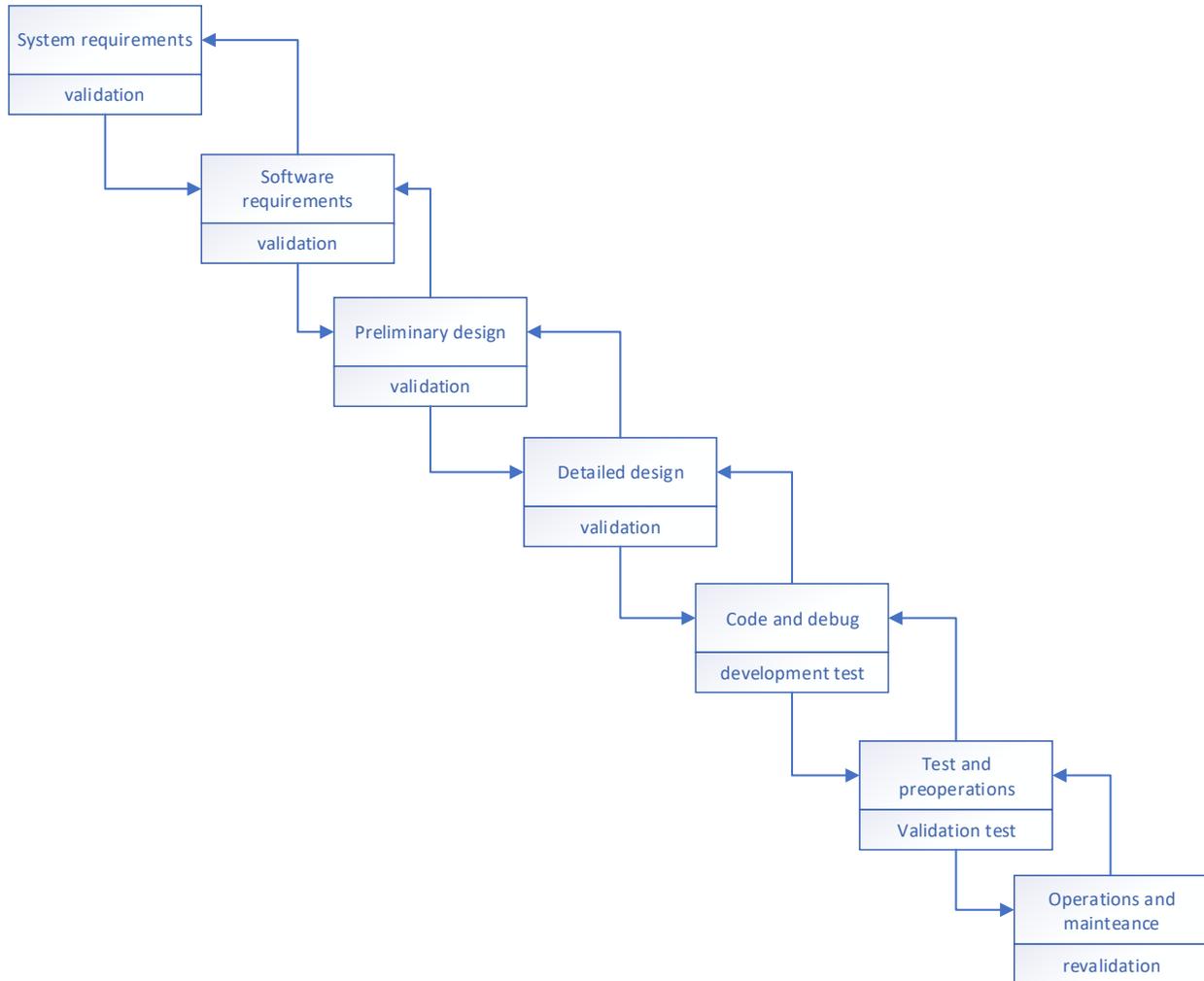


Figure 1: Software lifecycle, by Barry Boehm

Boehm (1976) describes the following stages as intrinsic to the software lifecycle: Specify system requirements; define software requirements; make a preliminary design; develop a detailed design; develop and debug codes; test what has been developed; conduct pre-operations procedures; send to operations personnel; and carry out maintenance. Further on in this article, he condenses these stages into the following phases: define requirements; design code; test development; carry out an acceptance test; and operate the software.

The work of Boehm has led to different silos being created inside software development, with their own responsibilities and tools.

If Agile software developers want to continuously deliver software that adds value to the customer, they should understand that that software should be running in operations. This kind of approach imposes the Agile principles being practised by development into operations, thus leading to a collaborative way of working between the two, one of the pillars of DevOps. The importance of this principle is also stressed by Jennifer Davis and Katherine Daniels in their book, *Effective DevOps* (2016), and is illustrated in table 3 below and in the next section.

		<u>The Phoenix Project</u>	<u>DevOps: A Software Architect's Perspective</u>	<u>The DevOps Handbook</u>	<u>Continuous Delivery and DevOps: A Quickstart Guide</u>	<u>Effective DevOps</u>	<u>Continuous Delivery</u>
<b>Culture</b>							
	Lean (value stream, cycle time, tact time)	X		X	X		
	Agile (people over process, working software over documentation)	X				X	
<b>Technology</b>							
	Automation	X	X	X	X		X
<b>Deployment</b>							
	Frequent	X	X	X	X		X
	Incremental	X	X	X	X		
<b>Measurement</b>		X	X	X	X		X
<b>Sharing</b>							
	Collaboration within IT between Dev and Ops	X	X	X	X	X	X
	Collaboration within IT between Dev, Ops, and security	X		X			
	Collaboration between business and IT	X					

Table 3: Overview of practitioner literature

## 3.2 Practitioner literature

This chapter explains important principles and concepts relevant to the implementation of DevOps, as developed by global specialists on this subject. Six highly influential books by experts are analyzed and the ideas contained in their works, their impact on the understanding of the CAMS framework, and their impact on DevOps are summarized in the tables.

### 3.2.1 *The Phoenix Project*, by Gene Kim, Kevin Behr, and George Spafford (2013)

The Phoenix Project has been portrayed in a book (by Gene Kim, Kevin Behr, and George Spafford, 2013) on DevOps thinking and is in the form of a novel, the purpose of which is to describe the rapidly changing world of IT; what DevOps does; and how businesses can emerge as winners in a competitive market. It chronicles the journey that its protagonist, Bill Palmer, Vice-President of Operations at Parts Unlimited, undertook to restructure IT operations and development in order to enable his business to gain a new competitive edge.

This journey was supported by an external advisor, Erik Reid, who explained that achieving DevOps Agility could be achieved by undertaking the following steps:

- The First Way involves the application of a left-to-right flow of work from development to IT, to operations, and then finally to the customer.
- The Second Way comprises the constant flow of fast feedback from right-to-left at all stages of the value stream, amplifying it to ensure that problems can be prevented from re-occurring and/or enabling faster detection and recovery.
- The Third Way involves the creation of a culture that fosters two things: continual experimentation—which requires taking risks and learning from success and failure—and understanding that repetition and practice are prerequisites to mastery (Kim, 2014).

The table below shows the link between the novel and the CAMS framework:

		<b>The Phoenix Project</b>
<b>Culture</b>		
	Lean (value stream, cycle time, tact time)	X
	Agile (people over process, working software over documentation)	X
<b>Technology</b>		
	Automation	X
<b>Deployment</b>		
	Frequent	X
	Incremental	X
<b>Measurement</b>		X
<b>Sharing</b>		
	Collaboration within IT between Dev and Ops	X
	Collaboration within IT between Dev, Ops, and security	X
	Collaboration between business and IT	X

Table 4: *The Phoenix Project*, by Kim, Behr, and Spafford

This novel about the Phoenix Project describes how Bill Palmer and Erik Reid visit a factory, during which Reid explains the theory of constraints to Palmer; subsequently, Reid suggests that Palmer apply this principle to his IT operations, all in the hopes of identifying bottlenecks and increasing productivity (Kim, 2013). The ideas that are mentioned are directly linked to Lean practices.

In the last part of the book, *Parts Unlimited*, the company where Bill Palmer works, starts to automate its deployments, resulting in its achieving 10 deployments per day. As a result, the company begins to implement the principle of frequent deployment; in addition, in the last part of the book, *Parts Unlimited* starts to work with a SWAT Team to deliver the most critical features first in order to respond quickly to market changes (Kim, 2014). This initiates a link to automation and results in frequent incremental deliveries.

Kim (2014) describes how this change-management procedure actually decreased the number of Sev1 outages (a production outage) by two-thirds and also resulted in a drop in incident recovery times (mean time to recover (MTTR)). In addition, a monitoring project was initiated and focused on collecting measurements.

By the end of the novel, Bill Palmer and Chris Allers (the VP of application development), discuss the IT problems they are experiencing and, at the end, agree that any automation project should involve collaboration between developers and operators. Thus, the founding principle of DevOps, which stresses that such collaboration is important, has been articulated.

During the Phoenix Project (2013), Bill Palmer and John Pesche, the CISO, sought to identify what the most crucial business elements were in the company. Based on the information supplied in this meeting, Pesche suspended a number of security-related actions because those controls concerned were already present on the business side and therefore were not needed in IT. At the end of the book, when the SWAT team decided to use the cloud as a platform to deliver services, Pesche offered to look at the security implications in order to help the team to make this move. All of these actions were clear signs of collaboration between the different teams, demonstrating not only that development and IT-operations had been combined, but that security operations had been integrated into both teams.

At the end of the book, the authors describe how important it is that those involved in IT form new SWAT teams which, in conjunction with the business side, will identify which features are most critical and thus should be delivered first, a tactic that will allow organizations to remain competitive. Another important development in the process described above is that the IT side of the company gained insights into the business side in order to support that sector. It also describes how, during a particular project, the sales manager put pressure on the Chief Operation Officer to release more funds to support IT during the monitoring project because such a step would help to develop a stable phone system, something which was needed to support the business. All of these developments were clear indications that the business and the IT sector, during the Phoenix Project, were developing successful cooperation.

### 3.2.2 *DevOps: A Software Architect's Perspective*, by Len Bass, Ingo Weber, and Liming Shu (2015)

The table shows the link between the book, *DevOps: A Software Architect's Perspective*, by Len Bass, Ingo Weber, and Liming Zhu (2015) and the CAMS framework:

		<b><u>DevOps: A Software Architect's Perspective</u></b>
<b>Culture</b>		
	Lean (value stream, cycle time, tact time)	
	Agile (people over process, working software over documentation)	
<b>Technology</b>		
	Automation	X
<b>Deployment</b>		
	Frequent	X
	Incremental	X
<b>Measurement</b>		X
<b>Sharing</b>		
	Collaboration within IT between Dev and Ops	X
	Collaboration within IT between Dev, Ops, and security	
	Collaboration between business and IT	

Table 5: DevOps: A Software Architect's Perspective, by Bass, Weber, and Shu

In their book, *DevOps: A Software Architect's Perspective* (2015), Bass, Weber, and Shu describe DevOps as a set of principles that are intended to reduce the time between implementing a change to a system and the change being transitioned into normal production, and in a manner that always ensures that high standards and quality are maintained.

The authors state that an organization's culture is important in the context of any discussions about DevOps; however, this book, although providing a chapter on deployment (which includes automation; monitoring, which is synonymous with measurement; and security), does not contain a chapter discussing culture. They briefly mention Agile, but their treatment of this subject area is very limited and few guidelines are given as to how to integrate Agile into DevOps. That is the reason why there is no link to the culture of the CAMS framework in the table.

The reason for a check in the automation column is the fact this book has a chapter entitled "Deployment", in which the authors describe the difference between continuous integration, continuous delivery, and continuous deployment.

Interestingly enough, the book does describe monitoring, which has the following five purposes: detecting a failure; diagnosing performance problems; planning capacity; obtaining data insights into user interactions; and detecting intrusion. This has more of a link to the work of Lkwatere (2016), but

monitoring is considered to fall under the pillar of automation in the CAMS framework by Humble (2011), another reason for a checkmark in automation.

Five different categories that represent key aspects that must be included when continuous delivery is being implemented are given (Humble & Farley (2011, p. 203)), which are: “culture and organization, design and architecture, build and deploy, test and verify, and information and reporting”.

According to the authors, even when all teams are using DevOps, the necessity will remain to separate operations and development. An important conclusion that can be made about their arguments is that they see DevOps as just another way of delivering software and that DevOps can be embedded in the current processes.

3.2.3 *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*, by Gene Kim, Jez Humble, Patrick Dubois, and John Willis (2016)

The table below shows the link between the book, *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*, by Gene Kim, Jez Humble, Patrick Dubois, and John Willis (2016), and the CAMS framework:

		<b><u>The DevOps Handbook</u></b>
<b>Culture</b>		
	Lean (value stream, cycle time, tact time)	X
	Agile (people over process, working software over documentation)	
<b>Technology</b>		
	Automation	X
<b>Deployment</b>		
	Frequent	X
	Incremental	X
<b>Measurement</b>		X
<b>Sharing</b>		
	Collaboration within IT between Dev and Ops	X
	Collaboration within IT between Dev, Ops, and security	X
	Collaboration between business and IT	

Table 6: *The DevOps Handbook*, by Kim, Humble, Dubois, and Willis

In *The Phoenix Project* (2014), Kim had already hinted that Bill Palmer would be writing a book describing how he had changed his department, and *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations* (2016) was that book. Although *The DevOps Handbook* incorporates Kim's description of the three vital principles of DevOps—flow, feedback, and continual learning and experimentation—Palmer describes companies that have already pioneered the implementation of DevOps practices.

In the first chapter of *The DevOps Handbook*, the authors link DevOps and Lean and introduce terminology related to a value chain, lead time, and processing time. Although the title of the chapter includes Agile, the principles are not described in this chapter nor is any mention of Agile made. That is the reason for a checkmark in Lean in the table, but not for Agile.

In part III, Kim (2016) introduces the concept of the deployment pipeline and establishes a relationship between automation and deployment. Kim stresses that the ideal batch size is a single item, and goes on to emphasize a key DevOps principle—the importance of an approach that emphasizes a frequent developmental deployment approach. This shows the fact that automation is addressed in this book.

In part IV, Kim (2016) describes the components inherent in relation to the technical practices of feedback. Kim (2016) emphasizes the importance of telemetry, that is to say, that it is important for businesses to receive feedback in the form of data, which can then be used to introduce improvements. Kim (2016) advises the application of a scientific method of testing, where the data is used to falsify a hypothesis. The results of this kind of testing leads to improvement and results in a company's outperformng its competitors. This is the reason for a checkmark in measurement.

In Part V, Kim (2016) analyses the importance of continual learning and experimentation; during this discussion, he stresses the need for sharing information between different departments and especially emphasizes the importance of maintaining security. He engages in a substantial discussion on development and operations, but crucially, fails to provide a detailed description and a set of practices that will facilitate collaboration between businesses and IT.

### 3.2.4 *Continuous Delivery and DevOps: A Quickstart Guide*, by Patrick Swartout (2012)

The table shows the link between the book, *Continuous Delivery and DevOps: A Quickstart Guide* by Paul Swartout (2012) and the CAMS framework:

		<b>Continuous Delivery and DevOps: A Quickstart Guide</b>
<b>Culture</b>		
	Lean (value stream, cycle time, tact time)	X
	Agile (people over process, working software over documentation)	
<b>Technology</b>		
	Automation	X
<b>Deployment</b>		
	Frequent	X
	Incremental	X
<b>Measurement</b>		X
<b>Sharing</b>		
	Collaboration within IT between Dev and Ops	X
	Collaboration within IT between Dev, Ops, and security	
	Collaboration between business and IT	

Table 7: *Continuous Delivery and DevOps: A Quickstart Guide*, by Swartout

Swartout, in his book describes how ACME, a fictitious company, implemented continuous delivery and DevOps, two critical ideas that he continually emphasizes. Swartout describes continuous delivery as a method of delivering fully working software in small incremental chunks to the production platform; he uses the term “DevOps” as describing a way of working that encourages development and operations teams to work together in a highly collaborative fashion in order to reach the same goal. This is the reason for checkmarks in the automation and collaboration columns.

Although Swartout (2012) does refer to Agile, he only mentions certain aspects such as “inspect” and “adapt”; in other words, he does not mention principles that are key components of the Agile Manifesto, but rather, emphasizes concepts that are in line with the principles of Scrum, and states that they form the core of Agile, which is not the case. In light of that fact, it cannot be said that Swartout emphasizes Agile principles in his book.

The software development life cycle of ACME is described in terms of a value chain, and Swartout (2012) describes cycle time and the tact time between work centres in the value chain. That is the reason for the placing of a checkmark in Lean practices in the table.

In a chapter about IT tools, Swartout (2012) describes a process of automation that uses a deployment pipeline, something which facilitates collaboration and automated testing, while at the same time implementing test-driven development. This justifies a checkmark in the automation column.

Swartout (2012) states that the goal of continuous delivery and DevOps at ACME is to deliver software 10 times per day during production, which is generally considered to constitute “frequent delivery”. He goes on to stress that working by means of the provision of small, incremental changes is beneficial to an organization. Two other principles he considers to be crucial are monitoring of the environment and frequency of deployment.

Swartout asserts that encouraging an open, honest, and courageous dialogue among staff members is critical to success, and thus he spend a considerable time discussing culture and behaviours.

Finally, Swartout (2012) states that establishing collaboration between development and operations personnel will lead to continuous delivery and the implementation of DevOps; however, one key component missing in his discussion is the importance of setting up security measures during this process. Although Swartout states that this will have an impact on the business, he does not provide a set of practices guiding this cooperation, something which is a definite drawback.

### 3.2.5 *Effective DevOps*, by Jennifer Davis and Ryn Daniels (2016)

The table shows the link between the book, *Effective DevOps*, by Jennifer Davis and Ryn Daniels (2016) and the CAMS framework:

		<b><u>Effective DevOps</u></b>
<b>Culture</b>		
	Lean (value stream, cycle time, tact time)	
	Agile (people over process, working software over documentation)	X
<b>Technology</b>		
	Automation	
<b>Deployment</b>		
	Frequent	
	Incremental	
<b>Measurement</b>		
<b>Sharing</b>		
	Collaboration within IT between Dev and Ops	X
	Collaboration within IT between Dev, Ops, and security	
	Collaboration between business and IT	

Table 8: *Effective DevOps*, by Davis

Davis and Daniels (2016), in their work *Effective DevOps*, define DevOps as a cultural movement, one which changes the way individuals think about their work; which values the diversity of the work done; which supports the intentional processes that accelerate the rate by which businesses realize value; and which measures the effect of social and technical change.

According to the authors (2016), there are four pillars that need to be put into place in order to make the implementation of DevOps effective: these are collaboration, affinity, tools, and scaling.

Although they raise the idea of identifying ways of measurement, it is only in the tools section that they mention this topic, and only in conjunction with monitoring. But measurement in the CAMS framework

is not just a matter of tooling, but also entails the assessment of processes. As a result of this missing element, Davis and Daniels (2016), do not give the topic the importance it deserves. Another glaring omission in the argument is the fairly meagre discussion about feedback loops, something which Kim covers in more detail in his works (Kim 2014, 2016).

However, Davis and Daniels(2016) discuss in detail the importance of culture, that is, of the relationships that exist between people and teams. The importance that they attribute to this principle is indicated by the fact that they spend about 75% of the book focusing on this concept, which shows that they consider people to be more important than processes; they underscore this belief in this by pointing out that DevOps encourages critical thinking, another “people” skill.

Davis and Daniels (2016) do not spend much time analyzing either the development or the deployment process. However, they do mention the vital importance of continuous integration and continuous delivery, specifically in the introductory chapter and in the section on tooling. However, these concepts are not given the prominence that the discussion on culture has.

What is of crucial importance in the discussion is that they emphasize that DevOps should transcend the usual silos, which are described by Boehm (1976), and that its implementation should not be limited to development and operations.

The fact that the book so prominently focuses on people and processes enables the columns of “Agile” and “collaboration” to be checked.

3.2.6 *Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation* (2011), by David Farley and Jez Humble

The table shows the link between the book, *Continuous Delivery: Reliable Software Release through Build, Test, and Deployment Automation*, by Jez Humble and David Farley (2011), and the CAMS framework:

		<b><u>Continuous Delivery</u></b>
<b>Culture</b>		
	Lean (value stream, cycle time, tact time)	
	Agile (people over process, working software over documentation)	
<b>Technology</b>		
	Automation	X
<b>Deployment</b>		
	Frequent	X
	Incremental	
<b>Measurement</b>		X
<b>Sharing</b>		
	Collaboration within IT between Dev and Ops	X
	Collaboration within IT between Dev, Ops, and security	
	Collaboration between business and IT	

Table 9: *Continuous Delivery*, by Farley and Humble

Farley’s and Humble’s book states that a deployment pipe, that is, the use of an automated release process, is the core principle shaping DevOps success.

In this book, the authors set forth the idea that their preferred development methodology is Agile, but that the book was written to encourage the use of the “waterfall” methodology of development. They state that the goal of software professionals is to deliver useful, working software to users as quickly as possible.

In order to do this, the development process should be automated and developers should consult with operations people on an informal basis and involve them in the development process when a project is in its initial stages. This supports the view that a focus on collaboration is one of the central principles of the DevOps movement.

The authors also make the point that encouraging a productive collaboration between development and operations requires an understanding of the important role that monitoring plays in this process, especially in terms of operations. They state that operational measurements should assess the time needed for repairs and mean times between failures. From an IT perspective, information is needed in relation to restoring point objectives and to recovery time objectives.

The last part of the book gives the following maturity model for continuous delivery:

Practice	Build management and continuous integration	Environments and deployment	Release management and compliance	Testing	Data management
<b>Level 3 - Optimizing:</b> Focus on process improvement	Teams regularly meet to discuss integration problems and resolve them with automation, faster feedback, and better visibility.	All environments managed effectively. Provisioning fully automated. Virtualization used if applicable.	Operations and delivery teams regularly collaborate to manage risks and reduce cycle time.	Production rollbacks rare. Defects found and fixed immediately.	Release to release feedback loop of database performance and deployment process.
<b>Level 2 - Quantitatively managed:</b> Process measured and controlled	Build metrics gathered, made visible, and acted on. Builds are not left broken.	Orchestrated deployments managed. Release and rollback processes tested.	Environment and application health monitored and proactively managed. Cycle time monitored.	Quality metrics and trends tracked. Non functional requirements defined and measured.	Database upgrades and rollbacks tested with every deployment. Database performance monitored and optimized.
<b>Level 1 - Consistent:</b> Automated processes applied across whole application lifecycle	Automated build and test cycle every time a change is committed. Dependencies managed. Re-use of scripts and tools.	Fully automated, self-service push-button process for deploying software. Same process to deploy to every environment.	Change management and approvals processes defined and enforced. Regulatory and compliance conditions met.	Automated unit and acceptance tests, the latter written with testers. Testing part of development process.	Database changes performed automatically as part of deployment process.
<b>Level 0 – Repeatable:</b> Process documented and partly automated	Regular automated build and testing. Any build can be re-created from source control using automated process.	Automated deployment to some environments. Creation of new environments is cheap. All configuration externalized / versioned	Painful and infrequent, but reliable, releases. Limited traceability from requirements to release.	Automated tests written as part of story development.	Changes to databases done with automated scripts versioned with application.
<b>Level -1 – Regressive:</b> processes unrepeatable, poorly controlled, and reactive	Manual processes for building software. No management of artifacts and reports.	Manual process for deploying software. Environment-specific binaries. Environments provisioned manually.	Infrequent and unreliable releases.	Manual testing after development.	Data migrations unversioned and performed manually.

Figure 2: Maturity Model for Continuous Delivery, by David Farley and Jez Humble

### 3.2.7 Conclusions

Both in the academic world and in the practitioner world, a great deal of discussion is taking place on how to define DevOps. Each practitioner has his/her own definition and his/her own focus on important aspects of this phenomenon. However, one thing is clear (as demonstrated in the initial paragraphs in this section): It is Gene Kim and his fellow authors (2013, 2016) that succeed in addressing all the relevant aspects of DevOps. Among practitioners, Kim is seen as the major influence on thinking about this subject.

It is clear that Dyck and et al’s definition of DevOps (2015) and the CAMS framework provide a sound basis for the analysis that is contained in this thesis.

### 3.3 Maturity models

This section elaborates on the importance of maturity models and explains: (1) where they originated from (their origins); (2) how they can be defined; and (3) what the main concepts in relation to maturity models and DevOps are (as described in the scientific literature).

#### 3.3.1 Introduction

Jörg Becker, Ralf Knackstedt, and Jens Pöppelbuß (2009), in their article “Developing Maturity Models for IT Management: A Procedure Model and Its Application”, describe the Capability Maturity Model (CMM) as a well-known maturity model for improving software development. This model was developed by the Software Engineering Institute (SEI) of the Carnegie Mellon University in Pittsburgh, Pennsylvania, United States of America.

Paulk (2009) describes a set of releases of this model: A version 1.0 released in 1991 and a version 1.1 released in 1993. In 2003, CMM was retired in favor of CMM Integration (CMMI) (Paulk, 2009). Paulk (2009) also states that CMM inspired other standards, such as People CMM, the Systems Engineering CMM, and the Systems Security CMM.

Clerc and Niessink (2004) state that, in 1995, a considerable number of methods and tools existed to support and improve software development, but that the field of IT Service Management had received less attention.

According to Clerc and Niessink (2004), as a reaction to the fact that the field of IT Service Management Maturity had been emphasized to a lesser degree, the first version of IT Service CMM was released in 1999 as part of the Kwintes Project. The Kwintes Project was a research project supported by the Dutch Ministry of Economic Affairs, the aims of which were to develop methods for specifying and controlling IT Services<sup>11</sup>.

Philips and Shrum (2010) provide an account of the history of CMMI for Services, detailing the first release in 2009 of version 1.2 and an expected release of version 1.3 in November of 2010 (CMMI-SVC, a comprehensive set of guidelines to promote superior services, version 1.3, was released in 2010).

According to CMMI’s website, the current release of CMMI for Services is 2.0.<sup>12</sup>

#### 3.3.2 Scientific DevOps maturity models

McCarthy, Herger, Khan, and Belgodere, in their article, “Composable DevOps: Automated Ontology Based DevOps Maturity Analysis” (2015), state that DevOps blurs the traditional boundaries between developers and IT by ideally bringing together development and operations to achieve a common goal. In line with this thinking, they created a DevOps maturity model that is based on composable DevOps

---

<sup>11</sup> [http://www.survuz.com/frameworks/it-service-cmm-it-service-capability-maturity-model#.XcgT\\_W5Fx9N](http://www.survuz.com/frameworks/it-service-cmm-it-service-capability-maturity-model#.XcgT_W5Fx9N)

<sup>12</sup> <https://cmmiinstitute.com/cmmi>

and deontic constraints, that is, on reasoning based upon deontic logic, a field of philosophical logic that is concerned with obligation, permission, and related concepts.<sup>13</sup>

Constraints Features		
Phases	Description	Modality
Dev/Test	Developers obligated to create a development sandbox that closely matches production - even when physical resources are constrained for all dependent systems	Obligation
Dev/Test	Developers must have access to the operations / support team incident details and conduct formal/informal meetings to improve application	Obligation
Dev/Test	Testing teams must share their validation mechanisms and strategies with Development and Operations	Obligation
Dev/Test	There must be an agreement between business, dev and ops on critical services (transaction counts, performance, uptime etc.) that are necessary to meet predefined business goals	Obligation
Dev/Test	Developers must have a centralized portal in order to access, develop and test API's as they code their applications	Obligation
Dev/Test	Development teams must use automated testing across the software development lifecycle	Obligation
Release/ Deploy	Team must provide overall visibility into your application release activities and timing to all major stakeholders	Obligation

<sup>13</sup>

[https://en.wikipedia.org/wiki/Deontic\\_logic](https://en.wikipedia.org/wiki/Deontic_logic)

Release/ Deploy	Application release deployments must be fully automated end-to-end across development, test and production environments	Obligation
Release/ Deploy	Teams must be able to provide selfservice, on-demand provisioning and management of cloud environments and infrastructure resources	Obligation
Release/ Deploy	Team must be able to speed lead times and make more frequent application deployments at the pace demanded by the business	Obligation
Operate /Assure	Number of tools required to monitor network, systems, databases etc. must be consolidated to reduce complexity and speed time to problem resolution	Obligation
Operate /Assure	Teams must have the visibility and insight you need to ensure reliable business service delivery for the applications and business services that have moved to third-party cloud providers	Obligation
Operate /Assure	Teams must have visibility into capacity in order to understand the impact on infrastructure when there are increases in workload, transactions, application upgrades, and hardware refreshes	Obligation
Operate /Assure	Teams must have access to feedback information on mobile app behavior and usage analytics to drive improvements	Obligation
Operate /Assure	Teams must automate the feedback of critical performance information across stages of the application lifecycle so as to further optimize applications and services	Obligation

Table 10: DevOps deontic constraints (as defined by Mc Carthy and others)

Although this article describes a maturity analysis and provides the deontic constraints for analysis, it does not deliver a detailed maturity model that would fulfill the requirements for a capability maturity model. The authors state that each and every component of DevOps follows its own path to maturity, meaning that any element of coercion would lead to disastrous effects.

The article proposes a meta-model driven solution, and therefore this model could not be used as an assessment tool during this research.

Fortunately, Samer I. Mohamed (2015), in his article entitled “The DevOps Maturity Calculator DOMC – Value oriented approach”, in the *International Journal of Engineering Research and Science*, proposes a DevOps maturity model based upon CMMI and which describes five levels of maturity. Each level has four dimensions:

- Quality
- Automation
- Communication/collaboration
- Governance

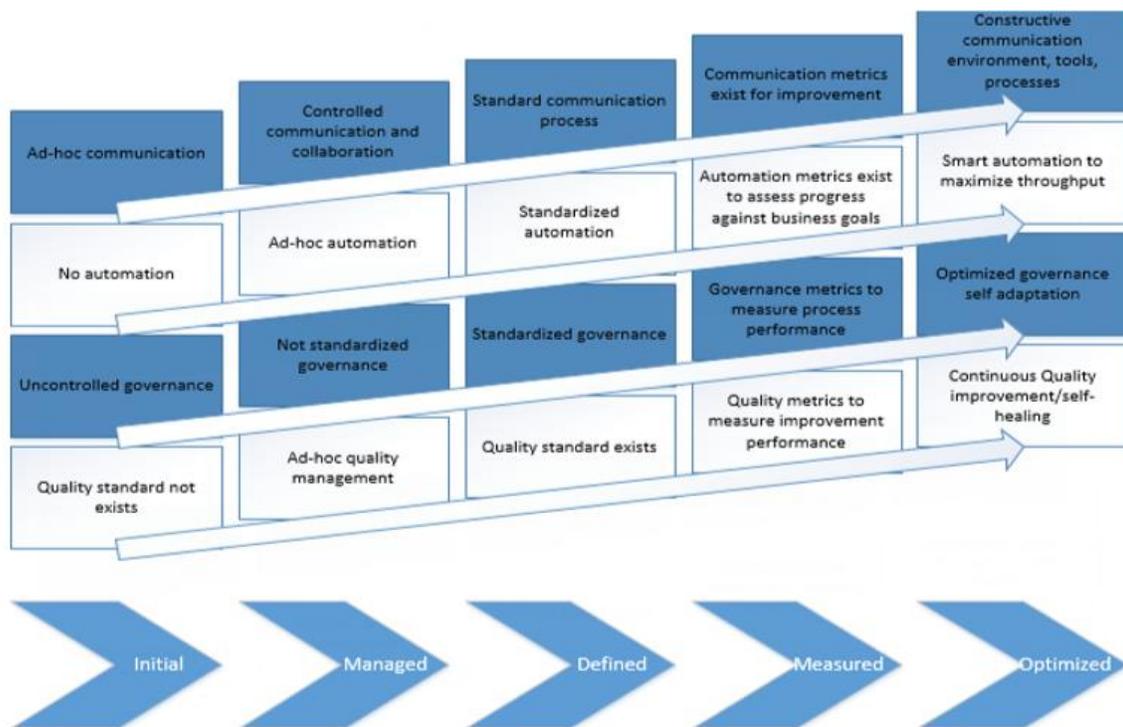


Figure 3: DevOps maturity levels, as defined by Samer I. Mohamed

This model provided the researcher with the opportunity to compare the different companies that were interviewed and provided the basis for the analysis that emerged.

### 3.3.3 Practitioner DevOps maturity models

Scientific research provides only one maturity model that was useful for comparing the various companies that were analyzed during this research, and thus resulted in a very limited assessment. For that reason, it was advisable for the researcher to refer to the practitioner literature to see if there were suitable maturity models that could be used to compare the different companies.

The consultancy firm of Gartner had already been referenced during discussions about the definitions of DevOps, and so another consultancy firm was accessed to see if it had developed a DevOps maturity model.

Kavis (2017) describes the following four stages of DevOps Maturity on the Forbes website<sup>14</sup>:

- “Stage 1: DevOps denial and misinterpretation;
- Stage 2: automation for the sake of automation;
- Stage 3: collaboration and reorganization; and
- Stage 4: a high-performing organization.”

Although he (Kavis 2017) delivers a maturity model, this model does not provide detailed information in relation to the key process areas nor to the processes that have to be implemented. So, this model is of little use in carrying out comparisons.

Another option is to look at vendors that have a track record in support of development teams and, in that context, Atlassian, with their solution JIRA has a large installed base inside Agile development teams.

Atlassian (2019), in “DevOps Maturity”, delivers an assessment and describes four levels of maturity<sup>15</sup>:

- base;
- beginner;
- advanced; and
- expert.

Atlassian posed the following multiple-choice questions to assess an organization’s DevOps maturity:

1. “How would you describe the organizational and team structure for developing your products?”
2. “How would you describe how your dev and ops teams share tools?”
3. “How would you describe the project prioritization process inside your company?”
4. “What best describes your testing environment?”
5. “What is your organizational structure around service delivery?”
6. “How would you describe your typical release process of key projects?”
7. “How often do you deploy code into production?”
8. “In general, how does your application perform once released into production?”
9. “What is the ratio to unplanned/bugfix work vs new features/improving infrastructure?”

---

<sup>14</sup> <https://www.forbes.com/sites/mikekavis/2017/11/17/the-four-stages-of-devops-maturity/#57e8>

<sup>15</sup> <https://www.atlassian.com/devops/maturity-model>

10. “How quickly can developers see the results of the integration/unit/etc. testing?”

Although these questions relate to the different aspects of the CAMS framework, there is no description as to which answers correlate to which maturity level, so for this thesis, this maturity model cannot be used to compare the different companies.

Another company delivering products for DevOps is Micro Focus (2019), and they describe a five-stage maturity model based on CMMI, with a focus on the following dimensions:

- people;
- process; and
- technology.

Maturity Model	1 Initial	2 Managed	3 Defined	4 Measured	5 Optimized
<b>People</b> 	Ad-hoc communication & coordination Silos still in place	Managed communication Shared decision making Ad-hoc collaboration Specialized people Some agile leaning	Shared accountability Dev and Ops teams inside Release Trains Some cross-functionality Lean-Agile disposition expressed	Collaboration as culture Integration roles between Dev & Ops standard Cross-functionality prevalent Lean-Agile mindset prevalent	Optimized for DevOps Shared Ops teams formal Knowledge sharing & individual empowerment High cross functionality Lean-Agile leaders with a collaborative mindset
<b>Process</b> 	Uncontrolled or reactive processes Predominantly waterfall	Unstandardized processes Dev & Ops Processes separately owned & managed Some agile adoption and iterative practices	Processes standardized Common KPIs Agile & Kanban in dev & ops	Visibility & predictability of entire process Continuous practices in a DevOps pipeline Lean-Agile approach scaled across the enterprise	Process risk & cost optimization Highly optimized processes Continuous assessment impacts optimization Agile\Lean drives process optimization
<b>Technology</b> 	Little or no automation	Siloed automation for Dev & Ops tools & practices No central infrastructure Some collaborative tools	Centrally automated & integrated Infrastructure as code exists Managed test environments Collaborative tools standardized	Automated metrics collected & analyzed against business goals Automated prod release Automated reuse & remediation Infrastructure as code e2e	Self-service automation for all infrastructure Analytics enables self & remediation DevOps pipelines are deployable & integrated across the organization

Figure 4: Micro Focus DevOps maturity

This model is a model that could be used to compare the progress of DevOps maturity within various companies that were the subject of the research.

Poepelbuß and Röglinger (2011, p. 3) describe the importance of the development of a maturity model as follows:

- “Descriptive: A maturity model serves a descriptive purpose if it is applied for as-is assessments, where the current capabilities of the entity under investigation are assessed with respect to given criteria.”

- “Prescriptive: A maturity model serves a prescriptive purpose if it indicates how to identify desirable maturity levels and provides guidelines on improvement measures.”
- “Comparative: A maturity model serves a comparative purpose of use if it allows for internal or external benchmarking.”

This thesis focuses on the uses of maturity models that serve comparative purposes.

## 4. Research method

This chapter describes the methodology of this research and the design and data used in this study.

### 4.1 Research strategy and design

The purpose of this research was to contribute to the understanding of DevOps and to investigate whether enough evidence exists to prove that the implementation of DevOps will enhance an organization's success.

DevOps is a young domain and there is little scientific information available in relation to it. What's more, the scientific information that is available for this domain tends to focus almost exclusively on definitions of DevOps and its dimensions. There is only one scientific article that generally discusses maturity models and DevOps, and at this point in time, there is no scientific information available about whether DevOps implementation contributes to success. There is a lot more material/publications available in practitioner literature; that material tends to focus on definitions of maturity. In short, very little discussion has taken place on whether DevOps is successful and why: Many individuals have given presentations touting the success of DevOps, but have done so without providing the requisite evidence and data.

In order to gain a better understanding of the subject, the research described in this thesis examined the choices available and how they influenced this investigation. In order to describe a relationship between real-world observations and scientific research, Jonker and Pennink's (2004, 2009) "box of bricks of research" was used as a guide as to how scientific research should be performed. This method gave guidance and structure during the different phases of this research.



Figure 5: Box of bricks of research

In other words, the following elements, what Jonker and Pennink (2004, 2009) call "the research pyramid", constituted the approach taken during this enquiry:

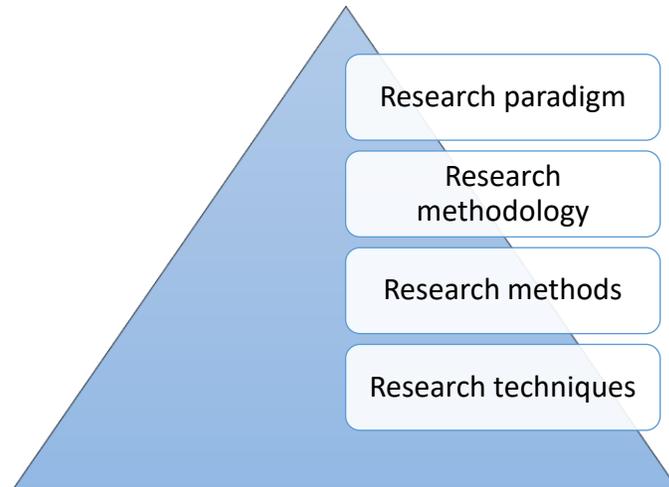


Figure 6: The research pyramid

This pyramid states that researchers will proceed from more abstract ideas into concrete research (which is found at the bottom of the pyramid.)

#### 4.1.1 Research paradigm

There is very little information available on DevOps and success. In addition, multiple possibilities exist as to how research into this topic could be conducted. One of the major influences on the choice of the method used is based on examining the bias, or as Jonker and Pennink (2009) call it, the “basic approaches” of the researcher. Jonker and Pennink (2009) describe two major “basic approaches”:

1. Knowing through the eyes of the researcher: The researcher begins his/her investigation with an image of the empirical reality already in mind.
2. Knowing through the eyes of someone else: The researcher depends on people working in the environment he wants to investigate to gain profound knowledge of that environment.

An application of the first basic attitude, knowing through the eyes of a researcher, requires that a common body of knowledge should be available about the subject. Of course, there is material available on DevOps, but there is very limited information available on DevOps and maturity, let alone DevOps and success.

For that reason and on the basis of Jonker’s and Pennink’s theory (2009), the researcher could only use the second, the latter approach, which is based on knowing through the eyes of someone else.

However, this thesis reflects a different approach, one that is not theoretical, but concrete; in other words, that success should be measurable. The search for concrete indicators of performance had a huge impact on the research reflected here. As a researcher, I wanted to prove that there are external indicators that proved the success of DevOps implementation. So I used the research philosophy that Saunders and others, in their book *Research Methods for Business Students* (2016), would call “positivism”:

Saunders et al (2016, p. 145) have created the following table to guide the process from reason to research:

	<b>Deduction</b>	<b>Induction</b>	<b>Abduction</b>
“Logic”	“In a deductive inference, when the premises are true, the conclusion must also be true.”	“In an inductive inference, known premises are used to generate untested conclusions.”	“In an abductive inference, known premises are used to generate testable conclusions.”
“Generalizability”	“Generalizing from the general to the specific”	“Generalizing from the specific to the general”	Generalizing from the interactions between the specific and the general
“Use of data”	“Data collection is used to evaluate propositions or hypotheses related to an existing theory.”	“Data collection is used to explore a phenomenon, identify themes and patterns, and create a conceptual framework.”	“Data collection is used to explore a phenomenon, identify themes and patterns, locate these in a conceptual framework, and test this through subsequent data collection and so forth.”
“Theory”	“Theory falsification or verification”	“Theory generation and building”	“Theory generation or modification incorporating existing theory where appropriate, to build new theory or modifying existing theory.”

*Table 11: Research strategies by Saunders and others*

Although there is very little information available on the subject this thesis investigated, the abductive method was used to come to testable conclusions.

#### 4.1.2 Research methodology

Saunders et al (2016) describe methodological choices relevant to research, which include the following options:

- quantitative research;
- qualitative research; and
- mixed-method research.

Since the research reflected in this thesis is based in a line of reasoning involving abduction, it lends itself to a “mixed-method” methodological approach. Thus, the research and analysis involve a combination of both quantitative and qualitative data collection techniques. In light of this, this thesis is based on embedded mixed research, a fact that is reflected in the questionnaire that was used, where quantitative questions were mixed with qualitative questions. In addition, the investigation looked at external quantitative data in order to support conclusions in relation to success factors. The survey clearly showed that the people interviewed were interested in achieving transformation, and so it might be argued that their perception of the success of DevOps could have been influenced by this fact. However, external data can provide some measure of validation in terms of any success.

#### 4.1.3 Research methods

As described above, this study was based on a mixed-method approach to research, where qualitative data about the vision for DevOps implementation was first gathered among the various companies studied. During the interviews, the researcher also asked for interviewees to provide quantitative data. After that, information was sought from external sources to see if any additional metrics of business success were available. In that context, several sequences of gathering data were carried out, involving an approach where the first qualitative methods were applied, followed by the application of quantitative methods. Saunders et al (2016) describe this as constituting a sequential exploratory research method.

#### 4.1.4 Research techniques

Jonker and Pennink (2010) describe a number of research techniques that can be used to gather and interpret data. The research carried out here involved multiple case studies. In order to define success in terms of DevOps implementation, what needed to be examined were companies that shared the same market and the same business; what was also required was that they had implemented DevOps in a series of stages. Such an approach ensured that the companies could be accurately assessed and compared in terms of their maturity in DevOps and their business success.

The case studies were based on semi-open interviews that were recorded; subsequently, the transcriptions were sent back to the interviewees for confirmation as to their accuracy. In each company, more than one person was interviewed in order to avoid the possibility of bias.

Data collection took place during face-to-face semi-structured interviews that made use of open-ended questions. The interviews took place in the period from 2017 to 2018, with 12 different people from four different companies.

In addition to the interviews, different external data sources have been used, enabling interview results to be compared with external quantitative data, thus enriching the analysis of the case studies and facilitating the extraction of different metrics that indicated whether DevOps had been successful. These external data sources included annual reports that contained information about NPS scores; mobile banking index data provided by banken.nl (2018) (which gave information about the features delivered by the mobile applications); the trust index from the Dutch Banking Association, which validated

customers' trust in the online services delivered; and the data from the IOS App Store indicating release frequency and the app ratings.

The reason that the financial sector was chosen for this study stemmed from repo, "Digital Vortex 2019", written by Tomoko Yokoi, Jialu Shan, Michael Wade and James Macaulay (2019) for the International Institute Management Development<sup>16</sup> In their report, they introduce the concept of a digital vortex. The closer an industry is to the digital vortex, the bigger the need is for it to reinvent itself. This reinvention will have a major impact on the velocity and the magnitude of change. In 2017, the financial industry was at position 4 and in 2019 the financial industry was at position number 5 (Yokoi and others, 2019). This, in combination with the quote from Peter Jacobs, that "ING is a software company with a banking licence", inspired the researcher with the idea that, because IT was having a major impact on the financial industry, DevOps would deliver significant changes to this industry.

All the interviewees were directly approached by the researcher and were requested to participate in this research.

#### *4.1.4.1 Company selection criteria*

There were a number of criteria that were applied when the researcher began his research. Some of these criteria were, that:

- the companies should be involved in the financial industry in the Netherlands, with a focus on retail banking;
- the companies should utilize mobile applications, that is, they should deliver online services to their customers; and
- since this research was examining IT development, the companies should have an IT department staffed with their own developers.

All the companies were chosen because of their focus on retail banking. They also had adopted a mobile interface towards their customers, thus demonstrating the significant impact of IT. This would indicate that they had adopted some elements of DevOps implementation. In that context, the researcher was able to interview people about how their organization had transitioned to DevOps and to ask whether the implementation had been successful.

---

<sup>16</sup> <https://www.imd.org/contentassets/d4b328f064c844cd864a79369ba8405a/digital-vortex.pdf>

#### 4.1.4.2 Interviewee selection criteria

In order to have a complete view within each company, the researcher selected for interviewing at least one manager and one person working in operations. Both needed to be working in IT-related areas and the interviewees needed to have been involved in implementing DevOps.

#### 4.1.4.3 Interviews

	Company A	Company B	Company C	Company D
<b>Interviewees</b>	4	3	2	3
<b>Managers</b>	2	1	2	2
<b>Team Members</b>	2	2	0	1
<b>Number of different teams interviewed</b>	4	2	2	2
<b>Average years of experience with DevOps</b>	2.4	7.3	6	1.7

The interviews took place face-to-face on premises of the person being interviewed and took about one hour.

The interviews were structured around five major subject areas:

- an introduction to the interviewee and his/her work;
- the interviewee's perception about DevOps;
- the state of the development and operations sectors before the implementation of DevOps;
- how DevOps, after implementation, had changed how development and operations functioned; and;
- measurement/assessment of the impact of the implementation of DevOps.

More details about the roles of the participants at the time of the interviews can be found in the different data sections reporting the results that emerged from the respective companies.

#### 4.1.4.4 External data collection

After the interviews were conducted, the external data was collected using different data sources on the internet, including the following:

- annual reports of the different companies, especially in terms of their Net Promotor Scores (NPS);
- the "vertrouwensmonitor" (confidence monitor), published by De Nederlandse Vereniging van Banken (the Dutch Banking Association);

- information collected from banken.nl in relation to mobile apps for the different Dutch banks<sup>17</sup>; and
- Apple Information from the Apple App Store, showing the release frequency and the app ratings.

The NPS is a management tool, which reveals the loyalty of the customers of a firm. This metric claims to be correlated with revenue growth<sup>18</sup>.

In a survey, customers answer the following question: “How likely is it that you would recommend our company/product/service to a friend or colleague?”. Usually, they answer this question on a 0 to 10 scale.

A respondent that scores this question with a 9 or a 10 are called “promoters”, and they are said to buy more, to be more loyal, and to give more referrals. Customers that respond with a score of 0 to 6 are called “detractors”, and they are considered to be non-loyal customers, to not purchase more products, and to not refer the company to anyone else. Respondents that score a 7 or 8 are called “passives”. The NPS is calculated by subtracting the percentage of customers who are detractors from the percentage of customers who are promoters.

Due to the fact that this is a standard and claims to have a direct impact on revenue, it has been used as an indicator for customer success in this research. It can also be used as an external comparison metric.

The second external monitor is not based on the score for the company as a whole, but is focused on the measurement of online services. This metric was chosen because online services will be directly impacted by IT, and therefore DevOps transformation could have a direct impact on this metric. The confidence monitor standardized this metric, and measured, and published it over a longer period of time.

The third external metric is based on the mobile application services of the company. This site has delivered two measurements, mostly in relation to the period that the companies were transitioning to DevOps. They created a standard of 90 functionalities that the application should deliver, and the applications were measured against this standard. Because this metric provides an indication if the company after the transition to DevOps is delivering more functionalities, it provides a mechanism to compare the number of features delivered by the different companies.

The fourth metric focuses on individual applications and the direct perceptions of the users of the applications. The information from the Apple App Store provided an external source on the release frequency, so it was used to check the internal information gleaned from the interviews. In addition, the app rating gives a view to how the application is perceived by the customers from the bank, and can, therefore, be seen as a metric assessing the success of that application.

---

<sup>17</sup> <https://www.banken.nl/nieuws/21000/mobiel-bankieren-apps-spelen-steeds-meer-in-op-wensen-van-consumenten>

<sup>18</sup> <https://en.wikipedia.org/wiki/NetPromoter>

## 5. Data collection

	Company A	Company B	Company C	Company D
<b>Interviewees</b>	4	3	2	3
<b>Year of DevOps Implementation</b>	2016	2012	2016	2018
<b>Full Time Equivalent (FTE)</b>	15,576	13,897	41,861	3,155

Table 12: Data collection overview

### 5.1 Company A

Company A is a major retail bank in the Netherlands that has a 15,576 FTE in the Netherlands. Four different individuals from this bank were interviewed:

- One interviewee was at the time of the interview a DevOps engineer, with a background in operations. He had been in this role for only six months at the time of the interview and called himself an “Ops Developer”.
- Another interviewee was an avid promoter of DevOps at that time; his role was to help new teams implement DevOps inside the bank. At the time of the interview, he had been in his role for three years.
- Another interviewee was responsible for the tooling in relation to the continuous integration/continuous development (CI/CD) pipeline. He had started working for the bank in connection with DevOps three to four years previously.
- The last interviewee was at the time of the interview a manager at the bank. In his management role, he was responsible for introducing DevOps implementation.

All the people interviewed worked in different teams at the bank at the time of the interview.

#### 5.1.1 What was the internal definition of DevOps used by the bank?

The common denominator in the answers given by the interviewees was that DevOps is a combination of development and operations. One person answered: “You build it, you run it, you fix it”. Another stated that the team was responsible from concept to cash or, in other words, “You think it, you build it, you run it”.

This company had implemented DevOps in a manner that combined the various roles that IT played within an organization inside one team. That new team consisted of the following members and used SCRUM methodology; in particular, the following were part of the team:

- product owners;
- developers; and
- operators.

Besides that, the following set of practices and guidelines were applied:

- automation software;
- accountability procedures;
- standards to ensure that the team performed to a high standard; and
- continuous improvement.

### 5.1.2 The situation before DevOps

The bank had outsourced a lot of work to third-party suppliers, and thus faced a situation where there were a lot of handovers. One of the interviewees explained that he had carried out a test involving the deployment of a simple programme called “Hello World”, which printed the words “Hello World” on the screen. It took six months to get this programme into production. For him, this demonstrated that it took far too long to implement a new functionality.

Many interviewees described difficulties related to handovers to third parties, and also expressed frustration about the lack of feedback. One actually stated that, although new programmes were being used, they were uncertain as to which issues they solved or whether, indeed, they were solving an issue that no longer existed because business requirements had changed. One interviewee described the following situation: “Our concept to cash turnaround time was longer, and we sometimes had a solution for a problem we did not understand properly. So this was not a real solution to the business”.

Development and operations were separate teams measured according to different criteria. One interviewee stated: “ IT Development was measured in relation to a new functionality, so change was important. IT Operations was measured on stability, so change was a risk”. One drawback was the fact that the sharing of knowledge occurred on an individual, and not on a group, basis. A culture of blame existed between the two groups: Operations employees were seen as bureaucrats, while development individuals were blamed for hiccups in production.

One fascinating fact that emerged was that this particular interviewee stated that s/he had not seen any benefits emerging from the movement of the development team to Agile, since the code was still not in production. As s/he stated: “ We did not see any improvement in Agility with Scrum, because we missed the last phase which was release, and the code is useful only when it is in production and in the hands of the customer”. Based on this statement, the implementation of Scrum had been, perhaps, technically successful, but there were no benefits for the company and for IT in general. People in development were speeding up processes, but because of the delays in the release process, the new functionality did not proceed into production more quickly, that is, it had not yet reached the stage where it could be used by customers.

### 5.1.3 The transition to DevOps

As a result of the interviews, the following timeline for the transition to DevOps for this bank emerged:

- 2013: The bank began Agile development.
- 2014: A pilot project involving DevOps was launched.
- 2015: DevOps implementation began.
- 2016: DevOps was rolled out to all teams.

At the time of the interviews, the bank had 30 DevOps teams and it was in the process of rolling out more teams. One major challenge it was confronted with was the fact that the bank was still outsourcing a lot of its operational IT, that is, a lot of its operators were not employees of the bank, but were employees of third parties who lacked the necessary knowledge of the infrastructure. In addition, the DevOps teams often chose to move to the cloud, a change which meant that the third parties concerned lost power and control, something which led to difficulties. So issues involving third-party involvement sometimes introduced elements of uncertainty.

#### 5.1.4 The current situation

According to the interviewees, the software delivery lifecycle did not change the terms of the specifics and the detail involved; the software was delivered in shorter cycles, but the following aspects were still present and needed to be addressed:

- Planning: When are you going to work on certain business requirements and what is the effort needed?
- Writing codes: Create the code that will deliver the functionality, which will then solve the business problem.
- Checking: Test the code delivered.
- Release: Make sure that the code is ready for deployment and plan the deployment.
- Deploy: Ascertain that the new functionality is installed in production.
- Operate: Confirm the code works in production and support customers in the usage of the new and existing functionality.

The bank at the time of the research was working with outsourcers, which meant that a significant amount of work was not carried out by the employees of the bank, but was being done by outsourcers. This work had already been contracted for before the transformation to DevOps was undertaken. The contracts had been created at a time when operations and development were separate departments. A change in structure and a change in culture that are implemented during a time when an existing contract is being fulfilled requires flexibility on the part of the outsourcers.

In addition, the bank had invested considerable sums of money in automation, and so it did not have a continuous delivery pipeline. Automation has an effect on the effort required for a successful release, a fact which has a major effect on the resources needed from the outsourcer, thus impacting the revenue of the outsourcer.

Although the structures in the bank, particularly in terms of management, had changed, with teams being combined, the teams were still reporting to different managers. So, developers were still being supervised by a development manager, while operators were reporting to an operations manager, an arrangement that did not fully enhance the implementation of DevOps. The bank indicated that it wanted to change this structure, but that that had not yet happened.

Another challenge the bank was facing is the fact that the bank had inherited an IT legacy, that is, old methods, technology, applications, systems, etc. that were in use before DevOps was considered; these

were all highly interconnected, and so efforts to move to small teams with end-to-end responsibility was difficult and was not being currently implemented by the bank.

The aim of the bank was to have one team with developers and operators, but due to the fact that operations were often outsourced, that was not always the case. But teams do exist that are co-located and that do share knowledge on a face-to-face basis. Management is trying to bring in collaboration between operators and developers, which will be easiest if people share a location; however, when this is not possible due to outsourcers, sharing knowledge is facilitated via tools like Jira and ServiceNow.

#### 5.1.5 Internal measurement

A common theme amongst all of the interviewees was that the bank should carry out measurements/assessments of whether DevOps had been a success, but currently, no mechanisms to do this had been put into place. Assessment about deployment frequency differed heavily between the different interviewees, and thus their estimates varied markedly. Three stated that deployment frequency took place every two weeks, while others stated that deployment took place on a regular basis.

Although the quality of the code is checked as information passes through the automated deployment pipeline, no key performance indicators (KPIs) exist to assess the quality of the software. One interviewee stated that DevOps implementation had had a positive impact, but that there were no metrics and no specifics to support this assertion. One interviewee stated the following: “We do see fewer critical failures in production, and we do see fewer blockers when we do our automated tests. We currently have new applications deployed completely defect-free”. But right after this statement, he stated that he did not have any data to back this up. Based on this, the researcher concluded that the quality of the software was improving, but that there was no data to support this assumption.

One interviewee stated that the bank did measure employee satisfaction, and feedback indicated that that had improved. Another interviewee, in answer to a question about the impact DevOps had had on turnover: “That is hard to tell, also because we do have outsourcers working for us. So we do see a turnover, but not usually among our own personnel”.

All of the interviewees stated that they had expected that the waiting time in relation to releases would decrease, but there were no statistics to back up this assertion. In addition, because not all teams were working according to DevOps practices nor were all of them using the automated deployment pipeline, it was difficult to measure the success of the changes that had been implemented, because the optimization was only partial. Some of the team were working according to DevOps, while other teams were still working in the traditional manner. Frequently a DevOps team needed to work in conjunction with a traditional team, which resulted in less frequent releases.

According to the interviewees, most of them were aware of the NPS, but did not understand the metrics behind it and nor did they know whether DevOps had had a positive effect on the functioning of the bank.

#### 5.1.6 External measurement

This bank had measured NPS scores and revealed the following in its annual reports:

2015	-23
2016	-15
2017	-9
2018	-9

*Table 13: Company A: NPS scores*

As previously stated, a higher NPS score is claimed to deliver more revenue. What can be stated was that the NPS score increased when DevOps was implemented in 2015, and that there was a rollout to other teams.

The mobile application index (banken.nl 2018) indicates that a mobile application should have 90 functionalities; the bank delivered 57 of those in 2014, while in 2018 they delivered 70 of those 90 functionalities. The bank was number 1 in the rankings in 2014, but was also in first place in 2018, even though it had delivered fewer functionalities during that period than other banks. This could be an indication that code development and the release process were less effective than what was taking place in other banks.

The following data emerged from an analysis of the trust index of the bank in terms of their mobile services:

2015	4.2
2016	4.2
2017	4.4
2018	4.2

*Table 14: Company A: Banking monitor trust index*

This rating was stable in this period and comparable to that of other banks. The index does not prove that DevOps resulted in a significant improvement.

An analysis related to the Apple App yielded the following ratings:

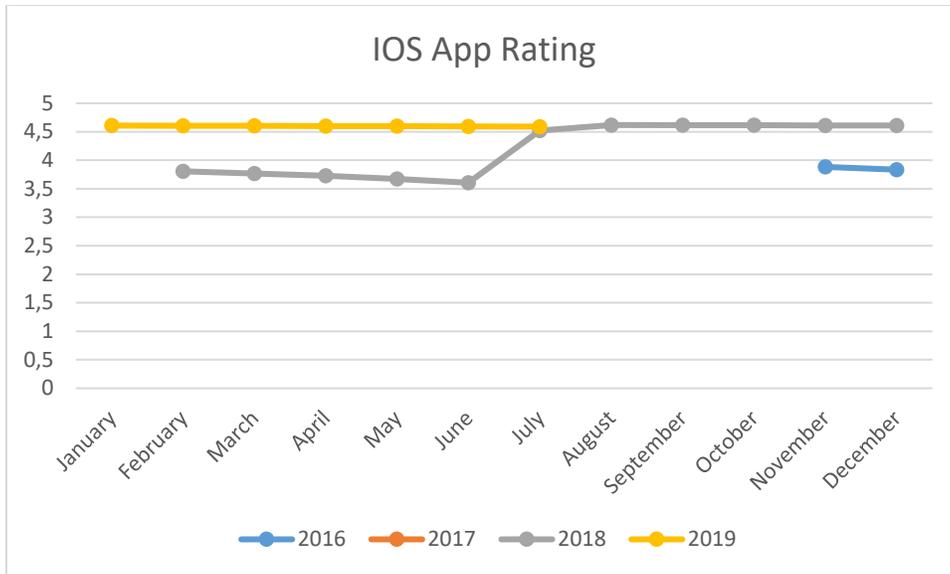


Figure 7: Company: IOS app rating

An examination of the information on the Google Play S indicated the following:

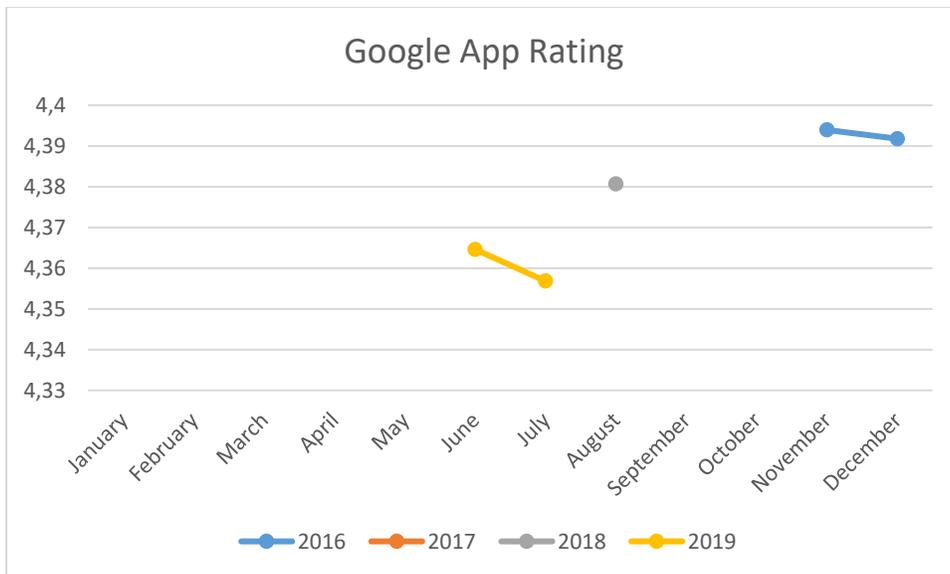


Figure 8: Company A: Google Play Store app rating

Unfortunately, the researcher could not supply any data on the app ratings in 2017, which was the first year in which DevOps was used. But there was an increase in appreciation of the application starting in 2018. But based on this data, there was no correlation between the app rating during the transition to DevOps.

### 5.1.7 Conclusions: Company A

The bank has a terse definition of DevOps: “You think it, you build it, you run it”. However, the bank had had difficulties with implementing DevOps, due to the fact that it had outsourced a significant number of their IT operations. This meant that the outsourcer’s personnel were managed and steered differently than the bank’s personnel. The contracts that were in place for the outsourcers were based on previous ways of operations; in addition, the sharing of knowledge between outsourcers might have had an effect on the competitive advantage of the outsourcer. This meant that transitioning to DevOps might have been difficult to accomplish.

From a measurement perspective, it is striking that the only measurement that the interviewees were able to back up with numbers was the release frequency. Other data was internally not available. This would seem to indicate that there was a focus on increasing the release frequency, a task that was accomplished.

Based on the external data, there was a rise in the NPS score, and of course, this could be related to the improved functioning of the IT in the organization; however, the other data provided, which has a closer link to IT-related services, does not support this assumption. So, in my opinion, the rise in the NPS score was not related to the implementation of DevOps.

## 5.2 Company B

Company B is a large retail bank in the Netherlands, but also has major operations worldwide. They have a 13,897 FTE in the Netherlands. Three people carrying out different roles from this bank were interviewed:

- One interviewee was at the time a chapter lead (line manager), who was responsible for implementing DevOps worldwide. A chapter lead is a term that comes from the scaling Agile model of Spotify. A chapter is a group of people who have similar competencies and are working in the same field of expertise<sup>19</sup>. A chapter lead is a line manager for members of the chapter, so he is responsible for personal development, salaries, and so on. But more importantly, he is squad (team) member, so still works in operations in order to ensure that s/he keeps in touch with what is going on on the ground. He was part of the team that introduced DevOps at the retail bank in the Netherlands in 2008.
- Another interviewee at the time of the interview was not only a Chapter Lead, but was also responsible (the product owner) for continuous delivery. As he stated, he had also been involved in the implementation of Agile in 2009.
- Another interviewee was a deployment engineer. He was responsible for maintaining the automation pipeline that was used in the DevOps way of working. He started his work on the deployment pipeline in 2012.

---

<sup>19</sup> <https://creativeheldstab.com/wp-content/uploads/2014/09/scaling-agile-spotify-11.pdf>

The two latter interviewees at one point had worked on the same team. Although the first interviewee had worked on the same team as the other, at the time of the interview he was working as a Chapter Lead for a different team.

### 5.2.1 What was this company's internal definition of DevOps?

All three interviewees gave different answers to the questions, but one common theme emerged—that all the tasks should be adding value for the customers of the bank.

One interviewee gave a wonderful definition of DevOps: “You think it; you build it; you run it; you monitor it; and you retire it”. The statement that “you monitor it”, has a direct link with the work of Lwakatare and others (2016), who have described monitoring as one of the key practices of DevOps. The statement that “you retire it” is an extension not only to DevOps, but also to the software lifecycle as described in the main concepts important to DevOps implementation. The other interviewee mentioned that DevOps had been implemented by a small team, but did not express himself in the colourful way that the previous interviewee had.

The third interviewee made a link to the practitioner literature because he spoke about the fact that his work had been greatly impacted by the ideas of Jez Humble, the author of a number of books about DevOps and about continuous delivery. The interviewee particularly mentioned the importance of the ideas listed below, which Jez Humble and Joanne Molesky (2011) articulated in their article: “Why Enterprises Must Adopt Devops to Enable Continuous Delivery”, published in the *Cutter Business Technology Journal* in 2011<sup>20</sup>:

- Culture, development, and operations should work as a team.
- Continuous automation should be applied to the release process.
- The company should function in a Lean manner, focusing on eliminating waste (which is the main focus of continuous delivery).
- Mechanisms for measurement were crucial (and were not currently being carried out).
- Sharing was critical, and should be the focus of the tribes.

In the Spotify scaling model, a tribe is a collection of squads (teams) that are working in related areas<sup>21</sup>.

Another important aspect described in the interviews was the important role that people and their skills played; one interviewee stated that they needed “T-shaped engineers”. He explained T-shaped in the following manner: “An important part of DevOps is that people can stand in for each other, so we need T-shaped people. We need a developer with operational knowledge and an operator with coding skills”.

---

<sup>20</sup> <https://www.cutter.com/article/why-enterprises-must-adopt-devops-enable-continuous-delivery-416516>

<sup>21</sup> <https://creativeheldstab.com/wp-content/uploads/2014/09/scaling-agile-spotify-11.pdf>

### 5.2.2 The situation before DevOps

All three interviewees had no insight into the components of the software delivery lifecycle; however, two of the interviewees mentioned the following steps as being crucial to software delivery:

- Plan
- Design
- Code
- Test
- Release
- Operate

They all spoke about the fact that there was tension between the operations and the development sectors; this was due to the fact that operations was responsible for stability, while development was responsible for change and for implementing new functionalities. As one interviewee stated: “There was friction between development and operations. Development was feature-driven and therefore driven towards change, while operations were focused on stability, and therefore reluctant to change”. In addition, what complicated matters was that there were a lot of handovers during a project. One mentioned that a number of projects were late and were over budget. The sharing of knowledge was done on an individual, and not on a group basis, a fact that limited knowledge transfer between development and operations.

Each department operated in silos and so had their own KPIs and their own tooling to support their processes. Another limiting factor was that many actions in relation to release management were carried out on a manual and not an automated basis.

### 5.2.3 The transition to DevOps

The transition that was identified with DevOps (in 2012) started with a change in management. One of the interviewees stressed the importance of the following timeline:

- the introduction of Agile and Scrum in 2009;
- the introduction of continuous delivery in 2011; and
- the introduction of DevOps in 2012.

The above is the process and timeline that this domestic bank in the Netherlands applied in terms of DevOps, and it is currently still implementing DevOps, together with changes to its banking culture and expansions of its tooling, throughout the rest of the world.

Another interviewee stated that the bank had already been using DevOps in their core banking applications in 2008. He stated: “I have had experience with DevOps since 2008 when we had an issue with our core banking issue. We had to create collaboration between developers and operators. They needed to collaborate to eliminate repetitive tasks; the developer can fix this, but the operator knows which ones are repetitive”.

But based on the internal definition of DevOps that had been given and that was being applied—"You think it; you build it; you run it; you retire it"—it can be stated that what they implemented was not DevOps at all. On top of that, other interviewees also stated that they had implemented DevOps in 2012. Because of statements made about DevOps during the interviews, the researcher utilized the timeline that had been described by the interviewees for his analysis of the external data.

#### 5.2.4 The current situation

The interviews indicated that there was a very good relationship between the developers and the operators inside the team. One interviewee stated the following: "There is for us no difference between development and operations. They are part of the same team". They are working side by side, so they are co-located.

The current software delivery lifecycle consisted of the following:

- An idea is generated and verified by the business to see if this will add value to the business.
- The idea is placed on the backlog of the product and will be owned by the product owner.
- The product owner refines the idea after s/he has understood its functionality, the business problem it solves, and the value it delivers.
- Sprint: This involves building, testing, and deploying the software.

The team does not accept any work that is not part of the sprint.

The interviewees indicated that the bank currently faced two major issues, which were articulated by the questions they asked:

- "How do we slice an idea?" "How can the tasks related to realizing an idea be divided up so that results can be delivered within a two-week sprint and that it delivers value?" One interviewee stated it as follows: "How do we slice the features? They have to be meaningful for the customer, but small enough to be implemented inside a sprint".
- Communication between teams was difficult, although the communication inside the team was good. One interviewee described it like this: "The main challenge we have today is re-factoring of the current pipeline. And within the teams, everything is running fine. But the communication between teams is a challenge. So DevOps at scale remains an issue".

In order to facilitate the communication between teams (squads), the bank organized the squads into tribes, that is, into a collection of squads working on the same business area. In addition, quarterly business meetings were scheduled in order to facilitate planning for a three-month period.

#### 5.2.5 Internal measurement/assessment

Since measuring/assessing the effectiveness of DevOps is an important part of its implementation, information (specific information) about how well it was working needed to be gained. One interviewee stated that the bank measured the following items:

- “the lead time; how long it took to get an idea into production”;
- “the mean time to repair (MTTR)”, which is the time it takes to restore service in case of a failure”; and
- “team happiness”.

In addition to the above-mentioned measurement, another interviewee stated: “We already check code quality and security in the continuous delivery pipeline using tooling”. They also measure code quality and code security that have been automated in the continuous delivery pipeline.

One of the measurements they undertook was connected to the number of deployments; this varied from team to team and could vary from 100 times per day in one team, and once per day or once every two weeks for other teams. The difference was in the technology that the teams used and supported. One interviewee was able to deliver the required number of deployments: “We deploy more frequently. We deploy 6000 per month, and we deploy 600 times per month into production”.

In addition, the bank also measured the number of incidents that arose and the availability of the applications. Based on all of the above, the interviewee's assessment indicated that DevOps had improved the quality of the services provided.

#### 5.2.6 External measurement

All of the interviewees stated that the bank measured Net Promoter Scores (NPS) and that these were mentioned in their annual reports. The individual score for this bank in the Netherlands was not available because that information could not be revealed.

The second external measurement was the ranking of the mobile applications and the number of features delivered: In 2014 the bank delivered 39 features out of 90 expected and in 2018 it delivered 55. In both years, this bank delivered the least number of features in this index, which is an odd statistic because, based on their timeline, they had already implemented DevOps in 2011. Based on the definitions of DevOps, one would expect that the company would more quickly deliver new functionalities in relation to their mobile apps.

Although the internal measurements show success for the implementation of DevOps, the index figures below do not back up these assumptions.

An examination of the trust index of the bank in terms of their mobile services yielded the following data:

2015	4.1
2016	4.2
2017	4.3
2018	4.3

Table 15: Company;, Banking Monitor Trust Index

Compared to the other banks in this research, the trust figures were in the same range, so there was no indication that the early adoption of DevOps gave a competitive advantage.

The following information on the bank's rating in the Apple Appstore was as follows:

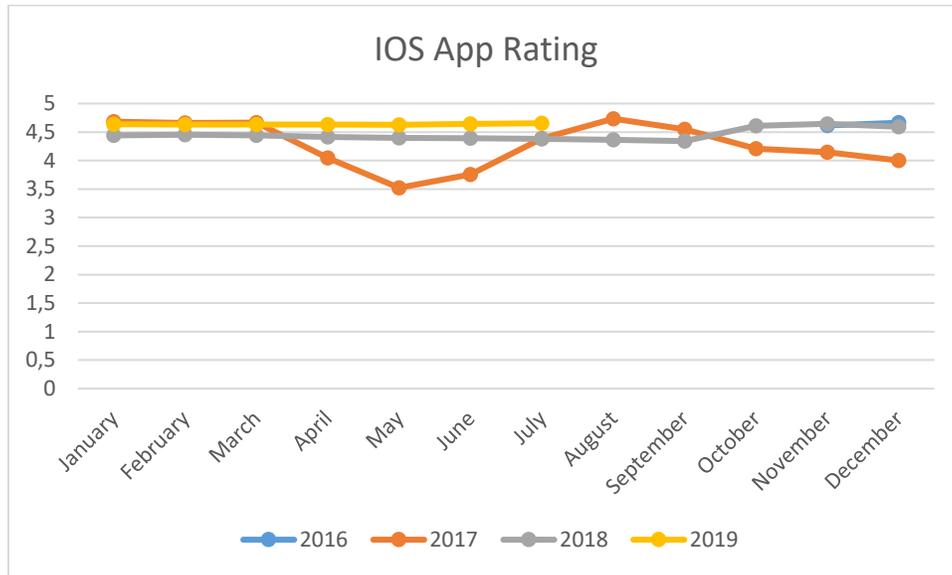


Figure 9: Company B: IOS App rating

And the following information was found in the Google Store:

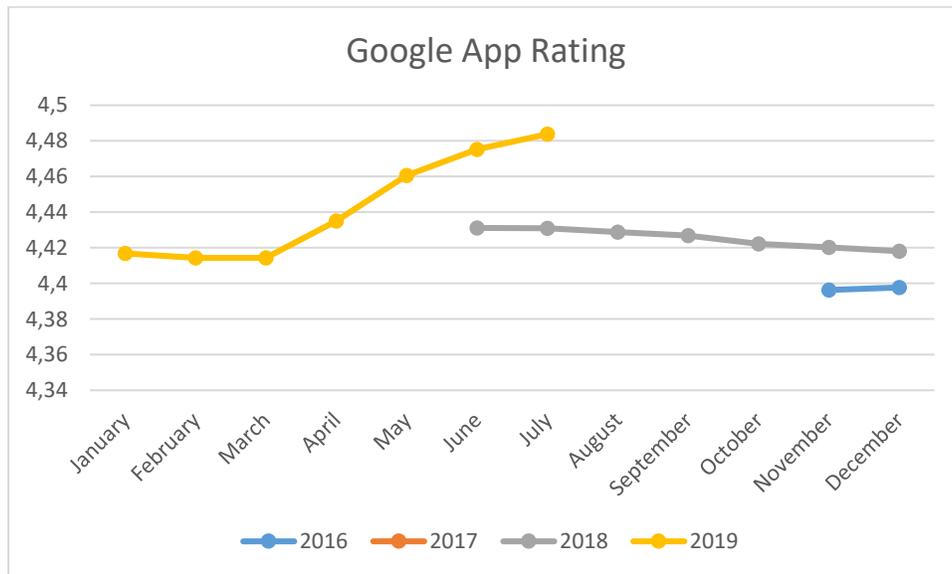


Figure 10: Company B: Google Play Store app rating

Due to the fact that the transition to DevOps had already been taken place in 2012, one would expect a very stable app rating of the mobile application. However, when the data examined, it can be seen that

2017 was a rough year for the mobile applications of this bank. This is another indication that DevOps implementation did not increase the company's success.

### 5.2.7 Conclusions: Company B

The bank has a wonderful definition of DevOps: "You think it; you build it; you run it; you monitor it; and you retire it". And in their definitions of DevOps, both Chapter Leads revealed a good understanding of the practitioner literature.

And in comparison with company A, their transition looked easier, due to the fact that the bank had not outsourced their IT operations. On the other hand, they described issues that arose because the bank was running DevOps at scale. Given the information that had been supplied about the transformation, one would have expected that the implementation of DevOps would have been at a more mature stage than that of the previous bank.

From an internal measurement perspective, the statements of the interviewees were interesting, but the statements were not backed up by data. The only data that the bank directly supplied was on the total number of deployments; in addition they provided statistics on the number of deployments that had gone into production.

From an external measurement perspective, one would expect a better rating in terms of the mobile index, a better trust index rating, and a better app rating, but based the results of this case study, one could question whether DevOps delivered success.

## 5.3 Company C

Company C was also a large retail bank in the Netherlands with a 41,861 FTE in The Netherlands. At the time that the research was carried out, this bank had a small footprint outside of the Netherlands. Two people carrying out different roles in this bank were interviewed:

- One interviewee was at that time a product owner and an IT manager. He was part of the transition to DevOps in 2015.
- The other interviewee was an architect for IT4IT. IT4IT is a reference architecture provided by the Open Group, which supplies prescriptive holistic guidance for the implementation of IT management capabilities for today's digital enterprises<sup>22</sup>. He stated to be familiar with DevOps since the adoption in a larger community, so he stated to follow DevOps since 2010.

Both of the interviewees worked in different teams.

---

<sup>22</sup> <https://www.opengroup.org/it4it>

### 5.3.1 What was this company's internal definition of DevOps?

In DevOps, a team is a combination of people responsible for change, and so those people responsible for continuity work together and share the same responsibilities. One of the interviewees stated the following: "In my opinion, DevOps is the next phase of Agile". On top of this statement, he described what the important aspects of DevOps were: "The most important aspect of DevOps is the cultural aspect, and the effect this has on the members of the team. The members need to become T-shaped individuals, i.e. a tester needs to also understand development and operations, while a developer needs to understand testing and operations and a system administrator needs to also understand coding and testing". It was clear from the Interviewee's statement the this company's definition of DevOps, one based on integration of the development and the delivery of software in accordance with Agile principles, facilitated its implementation.

### 5.3.2 The situation before DevOps

Before DevOps was implemented, the relationships between the operations sector and the development sector were tense; little to no information was shared.

The bank worked according to the following software delivery lifecycle:

- Design state: This involved the creation of a functional design, by its asking, "What are the business capabilities that this programme is going to address?" This also involved the putting into place of a technical design by its asking, "How will the capabilities be implemented technically?"
- Building and testing stage: This meant assessing the capabilities required by the technical design.
- Implementation: This involved deploying the technical capabilities into production.
- Operation: This meant that the company maintained the technical capabilities so that the customer could use the delivered functionalities.

A considerable number of handovers took place while work was being carried out in accordance with this lifecycle. However, this setup could be considered ideal for a bank, due to the fact that working according to this framework led to a segregation of duties, something which was well received by the bank. However, working according to this framework meant that no one took responsibility for tasks/solutions from start to finish.

### 5.3.3 The transition to DevOps

Although both the interviewees mentioned the same year as the beginning of the transition to DevOps, the actions and the implementation of the different phases differed. The first interviewee reported the following:

- Agile development began in 2010.

- In 2015, systems administrators were added to the development teams; however, one administrator was involved in multiple teams.
- Each team had at least one system administrator in 2016, so the teams were also supported during operations through their own system administrator.

The other interviewee summarized the steps that had been undertaken as follows:

- In 2010, a form of DevOps was introduced, one based on tooling architecture and stemming from policies and compliance discussions that had taken place.
- In 2014, Lean-Agile was introduced as the bank focused on eliminating waste.
- In 2015, DevOps began to be fully implemented, after a discussion had taken place as to how the suggested model could contribute to achieving a compliance-driven organization and after a reorganization had resulted in the formation of autonomous teams.

#### 5.3.4 The current situation

Due to the implementation of Agile, the bank works in short cycles, which the bank calls sprints (in terms of releasing software to customers). Since those sprints are short, the teams do not accept any work outside the sprint. So new functionalities will only be accepted in a new short cycle. The current software steps in the delivery cycle are:

- Refinement: The product has the required capabilities and reflects the business value of those capabilities; this is explained to the team.
- Sprint : This aspect involves the following steps:
  - Plan: agree as what is going to be delivered in this release cycle.
  - Code: Create the code that will deliver the business capabilities.
  - Test the code that is delivered against the required business capabilities and the other requirements.
- Release the code into production.

The bank at that time had one team that was responsible for introducing changes and for supporting the products that they offered. That team consisted of developers and operators, but it was clear that the bank thought that those members should become “T-shaped”, that is, that they should have deep, vertical skills in a specialized area but also have broad skills in other areas that enable them to become flexible and to become better team players. In the context of this bank, this meant that operators should be able to do basic programming and testing, while developers should be able to perform basic operational actions. Or as one interviewee stated: “We do see more and more that people inside the team are more and more able to perform each other’s tasks”. Because the teams were co-located, they shared information and ideas on a constant basis, which is a sign of collaboration.

The organization is based on tribes and guilds that share knowledge. The bank is based on the Spotify model for scaling agile, similar to company B. For example, during the weekly standup, one person might raise an issue related to his knowledge base or an ambition for a project that he might want to see addressed, and then the team took on the responsibility to educate that member.

### 5.3.5 Internal measurement

One interviewee stated that he measured the ratings for the performance of mobile applications and had shared that with the teams as guidance for their performance. He was able to recall the following app ratings, which were based on the Apple App Store rating, so on a scale from 1 to 5:

2015	4
2016	2
2017	4

*Table 16: Company C: Internal measurement of mobile app ratings*

He explained that the drop in ratings in 2016 was due to the fact that the bank had migrated to a new platform for their mobile development.

The bank did not measure the turnaround time for an idea, but one interviewee stated that this could vary from one day to several months. They deployed into production at the end of every sprint, which was every two weeks.

At the time of the interview, the respondent stated that the bank did not measure quality at that time, but that they did measure the app rating; the interviewee also stated a proper app rating could not be obtained a quality application. In addition, the bank calculated the number of incidents and whether the service in question was available immediately after a change.

The bank at the time of the research did not measure the rate of turnover, but both the interviewees stated that they had witnessed a stream of contractors leaving the bank for new assignments. One interviewee summarized this as follows: “We do not measure this [the turnover rate], but we also think that we are currently losing people not because we are doing DevOps, but due to the effect that we are not using the latest and greatest anymore. So, in the beginning, we had the most up-to-date environment and that attracted a certain group of developers. They are currently leaving, to pursue new opportunities”.

### 5.3.6 External measurement

It was pointed out that this bank had measured its NPS score, which was revealed in its annual reports:

2015	33
2016	36
2017	53
2018	57

*Table 17: Company C: NPS Scores*

The bank had the highest NPS scores, and these improved in 2017, while in the other years the improvements were minor. But based on the timeline of the transition to DevOps, this improvement

took place in 2015, so the implementation of DevOps does not appear to be correlated with the increase of the score.

The second external measurement was based on mobile application index, which states that a mobile application should have 90 functionalities; the bank delivered 41 of those in 2014, while in 2018 they delivered 67 of those 90 functionalities. This was a growth of 26 functionalities, which is the largest increase in functionalities that is reflected in this research. The bank transitioned to DevOps within the timeframe being measured, which could indicate that DevOps had been successfully implemented.

The trust index of the bank in terms of their mobile services yielded the following data:

2015	4.3
2016	4.2
2017	4.3
2018	4.3

Table 18: Company C: Banking Index Trust Monitor

This data is in the same range and there is not a huge increase during the transition, nor is there a decrease.

The Apple store yielded the following ratings:

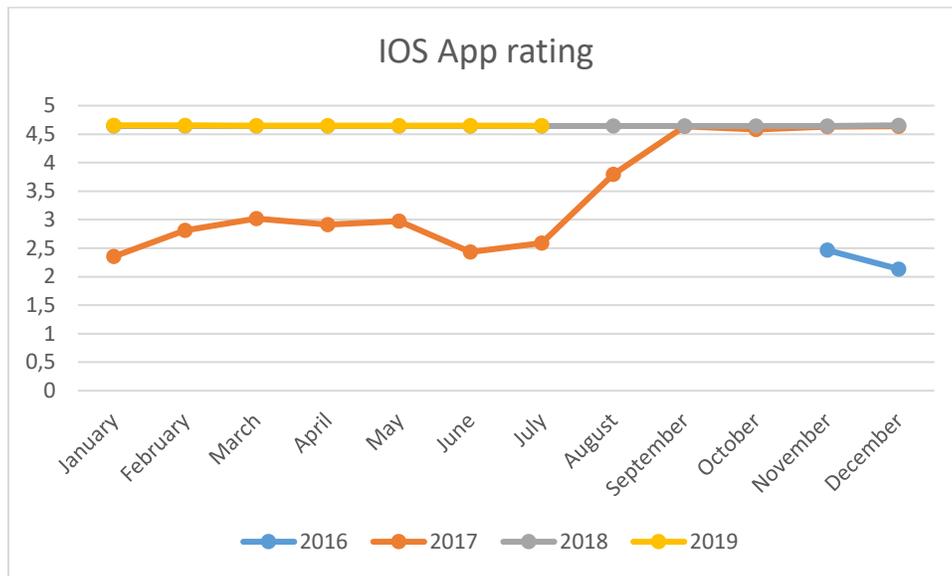


Figure 11: Company C, IOS App Rating

The internal data on the app ratings indicated issues in 2016, while the IOS app rating showed that there was also a low app rating until September 2017. The measurements indicate inconsistencies between the internal data and the external data.

The Google Play Store yielded the following information:

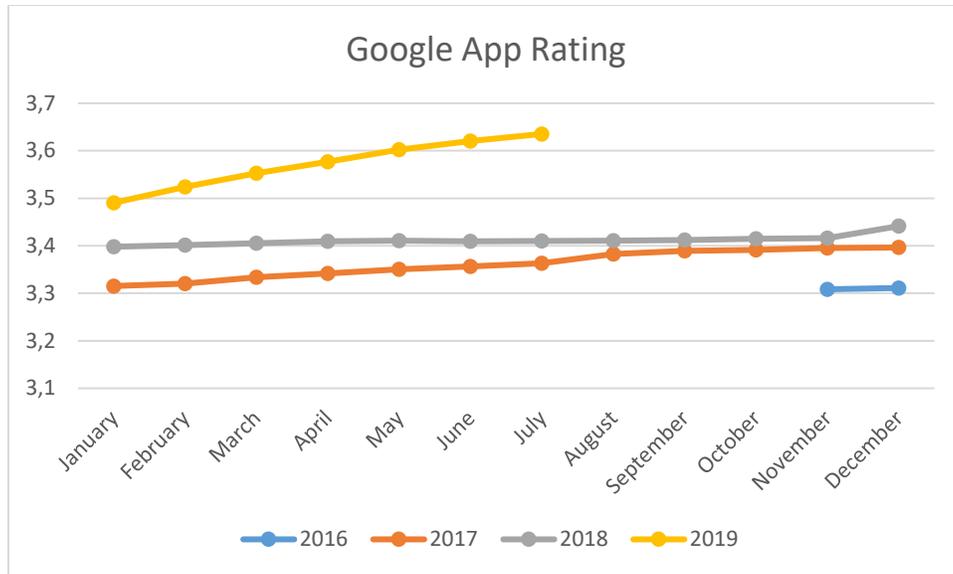


Figure 12: Company C: Google Play Store app rating

A correlation between the Google app rating and the transition to DevOps can not be demonstrated. In addition the customers’ appreciation for the Android mobile applications is consistent, but a correlation cannot be made between DevOps and increased customer satisfaction

### 5.3.7 Conclusion: Company C

The bank had defined its DevOps terms and it was clear that it focused on its employees. What was striking about the interviewees’ feedback was that they had differing perceptions of the transition; this made it difficult to draw conclusions.

A very positive point that emerged during the interviews was that those involved in the DevOps transition not only could describe what had taken place, but that they could also measure app ratings and could deliver actual data that proved their assertions. On the other hand, there were other internal measurements that were not backed up by data. However, the external data, apart from the IOS app ratings in 2016 and 2017, were positive for this bank.

A fascinating point that emerged during the interviews was that the segregation of duties causes difficulties when a company was transitioning to DevOps. This is an issue that must be further explored.

## 5.4 Company D

Company D is a bank with different brands in the Dutch market, and a 3,155 FTE in the Netherlands. The IT department delivers services to the brands, which still have their own IT departments, which customize the services delivered by the central IT department. To gather information in relation to external results, the researcher looked at the largest label this company was using in the market. Three different interviewees were questioned:

- One interviewee was at the time of the interview a manager in charge of Information and Communications Technologies (ICT), that is, he was responsible for internet banking and the automated teller machines (ATMs). The interviewee claimed that he had only limited experience with DevOps (in fact, less than a year).
- One interviewee was a project manager who was implementing test tooling for a DevOps team. According to this interviewee, the bank had started with the implementation of DevOps in 2015, and he was present from the beginning. He claimed to have had 3 years of experience with DevOps.
- One interviewee was a department manager in charge of three DevOps teams. The manager had recently been appointed as a manager for three DevOps teams. He had had less than a year's experience with DevOps at the time of the interview.

In this company, the project manager and the manager of the DevOps teams worked in/managed the same teams. The other manager was at a different location and was responsible for other teams.

#### 5.4.1 What was this company's internal definition of DevOps?

The common theme in the bank in terms of DevOps was: "You build it, you run it". Because DevOps is a combination of development and operations, both these sectors should operate as a team and should support the entire chain from start to finish. One of the interviewees gave the following definition of DevOps: "My definition of DevOps is that a team is responsible for the full chain from start to finish. This means that the team understands the business purpose, so it will create the requirements, code the solution, test it, release it and maintain the solution in production."

During the interviews, people stated what they considered to be the important aspects of DevOps:

- A member of a DevOps team should be able to stand in for another team member, so every member of the team should have T-shaped skills.
- The team should also be able to operate autonomously, but at the same time, it should be transparent about its performance and the tasks every member of the team performs.
- It was clear that the team viewed automation as critical: One interviewee used automation to segregate duties. This approach conflicted with the approach of another interviewee, who stated that developers should have access to production in order to make changes.

#### 5.4.2 The situation before DevOps

According to one of the interviewees, the bank had a software delivery cycle which was comprised of the following steps:

- Define the architecture for the solution.

- Carry out a business analysis if the solution is feasible and if it will provide business value.
- Discuss the budget needed to build the solution.
- Refine the requirements, especially when knowledge about the issue and about a possible solution have been provided.
- Define the test scripts.
- Build or code the solution.
- Perform all the tests.
- Promote the solution into production.
- Maintain the solution into production.

Another interviewee made an interesting observation, stating that once the software had been released into production, the team had been dismantled. The major issue was that the bank had multiple projects running at the same time in an environment where there was a limited set of resources. What ended up happening was that conflicts arose over scarce resources, and those resources were supposed to support multiple projects. This resulted in a rise of overheads and a lack of adequate coordination for those projects. Then the bank faced two other issues because of this setup: One issue was that, because testing was a separate stage, feedback to developers was delayed; the other issue was that, because the operations sector was not involved in the projects, sometimes the burden of maintenance was increased. Since operations personnel had not been involved, the needs that operators had were not addressed, so the solution that was proposed was limited in its serviceability. That resulted in additional efforts having to be made to maintain the solution in production.

Two of the interviewees describe the relationship between development and operations as tense. One even mentioned a situation involving a project in which the aim was to remove administrator rights in production from developers in order to comply with standards.

The sharing of information was limited and was present mainly in the process from development to operations; that information was limited because it focused on issues involving the functional and technical designs. Known errors, of course, were shared. A known error is an error for which the root cause and the workaround are documented:<sup>23</sup>

### 5.4.3 The transition to DevOps

All the interviewees stated that the bank was currently in the transition towards DevOps and that the following steps needed to take place:

- implementation of continuous delivery before 2017;
- building a vision for the bank in relation to DevOps in 2017;
- conducting experiments involving DevOps in 2018; and
- roll out DevOps to scale in 2019.

---

<sup>23</sup> <https://advisera.com/20000academy/blog/2014/02/04/known-errors-repetitio-est-mater-studiorum-case/>

One of the interviewees stated that the bank was not using a consulting firm to help them implement DevOps. But he thought that the following questions needed to be addressed:

- How should we transform ourselves in terms of the organizational culture? How can we help people to develop T-shaped skills and what is the role of Human Resource Management in this transformation?
- How should we re-engineer our processes, which are currently based on the Information Technology Infrastructure Library (ITIL)? Are release management and change management still needed? Is a Change Advisory Board still necessary?
- What technology solutions should we focus on in terms of continuous delivery, automated testing, or in relation to a software-defined data centre?

One other interviewee stated that they were looking at DevOps implementation in terms of following these steps:

1. Combine the people into one team:
  - Address the skills and enable people to acquire T-shaped skills.
2. Focus on the means:
  - Make sure the right processes are in place.
  - Make sure the right tools are available.
3. Connect the business:
  - Ensure that the business also uses Agile.
4. Ensure that the teams focus not only on change but on continuity as well.

#### 5.4.4 The current situation

Since the bank was transitioning to DevOps during the period of research, this analysis focused on examining and describing the teams that were currently already working according to the requirements of DevOps. Their software delivery lifecycle was made up of the following steps:

- Gather requirements, and then make a presentation of those ideas.
- Put items in the backlog: This means that the idea is accepted, but the team currently does not have the capacity to deliver the functionality.
- Execute sprints and deliver codes.
- Operate the solution.

In one of the interviews, one person stated that the bank needed to solve the issue of segregation of duties. Another challenge that was mentioned in relation to the new way of working was the fact that this bank was operating multiple brands. It was recommended that those brands should use the same platform to gain efficiency, which is currently not the case; this will necessitate further testing.

In addition, the organization is project-oriented, a fact that is reflected in the way financial planning is done: Someone needs to ask for a budget to run a project, an approach that conflicts with having stable and self-organizing teams.

The interviewees stated that the relationship between development and operations inside the teams was operating smoothly: They are co-located and are sharing information because they were sitting next

to each other. However, one of the interviewees stated that, for some reason, changes were more readily accepted by the operators than by the developers. The reason for this was that the operators were picking up knowledge from the developers and thus were growing towards becoming employees with T-shaped skills, while the developers remained stuck in old behaviours.

#### 5.4.5 Internal measurements

Interviewees stated that the bank measured both the velocity of the teams and the story points produced by a team. In that way, the bank checked the performance of the teams. In addition, it still carried out a function point analysis, a mechanism for assessing functionality.

The interviewees also reported that the bank had also seen an increase in the number of deployments. At the time of that this research was carried out, they were doing 20 deployments per year of their presentation layer, and they performed 8 deployments of their data access layer per year. In addition, the bank was measuring the number of incidents after a change, and they stated that they had seen an improvement.

One of the interviewees mentioned that the bank was looking at the ratings related to mobile applications, but could not provide the ratings; his statement was not corroborated by others. The bank was also tracking coverage, a measurement that was used as a standard for quality. At the time of the research, the bank was at 85% test coverage. The interviewees stated that the bank measured the NPS, but they were unable to provide any specifics. This bank was easily able to provide figures on the number of deployments, but other internal metrics were not given, apart from the test coverage.

#### 5.4.6 External measurements

This bank measured the NPS scores, which were mentioned in its annual reports:

2015	Unknown
2016	-8
2017	-3
2018	-0

*Table 19: Company D: NPS scores*

Although this company was at the start of their DevOps transition, their NPS was stable and was slowly increasing. Their score was mid-way of those for the other two banks that were part of this research and that had supplied their NPS scores. The fact that they had not yet succeeded in transitioning to DevOps was not reflected as a major disadvantage.

The mobile application index (banken.nl) stated that a mobile application should have 90 functionalities; this bank delivered 40 of those in 2014, while in 2018 they delivered 58 of those 90 functionalities. Compared with the others, this bank delivered an additional 18 features and this was in the normal range of this index. So, according to this metric, this bank's not being fully transitioned to DevOps was not a disadvantage.

The trust index of the bank in relation to their mobile services yielded the following data:

2015	4.4
2016	4.4
2017	4.5
2018	4.5

Table 20: Company D: Banking Index Trust Monitor

This bank had the highest scores in the above index, again indicating that the slow transition to DevOps was having no effect in the relation the bank had with its customers.

Very little information was able to be obtained in terms of its app ratings. The only data that was available was for 2019.

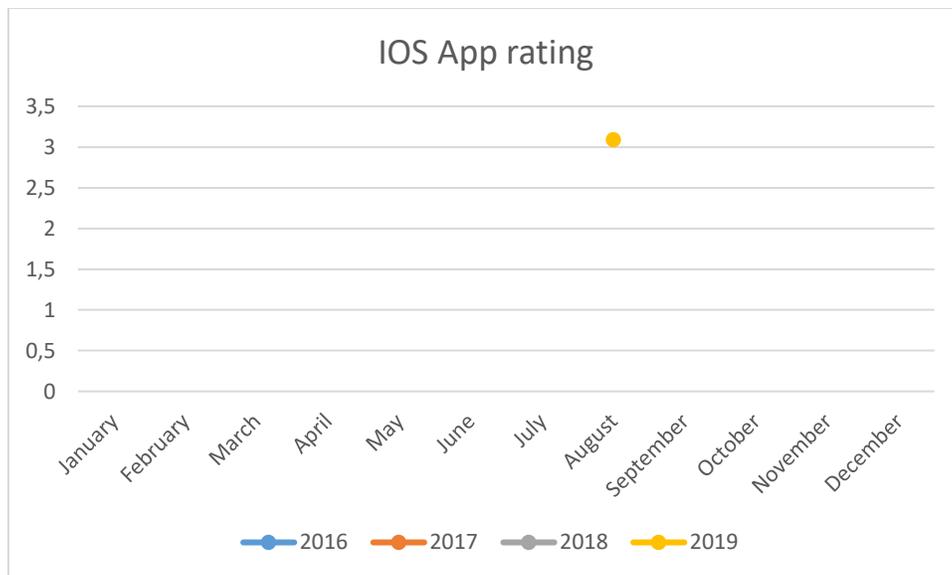


Figure 13: Company D: IOS app rating

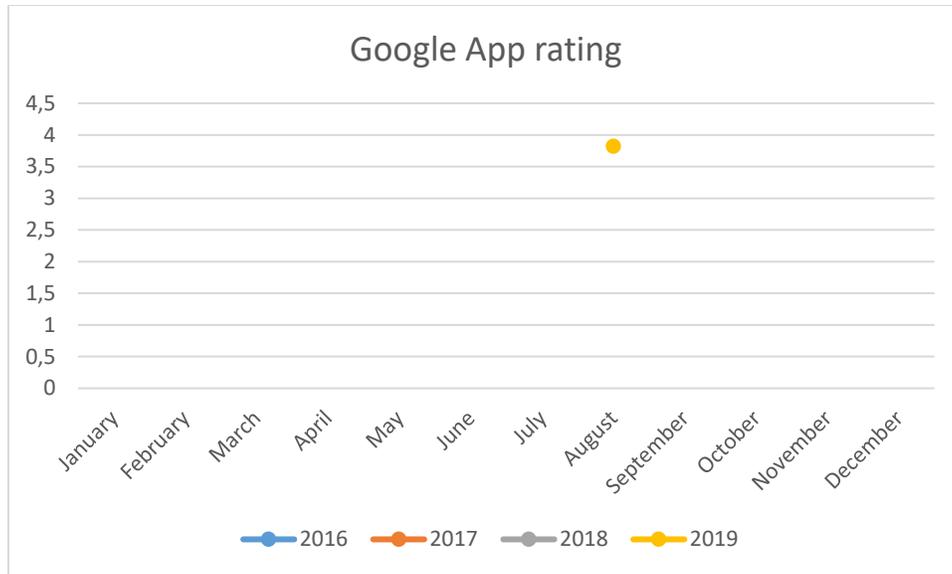


Figure 14: Company D, Google Play Store App Rating

The data referenced is very limited and out of the scope of this research, and so was not applicable to the discussion taking place here.

#### 5.4.7 Conclusion: Company D

This bank was on the verge of its transition to DevOps. It has a clear idea of DevOps and a strict focus on culture.

On the other hand, there was a complicating factor impacting this bank in its transition. The complicating factor was the fact that this bank consisted of multiple brands, so a part of its IT was being shared, meaning that the company could not be responsible for a solution from start to finish. Meaning that the company could not be truly customer-centric.

One would expect that an assessment of the external data would demonstrate a negative trend, due to the fact that this bank was slow in their adoption of DevOps, but the metrics chosen and provided do not show this.

## 6. Analysis

As stated at the beginning of this thesis, the principal research question was:

How do companies compare in terms of their success in implementing DevOps?

Answers to these questions were to be answered on the basis of supporting questions, which were subsequently investigated:

1. What is DevOps? Initially, the focus was on the scientific theory and the practitioner literature; however, in this chapter the research combines the definitions from people that were part of a transition to DevOps, thus adding a new dimension.
2. How can the possible maturity stages for DevOps inside an organization be defined? The analysis which follows uses two of the maturity models presented at the beginning of this research to compare the different organizations and make a detailed analysis of those maturity models.
3. What are the success indicators for DevOps inside an organization? It is important to ascertain, on the basis of the information gleaned from the interviews, whether there are KPIs that can show the success of DevOps.

### 6.1 Definitions for DevOps

As described in the literature section, there are varying and numerous definitions for DevOps. But one theme can be distilled as a result of the research: “You build it, you run it.” And of course, a lot of variations of this theme have emerged.

The interviews confirmed the descriptions of DevOps as given by Lwakatare and others (2016). Successful DevOps implementation involves the following:

- collaboration, rethinking, and re-orientation of roles and teams in terms of development and operations activities;
- automation, infrastructure, and deployment process automation;
- culture, empathy, support, and the presence of a good working environment between development and operations personnel;
- monitoring; the drive to instrument applications; and the necessity to aggregate monitored data into insights; and
- the creation of a system of measurements (useful metrics).

The presence of a productive and collaborative environment was mentioned as a necessity by all the companies consulted; this means that development and operations activities are combined and that the companies promote the development of personnel with T-shaped skills.

The need to enhance automation was a key point mentioned during all of the interviews: What was stressed was automated delivery through continuous delivery.

Developing a positive culture, that is, good relationships between development and operations personnel was key because this ensured that both sectors shared the same goals.

Monitoring was not mentioned during any of the interviews. The interviewees were asked during the interview process whether their companies had made significant changes in terms of their tooling infrastructure and, if so, which tools had become more prominent. Although all the companies mentioned that it was important to increase the number of tools that would support continuous delivery, none mentioned that the number of monitoring solutions had become more numerous. An important point to mention is that, during the interview process, no one had mentioned whether their companies had invested significantly in monitoring solutions.

In terms of measurement, most of the companies measured the number and gravity of incidents after a change and mentioned specifics related to the frequency of deployments. So, measurements were and are being carried out, but none of the companies were capable of proving/backing up their success with DevOps through the provision of metrics. Because none of the companies had started with a baseline before DevOps, no successes could be claimed because a basis for comparison did not exist. Company A did carry out a test to see if a simple programme could be delivered into production. The transition took more than six months, but this test was not carried out after the implementation of DevOps.

The research carried out and the interviews that were conducted confirmed the validity of the principles underpinning DevOps (as summarized by Rana and Staron (2016)). The results indicated that, when these principles were applied, DevOps genuinely impacted the productivity and success of the company concerned. The following observations could be made about the companies analyzed:

1. Culture, development, and operations worked together to deliver services to the business.
2. Automation was used to speed up delivery and feedback, and automation was used to build, test, and deploy by all the companies studied.
3. The carrying out of measurements was only partially fulfilled by the different companies; one interviewee actually believed that the mobile app ratings could be considered as a measure for customer satisfaction. The measurements mentioned by the other interviewees, that is, the number of incidents after a change and the frequency of deployments were technical measurements, but only provided limited technical information. What is interesting is that, when companies were asked if DevOps had actually improved their quality, all of the interviewees stated that they thought it had, but they could not deliver the statistics to back this up. In other words, perceptions could not be corroborated by hard, measurable data. Next to that, there were no measurements provided in relation to the optimization of processes; during the interviews, it was stated that the lead time decreased, but this was not measured.
4. The importance of sharing: All of the companies saw an improvement in the relationship between development and operations, a critical component of DevOps.

Based on this information, the description of DevOps that Soni (2015) gives has been verified by this research and by the interview that were carried out—that establishing a DevOps culture extends the Agile methodology in a manner that rapidly creates applications and delivers them across the environment in automated manner, thus improving performance and quality assurance.

## 6.2 DevOps maturity

In his value-oriented approach of DevOps, Mohamed (2016) describes the following stages of DevOps maturity:

1. Initial: This is the first stage, and during this stage, the delivery of services is unpredictable and the actions in the deployment are manual.
2. Managed: During this phase, DevOps will use timeboxing, but the predictability of the performance of the team is an issue. This results in constant reprioritization.
3. Defined: During this stage, there is a release cadence that will deliver an enhanced service on a regular basis.
4. Measured: This means that the team can release when the business demands it.
5. Optimized: The team not only delivers, but also measures and optimizes results on the basis of those measurements.

Mohamed (2016) describes the five stages as follows:

1. At the initial level of DevOps maturity, organizations are unpredictable as to when a service will be production ready, since the success of their efforts depends on “local heroes”. Those heroes are talented and scarce resources. At the initial level of maturity, there are many manual tasks that must be performed during the deployment of a new version of software service/application. Due to the manual tasks, it can take days or weeks to complete a full-service release cycle (inclusive of testing).
2. At the managed level of DevOps maturity, software development teams and business leaders work together to set release timeboxes. After this initial scoping of the work to be performed, development teams then vary the scope of the work to be performed and calculate in detail the effort needed during their sprint refinement so that they can meet the agreed release date. The business needs are usually not met by the delivery timeboxes at the managed level. As a result, a managed level of DevOps maturity is characterized by discussions between the business and the development teams. In order to facilitate this discussion properly, the presence of a product owner, who can assess the trade-offs in quality and functionality, is extremely important, due to the fact that the discussion between business and development will continue throughout the sprint.
3. At the defined level of DevOps maturity, a key indicator for the health of a project is the release process. Most organizations that operate at the defined will make sure that developers will check in their code at least once a day. Out of that code, a deployable product will be created. At this stage of maturity, the release cadence is regular. One of the early signs that a project is in trouble is the fact that a team misses the cadence.
4. At the measured level DevOps of maturity, a new release can be deployed whenever it is needed. This is a critical maturity stage. The service can be released when the business needs it. When a development team reaches this level of maturity, it will require that it operate in a cross-

functional manner. This cross-functional team will consist of developers, testers, operators, and business analysts.

5. At the optimized level of DevOps maturity, incremental innovation will be delivered continuously by software development teams. They will adopt a “scientific approach” to service development, which will entail the following steps:
  - They will form a hypothesis about customer needs.
  - Then they will create a plan to meet those needs.
  - Experiments will be conducted to test the hypotheses.
  - Finally, feedback will be used to ensure that an excellent service is provided.

The above detailed information about maturity models has enabled comparisons to be made in terms of the DevOps maturity of the different companies that have taken part in this research.

Company B, during the interviews, explained that they were transitioning to BizDevOps (business, development and operations in one team), and for their presentation layer, they released software whenever it was needed. For the presentations layer, company B was assessed at the measured level, as the previous statement shows. But during the same interview, the interviewee stated that, in relation to the data access layer, they were releasing software every two weeks. This means that the company can set a release cadence based on a timebox (of every two weeks), so this means that this organization has reached the defined level of maturity.

Assessing Company A in accordance with Mohamed’s categories, it is clear that some of its operations still take place in silos, due to the fact that much of their IT work is outsourced and thus is not co-located. As one of the interviewees stated: “We have outsourced our IT. We need to work not only cross boundaries between development and operations, but also across companies. That is challenging in the current contracts”. But they do have a release cadence and a timebox in accordance with which they deliver functionality, but based on the previous statement, there is a discussion about the priority of what to deliver in the timebox, so this company can be assessed as having reached the managed level of DevOps.

During the interviews, Company C stated that the teams released software at the end of every sprint, so the team did have a release cadence. They indicated that they had teams that combined operators, developers, and testers, so they had cross-functional teams. But the teams were not capable of deploying on-demand; this means that the company reached the defined level of maturity.

Company D has described that they are in the process of migrating to DevOps. They still maintain centres of excellence, and they are still competing over resources. The various interviewees differed in terms of what they considered to be the frequency of deployment, thus making it hard for the researcher to assess exactly what they were doing and how and when they were deploying. Responses indicated that there were variations between teams and between applications. Based on this information, this company can be assessed as only reaching the initial level of DevOps implementation.

Company A	Managed
Company B	Measured
Company C	Defined
Company D	Initial

Table 21: DevOps maturity, according to Mohamed

### 6.2.1 Micro Focus

Another maturity mode for DevOps, one which focused not just on the element of automation, was created by Micro Focus, (and described the following three additional elements that could allow an assessment of the success of the implementation to take place):

- People
- Process
- Technology

Maturity Model	1 Initial	2 Managed	3 Defined	4 Measured	5 Optimized
<b>People</b> 	Ad-hoc communication & coordination Silos still in place	Managed communication Shared decision making Ad-hoc collaboration Specialized people Some agile leaning	Shared accountability Dev and Ops teams inside Release Trains Some cross-functionality Lean-Agile disposition expressed	Collaboration as culture Integration roles between Dev & Ops standard Cross-functionality prevalent Lean-Agile mindset prevalent	Optimized for DevOps Shared Ops teams formal Knowledge sharing & individual empowerment High cross functionality Lean-Agile leaders with a collaborative mindset
<b>Process</b> 	Uncontrolled or reactive processes Predominantly waterfall	Unstandardized processes Dev & Ops Processes separately owned & managed Some agile adoption and iterative practices	Processes standardized Common KPIs Agile & Kanban in dev & ops	Visibility & predictability of entire process Continuous practices in a DevOps pipeline Lean-Agile approach scaled across the enterprise	Process risk & cost optimization Highly optimized processes Continuous assessment impacts optimization Agile/Lean drives process optimization
<b>Technology</b> 	Little or no automation	Siloed automation for Dev & Ops tools & practices No central infrastructure Some collaborative tools	Centrally automated & integrated Infrastructure as code exists Managed test environments Collaborative tools standardized	Automated metrics collected & analyzed against business goals Automated prod release Automated reuse & remediation Infrastructure as code e2e	Self-service automation for all infrastructure Analytics enables self & remediation DevOps pipelines are deployable & integrated across the organization

Figure 15: Micro Focus DevOps Maturity Model

Although the interviews showed that all of the companies that had been interviewed focused largely on automation, most of the interviewees also spoke about the need for a change in culture within the organization, one which would ensure DevOps success. The change that was needed focused on people, so the Micro Focus model was applied to ascertain whether this model would yield different results in

terms of maturity levels. Mohamed’s five stages were applied to three additional categories: people, process, and technology.

Company A stated that, in some instances, people worked within one team and shared the same goals; however, if outsourcing was required, the development and operations teams were not combined into one team. One of the interviewees stated that managers were still divided into two groups—operations managers and development managers, a fact that meant that people reported to different teams. What this implied was that there was no shared responsibility and that people were working to achieve different KPIs. Thus company A, in terms of the stages of DevOps, ranked as being in the “initial” phase of maturity in terms of its focus on people.

An examination of the process area indicated that the company did have controlled processes; they had adopted some Agile processes, but Dev and Ops processes were separately owned and managed. But there were not common KPIs for the team as a whole. So, in relation to the focus area, they could be ranked as operating at the “managed” level.

At the time that the research had been carried out, company A’s technology was centrally managed and integrated. They were working towards developing infrastructure as code, reflected by the fact that some teams were completely cloud-oriented. However, for the teams that were still using the legacy infrastructure, this was not yet the case. In addition, test environments were being managed and collaboration tools were being shared. Since the required metrics had not yet been collected, those metrics could not be measured against business performance. So, company A could be ranked as being at the “defined” stage in terms of technology.

	<b>People</b>	<b>Process</b>	<b>Technology</b>
Company A	Initial	Managed	Defined

Company B, during the interviews, spoke about the fact that they were focusing on changing the culture in which they operated. The people were working as one team and shared common responsibilities. One of the interviewees spoke about the fact that they needed to focus on eliminating waste, that is, on developing a Lean action mindset. Their speaking at some length about BizzDevOps revealed that they were thinking in a cross-departmental and cross-functional manner. A culture of collaboration had been developed. So, in terms of the people focus area, this company could be ranked as being in the “measured” category.

The interviews indicated that company B used Agile in their DevOps teams, so Agile principles and practices were applied to both development and operations. The company was striving to achieve a standardized way of working, something which was already taking place in the Netherlands and which it was rolling out on a global basis. According to one interviewee, the following KPIs were being measured:

- the lead time: how long it took to get an idea from its initial stages into production;
- the mean time it took for repairs to be carried out; and

- the levels of team happiness.

One other interviewee stated the following: “Measurement is for us still an issue”. What the individual meant is that, at the time, the company did not measure the effectiveness of processes and the effect of DevOps, but what s/he did state is that s/he had seen a decrease in the number of incidents. Both interviewees implied that there were shared KPIs for the team. Thus, based on this information that emerged from the interviews, in terms of the process level, company B could be ranked as “defined”.

Company B also had a centrally managed and integrated tooling environment. The interviews did not provide any information as to how the company deployed its infrastructure, and none of those questioned mentioned this point. Since sharing was done on an informal basis, they did not have standardized tools for carrying out this activity. However, since automation was centralized and integrated, the company could be ranked as “defined” in terms of technology.

	<b>People</b>	<b>Process</b>	<b>Technology</b>
Company B	Measured	Defined	Defined

During the interviews carried out with company C, the interviewees spoke about the fact they considered DevOps to be an extension of Agile and of the Agile mindset. One interviewee spoke about the fact that the company was striving to develop people with T-shaped skills. In other words, developers needed to broaden their knowledge and acquire system administrator capabilities. The interviewee also stated that they did not accept any tasks that had not been defined in a sprint (the team does not accept a change of scope, so they actually rejected work), and thus, within a release train, responsibilities were shared. These statements led to the conclusion that company C could be ranked as “defined” in terms of the people focus area.

In the focus area of process, the interviews showed that the teams did have KPIs in common. They have had only one detailed view on the mobile app rating, and that was used as input for the teams.

The company had added system administrators to development teams that were already working in accordance with Agile principles and practices. This meant that Agile practices were shared amongst all the development and operations personnel. Given the above information, company C would be ranked as “defined” in the area of process.

The interviewees from company C did speak about three different pipelines coming up with different solutions, a process that was not centrally managed. They also indicated that different solutions were used by different operators, meaning that there was no standardized solution for technology issues. They also spoke about different infrastructure solutions: Mainframe, Windows, and Unix. The feedback indicated that there was a limited central infrastructure. Because of this, this company could be ranked as “managed” in terms of the area of technology.

	<b>People</b>	<b>Process</b>	<b>Technology</b>
Company C	Defined	Defined	Managed

The respondents from company D stated that it had only three DevOps teams and that the organization was in a period of transition. At the time of the research, the interviewees stated that operations was currently working in a silo, but that, through an application of SCRUM, they had combined testing and development. One of the interviewees stated that the organization was still project-oriented, so communications tended to take place on an ad hoc basis and that a lot of coordination between different parties was still required. The information that was provided led to company D being assessed at the level of “initial” in terms of the focus area of people.

The teams were still separated and were separately managed, but the company was busy implementing Scrum, which qualifies as an Agile methodology; this indicates that there was some adoption of Agile and its iterative processes practices. The company used a WIKI to share information, so some collaborative efforts had been made. This information meant that company D ranked “managed” in terms of the focus area of process.

One of the interviewees stated that the company had considerable technical debt; the principal issue they were facing was that the organization wanted to standardize and their main issue was that they wanted to standardize their IT infrastructure and processes to using one platform, indicating that they were not using a single, standard infrastructure. In addition, they stated that they were currently implementing an automated continuous delivery pipeline. The answers that were given indicated that the company was making little or no use of automation, and therefore it could be ranked as “initial” in the technology focus area.

	<b>People</b>	<b>Process</b>	<b>Technology</b>
Company D	Initial	Managed	Initial

Combining both scores (Mohamed’s and Micro Focus’) resulted in the following table summarizing levels of maturity:

<b>Company</b>	<b>Mohamed</b>	<b>Micro Focus People</b>	<b>Micro Focus Process</b>	<b>Micro Focus Technology</b>
Company A	Managed	Initial	Managed	Defined
Company B	Measured	Measured	Defined	Defined
Company C	Defined	Defined	Defined	Managed
Company D	Initial	Initial	Managed	Initial

*Table 22: DevOps maturity comparison*

The results revealed in the table above would seem to indicate that, although Mohamed’s categories of maturity were useful in terms of measurement, Micro Focus’ more definitions/descriptions resulted in more nuanced interpretations of maturity levels.

**6.3 Success**

Riungu-Kalliosaari and others (2016) state that implemented features and frequent releases emerged as benefits as a result of implementing DevOps.

The data stemming from the interviews enables the following table to be created describing information related to frequency:

	<b>Releases Before</b>	<b>Releases After</b>
Company A	Less than every two weeks; it took 6 months to bring “Hello World” into production.	Every two weeks
Company B	Differed from project to project, but less than every two weeks	Every two weeks
Company C	Fewer releases than before	Every two weeks
Company D	Fewer releases; one stated that releases took place on a quarterly basis.	20 deployments per year for the Presentation layer 8 deployments per year for the data access layer

Table 23: Internal release frequency

Externally, the following figures indicating frequent deployment summarized performance for company A in terms of their mobile application on IOS:

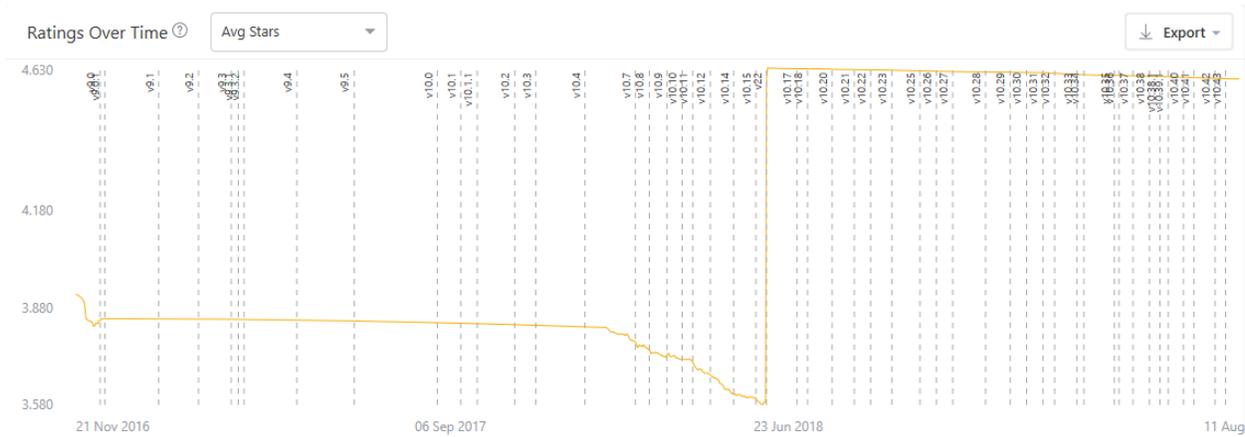


Figure 16: Company A, Apple App Store release frequency

Externally, the following figures indicating frequent deployment summarized performance levels for company B in terms of their mobile application on IOS:



	<b>Features 2014</b>	<b>Features 2018</b>	<b>Increase</b>	<b>Number</b>
Company A	57	70	23%	13
Company B	39	55	41%	16
Company C	41	67	63%	26
Company D	40	58	45%	18

Table 24: Banking Monitor Overview

On the basis of the information provided by this table, company D had placed second in terms of delivering the most features, while they stated in the interviews that they had released the fewest features.

Another interesting fact was that the interviewees from company B stated during the interviews that the organization had already introduced DevOps in 2012, leading to the assumption that they would have a more advantageous position in terms of features related to their applications in 2015, whereas they had the least. So, the conclusions that can be drawn on the basis of this table and on the previous information, companies using DevOps are delivering more frequently, but their applications are not richer in features.

Another advantage that Riungu-Kalliosaari and others (2016) mention in relation to DevOps implementation is enhanced collaboration and communication, a benefit that was reflected in the performance of the various companies.

	<b>Communication Between Development and Operations Before</b>	<b>Communication Between Development and Operations After</b>
Company A	A blame culture existed between development and operations. They worked in different teams that were not co-located.	The company is seeking to establish co-location. Sometimes knowledge is shared via JIRA.
Company B	The teams were measured on the basis of different KPIs. Sharing information was done on an individual basis.	The teams are co-located and communication within a team is good. However, there is a lack of communication between teams.
Company C	The relationship between development and operations is described as tense.	The teams are co-located and information is constantly shared.
Company D	The relationship between operations and development is described as tense. Operations describe new items of development as a maintenance burden.	The teams are constantly sharing information and are co-located.

Table 25: Collaboration and communication overview

A third benefit described by Riungu-Kalliosaari et al (2016 page 594) is the following: “Working with real customers, companies are better equipped with knowledge on customer preferences and are ultimately able to tailor their products to meet the market demands”. That information would lead one to conclude that a company that had implemented DevOps would take more advice from their customers and, based on that assumption, the NPS scores were checked for the different organizations, which yielded the following results:

Year	Company A	Company B	Company C	Company D
2013	Unknown	Unknown	12	Unknown
2014	unknown	Unknown	17	Unknown
2015	-23	Unknown	33	Unknown
2016	-15	Unknown	36	-9
2017	-9	Unknown	53	-3
2018	-9	Unknown	57	0

Table 26: NPS overview

The timelines supplied during the interviews did not yield a correlation between the NPS score and the implementation of DevOps. There was no negative effect, nor was there a positive effect, so a connection cannot be made between the introduction of DevOps and increases in NPS scores. A link also cannot be made between decreases in NPS scores and DevOps. Company C had the biggest increase in NPS scores, but this did not take place in the year DevOps had been implemented.

The Dutch banking industry, “De Nederlandse Vereniging van Banken”, released in 2015 the “vertrouwensmonitor” (confidence/trust monitor) (De Nederlandse Vereniging van Banken, 2015, 2016,2017, 2018). In this document, they state that trust is important for a bank. On the basis of this information, the researcher examined the banks’ digital presence in terms of trust, and compared that information with their DevOps transition, to see if DevOps implementation had impacted the trust levels of a bank’s clients:

Year	Company A	Company B	Company C	Company D
2015	4.2	4.1	4.3	4.4
2016	4.2	4.2	4.2	4.4
2017	4.4	4.3	4.3	4.5
2018	4.2	4.3	4.3	4.5

Table 27: Trust Monitor overview

The Trust Index Monitor did not corroborate a link between an implementation of DevOps and better scores in terms of trust. Company D, the company that was the last to implement DevOps, had the best score in this monitor.

Apart from the benefits of DevOps described and assessed above, Floris Erich, Chintan Amrit, and Maya Daneva in their article, “A Qualitative Study of DevOps Usage in Practice” (2017), describe reasons why other companies have implemented DevOps:

- One company described a shorter lead time for projects and a better quality of the software that was delivered.
- Another company described an increased effectiveness in terms of problem-solving.
- The third company reported a reduction in false alarms and a decrease in escalations of incidents.
- The fourth company had implemented DevOps in order to receive continuous feedback from stakeholders, but they stated that the effect was difficult to measure.

The companies claimed that a better quality of software had been achieved after the implementation of DevOps, but none of them had measured the impact.

	<b>Quality Statements</b>
Company A	The number of incidents in production is measured. In the deployment pipeline, the quality of the software is measured. The number of failed tests that stop automatic deployment has decreased.
Company B	The no. of incidents and levels of availability are measured. Quality has improved.
Company C	The no. of incidents in production and availability is measured.
Company D	The number of incidents is measured. DevOps has led to a decrease in these incidents. Test coverage is assessed and code checks take place on a regular and external basis.

*Table 28: Quality statements*

In addition, the companies in this research give qualitative statements on the improvements derived from the transition to DevOps. But there is no quantitative information to back up these statements: The next section will focus on quantitative data to validate if the transition to DevOps delivered success.

Of course, a company would like to investigate how customers view their services; the metric for this can be the app rating of their mobile applications.

The only company that delivered that information from an internal perspective was company C. Based on the information provided, the following tables give the results:

2015	4
2016	2
2017	4

Table 29: Company C: Internal app rating

From an external perspective, the company's app ratings were:

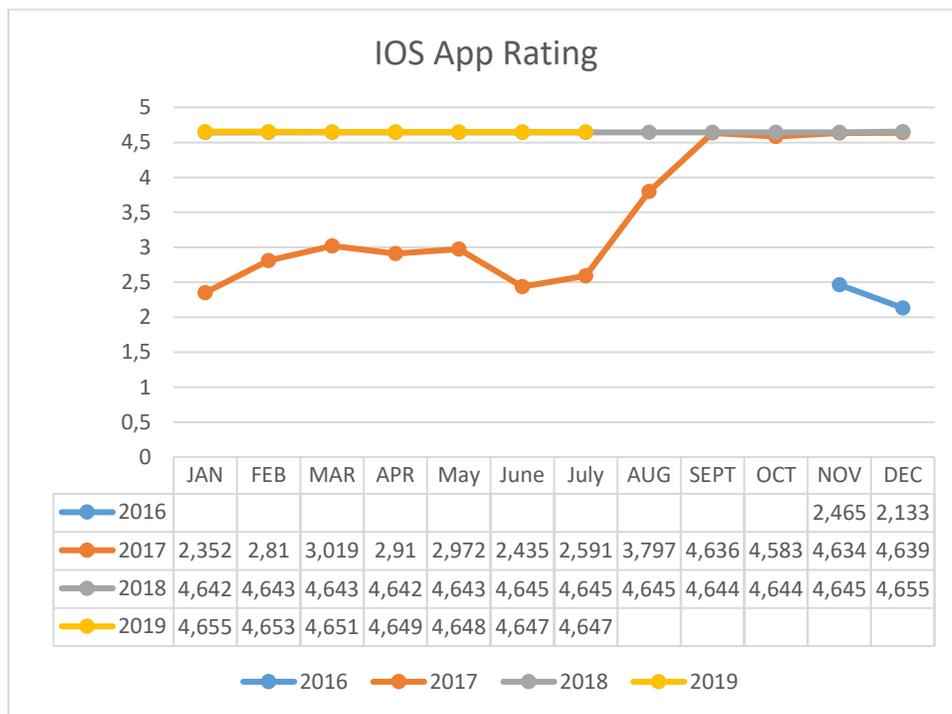


Figure 18: Company C: IOS app rating

A drop can be seen at the end of 2016, but the rating rises in 2017. Again, a major drop can be seen in June 2017 and then, in August 2017, the figures again improve. As stated previously, the data is not completely in line with the internal measurements, and it does not correlate with the transition to DevOps.

Examining data related to mobile application releases yields the following:

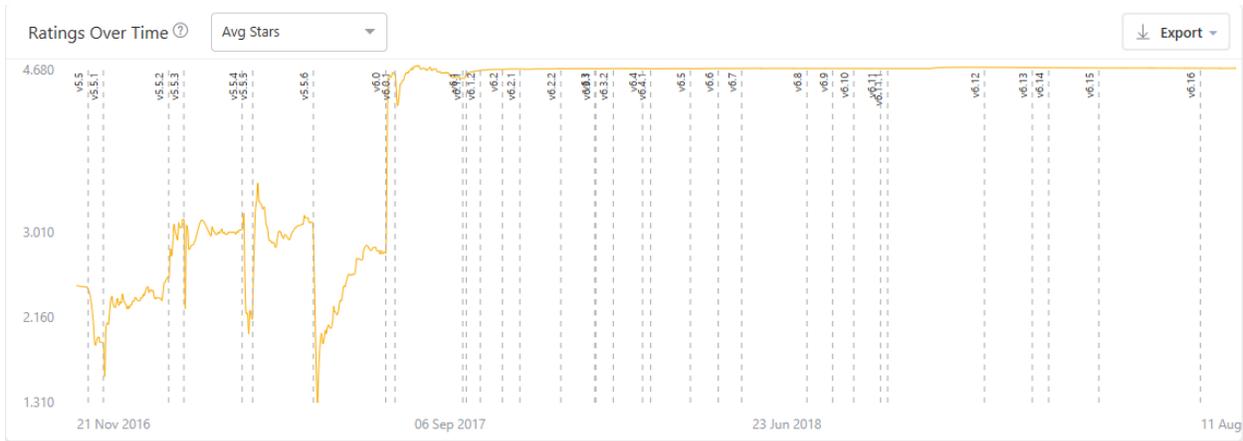


Figure 19: Company C: IOS app release frequency

The data shown in this table gives an indication of the release frequency, which increased in the July 2017 timeframe, as did the app ratings figures. This increase in release frequency and the increase in app ratings remains to this day.

There might be a correlation between the release frequency and the app rating and that could be a sign of a continuous feedback of the stakeholders, as described by Erich and others (2017).

Although the drop of the rating in 2016 could be explained by a change in tooling, the usage of the new solution could also explain the increase in app ratings.

Information related to the release schedule of company B's mobile application and their ratings indicated that there were no major drops in their app rating for their mobile applications.



Figure 20: Company B, IOS App Release Frequency

Data extracted from the IOS App Store yielded the following:

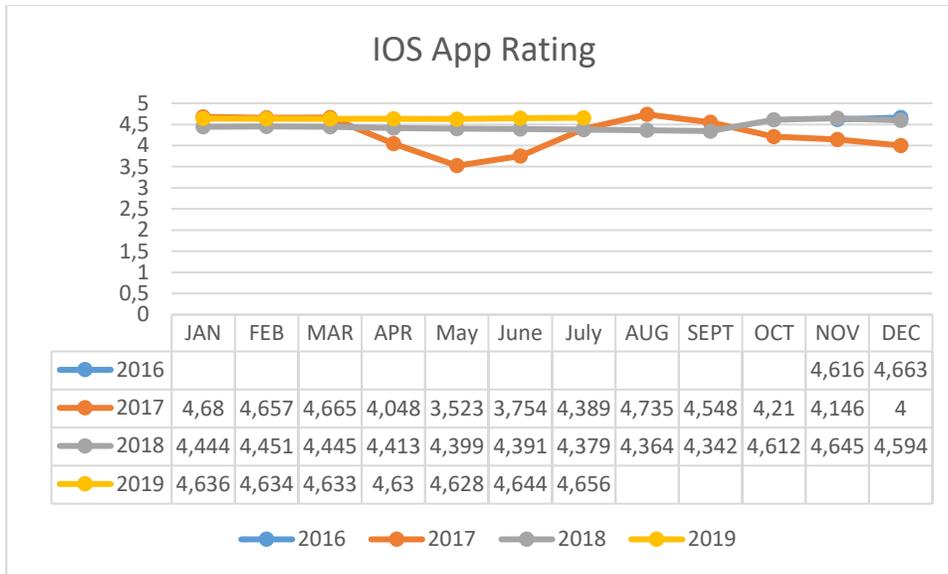


Figure 21: Company : IOS app rating

The trends for company B indicate that ratings dropped from April to June 2017; ratings recovered again in July 2017 and in August 2017, and remained stable for the rest of the measurement period.

Based on this information and the release schedule that is reflected in the tables, the conclusion can be drawn that company B did not have the ability to respond quickly to feedback; this is indicated because no cycle of frequent releases is shown during this period, whereas frequent release cycles emerge for 2018 and 2019. During this period, an increase of the app ratings to 4 can be seen in December 2017, rising to a 4.6 in July 2019. Accompanying this is data indicating that frequent releases took place from September 2018 to June 2019. This could also be an indication of the truth of the statement of Erich and others (2017), confirming the importance of continuous feedback from the stakeholders.

According to this data, company A has delivered frequent mobile releases since 2018; however, a drop in ratings is shown for the beginning of 2018.



Figure 22: Company: IOS app release frequency

An examination of the data revealed the following:

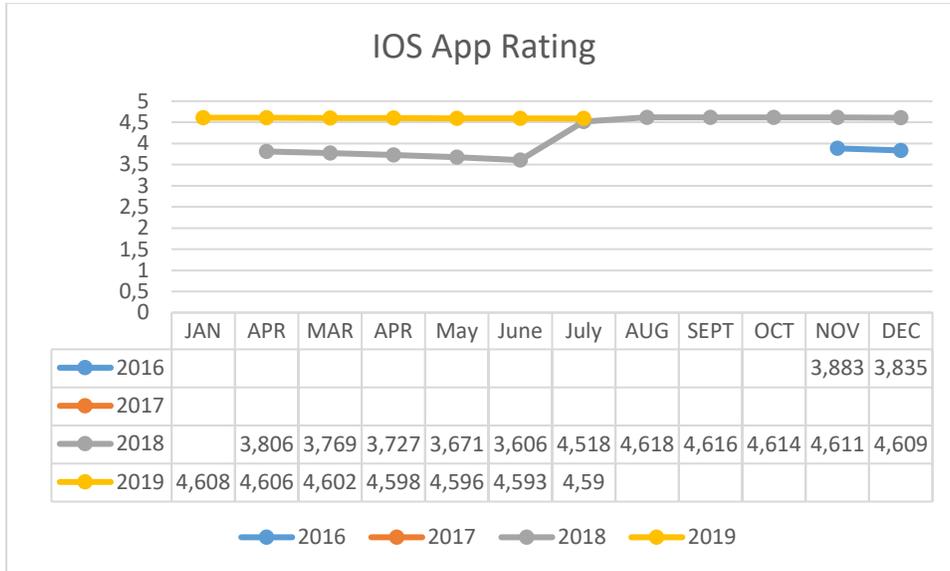


Figure 23: Company A, IOS App Rating

Similar patterns seem to emerge: that app ratings improve significantly when the release frequency grows. This information could also support the statement of Erich and others (2017) that continuous feedback from stakeholders is crucial to improving an organization’s performance.

## 7. Discussion

The results of the research clearly indicate that there are three important and crucial topics that need further investigation and discussion:

- development of more extensive definitions of DevOps and its relationship with release engineering;
- development of a maturity model for DevOps, one that will address the different practices, and will also show the impact of the different practices upon each other; and
- why the implementation of DevOps has been successful, and the relationship between release frequency and mobile application ratings.

As stated in chapter 1, the main question of this research is: “On what basis can companies be compared in terms of their success in implementing DevOps?”. Based on the topics stated above this research did not provide a basis to compare companies in their success in implementing DevOps.

### 7.1 Definitions of DevOps vs. the realities of actual implementation: Release engineering vs. DevOps

Although there are a myriad of definitions of (as shown in Chapter 3), they all focus on the dimensions summarized in the table below. In spite of the fact that all of the interviewees stated that the culture of an organization was important, during implementation, they tended to focus on the automation of the CI/CD-pipeline.

	Humble and Molesky 2011	Lwkatare and others (2016)	Dyck and others (2015)	Company A	Company B	Company C	Company D
<b>Culture</b>	"An important step is for operations to be involved in the design and transition of systems."	"Empathy, support and good working environment between development and operations"	"DevOps is an organizational approach that stresses empathy".	"DevOps is a combination of operations and development".	"We need people that can stand in for each other. So we need a developer with operator skills and an operator with development skills".	"DevOps combines the teams that are responsible for continuity with the teams that are responsible for change".	"The team is responsible from start to finish for the application. So, they will not code, test, and operate it. But they are also responsible for the functional requirements and retirement of the application".
<b>Automation</b>	"Automation of build, deployment, and testing is key to achieving low lead times and thus rapid feedback."	"Infrastructure and deployment automation"		"Automation of the SLDC is important."	"Automation is based on the book <i>Continuous Delivery</i> , by Jez Humble".	"We do have a CI/CD pipeline"	"Automation should help the company in the segregation of duties".
<b>Lean</b>					"The focus is on eliminating waste".		
<b>Measurement</b>	"Measurement includes monitoring high-level business metrics, such as revenue or end-to-end transactions per unit time. At a lower level, it requires careful choice of key performance indicators."	"Useful metrics" "Instrumenting application and aggregating data into insight"			They stated that "they do measure, but this is getting more attention currently".	"We do measure the app rating".	"We do not measure at the moment".
<b>Sharing</b>	"Development and operations celebrate a successful release together"	"Rethinking and reorientation of roles and teams in development and operations activities"	and cross-functional collaboration between development and operations within and between teams - especially development and IT-operations- in order to operate resilient systems and accelerate the delivery of changes"	Cooperation is mentioned as one of the main practices.	"You should trust each other in the team".	"We do work in squads and guilds and the people share knowledge in this model".	"The people in the team work side by side, that is how they share knowledge".

Table 30: An overview of the practices in theory and in the cases

Dyck and others (2015, p. 3), who have emphasized that the central goal of release engineering is the deployment of high-quality software, have delivered the following definition of release engineering: “Release engineering is a software engineering discipline concerned with development, implementation, and improvement of processes to deploy high-quality software reliably and predictably”.

All of the interviewees provided the researcher with their definitions of DevOps and, what is interesting, is that their definitions were not in line with definitions about release engineering. During the interviews, a question about the tooling used before and after the transition to DevOps was asked. In the answers to these questions, all the interviewees focused on the importance of the automated delivery pipeline; no changes had occurred in terms of coming up with solutions in terms of operations. One interviewee made the startling statement that operators changed more often than developers. This was considered to be a positive sign in terms of DevOps, but it can also be said that operators were forced to change. In a personal conversation, somebody stated that implementation of DevOps meant that development was taking over operations, and they were not treated like first-class citizens, which is one of the principles of DevOps (Bass 2015).

It is clear that, although the practitioner literature, e.g. the books, and articles written by Davis and Daniels (2016) and others, stresses the importance of the cultural aspects of DevOps, the focus has been and continues to be on automation.

During the interviews, the companies described their journey to DevOps as beginning with the implementation of Agile development, actions that led to the implementation of an automated pipeline. Once this was in place, they tended to focus on adding operators to the development team, eventually naming that team “DevOps”.

This approach might be in line with the definition of DevOps by Dyck and others (in their article “Towards definitions for release engineering and DevOps” (2015, p. 3): “DevOps is an organizational approach that stresses empathy and cross-functional collaboration within and between teams—especially development and operations—in software development organizations, in order to operate resilient systems and accelerate delivery of changes”.

Company B stated that the communications within teams were fine, but that communications between teams were definitely an issue that needed improvement. In other words, communications between the different DevOps teams were still fraught with difficulties and were not effective. But the communication inside the team, between both development and operations, personnel within the same team was good. In short, cross-functional communications were good, but communications between different teams were not, which is an indication that they are not in line with the definition of DevOps.

The data from the other companies indicated that, although the empathy and understanding between development and operations have risen, the focus of the teams and what the company actually measured were based principally on the delivery of changes. The interviewees confirmed this, stating that there had been a rise in availability and a decrease in incidents; however, no data had been supplied to corroborate this. But what was interesting was that almost every interviewee had in his/her possession data/metrics about the release frequency, a fact that clearly demonstrated that the focus was on the acceleration of delivery of changes rather than on resilience.

One of the questions on the questionnaire asked which tools the companies used and which of them had increased in importance. All of the interviewees stated that the tools used to automate the deployment had increased in importance and that those tools had been introduced during the initial stages of DevOps implementation.

The interviews indicated that the tools focused on monitoring did not increase in importance, and neither did the tools for monitoring. So, tools and innovations in this area did not receive increased emphasis, which is a sign that monitoring did not receive the attention it deserves.

The data clearly supports the contention that companies tended to focus on automation, and not on operating resilient systems. The principle actions that were stressed were related to the development, implementation, and the improvement of processes to deploy high-quality software reliably and predictably. In other words, companies tended to treat DevOps implementation as something that facilitated release engineering, and so did not stress the other qualities/skills that DevOps was meant to promote.

On the other hand, it can be said that each of the interviewees stated that the previously tense relationship between development and operations had grown to one of respect and collaboration. That data reflected that a significant change in culture had taken place, as had organizational changes. Developers and operators had often been combined into one team who worked side by side and shared information. That development would seem to point to the fact that DevOps practices, rather than a simple emphasis on release engineering, were taking place. However, those developments need to be investigated and backed up by data and assessment.

It is clear that the role of operations and their tooling need further research and analysis.

## 7.2 A DevOps Maturity Model: Towards a focus area maturity model

The current DevOps maturity model, as described in the scientific literature, provides a detailed description for automation only, and thus cannot really be said to constitute a proper DevOps maturity model. The Micro Focus DevOps maturity model, which uses a five-point scale and which was used during this research, is a fixed maturity model. The problem with this model is that it is not geared towards expressing a relationship between the different key processes.

During the interviews, all the companies explained that they had applied some kind of maturity model, although some of the interviewees were reluctant to admit this. Their reluctance stemmed from the fact that the maturity model was viewed as some kind of scoring card, that is, that the use of the model would impose some kind of prescriptive rules on the company and force them to implement DevOps in certain ways. In other words, the DevOps model was not a form of guidance, but a form of subtle coercion.

This view is corroborated to a certain extent by Poepelbuß and Röglinger (2011, p.3), who state that a maturity model fulfills a number of purposes:

- “Descriptive: A maturity model serves a descriptive purpose of use if it is used for “as-is” assessments, where the current capabilities of the entity under investigation are assessed with respect to given criteria.”

- “Prescriptive: A maturity model serves a prescriptive purpose if it indicates how to identify desirable maturity levels and provides guidelines on what measures can be undertaken in order to improve performance. “
- “Comparative: A maturity model serves a comparative purpose if it allows for internal or external benchmarking.”

When the interviewees described a maturity model for DevOps, a lot of the answers focused around the importance of the people involved and around cultural aspects. However, the answers also clearly reflected their focus on automation.

In their article, “The Design of Focus Area Maturity Models”, Marlies van Steenbergen, Rik Bos, Sjaak Brinkkemper, Inge van der Weerd, and Willem Beckers (2010) define a fixed-level maturity model as a maturity model with a fixed number of levels. An example of such a model would be the Capability Maturity Model. This type of model is ideal for the prescriptive and comparative purposes of Poepelbuß and Röglinger, where each maturity level is associated with a number of processes that have to be implemented. This model has been criticized because it is not geared towards expressing interdependencies between the processes making up the maturity levels.

Instead of a fixed level maturity model, van Steenbergen et al (2010) have created another type of maturity model called a “focus area maturity model”. A focus area maturity model is based on the principle that there are a number of focus areas that have to be developed in order for an organization to achieve maturity in a functional domain.

As stated previously, a maturity model for DevOps should not be a scoring card, because, as many interviewees stated, organizations grow into DevOps and every company grows differently. In that context, the different maturity models described in Chapter 5 should be examined as to their effectiveness.

The maturity model of Mohamed (2010) is a fixed model, and in addition tends to focus on only one process, the release process, which Dyck and others (2015, p.3) define as follows: “Release engineering is a software engineering discipline concerned with the development, implementation, and improvement of processes to deploy high-quality software reliably and predictably”. Thus Mohamed, in the researcher’s view, has created a maturity model for release engineering.

Lwakatare and others (2015) have described the different dimensions inherent to DevOps:

- collaboration: rethinking and reorientation of roles and teams in relation to development and operations activities;
- culture: empathy, support, and the presence of a good working environment between development and operations;
- automation: the existence of an effective infrastructure and of automation related to the deployment;

- monitoring: activities related to instrumenting application and to aggregating monitored data into insights; and
- measurement: the existence of mechanisms that deliver useful metrics.

Apart from the area of deployment process automation, the other dimensions are not described in Mohamed's maturity model.

The DevOps maturity model from Micro Focus is also a fixed model, which describes three main areas:

- People
- Process
- Technology

The area of "people" addresses the issues of collaboration and culture; the area of "technology" refers to the dimensions of automation and measurement. A key component missing in Micro Focus' maturity model is the whole area of monitoring.

Dyck and others (2015, p3) created the following definition for DevOps: "DevOps is an organizational approach that stresses empathy and cross-functional collaboration within and between teams—especially development and IT operations—in software development organizations, in order to operate resilient systems and accelerate the delivery of changes". These statements clearly prove that the DevOps model of Micro Focus is, indeed, a DevOps maturity model.

Van Steenbergen and others (2010) state that one of the limitations of fixed maturity models is that they are not geared towards expressing interdependencies between the processes making up the maturity levels. This kind of interdependency between processes is crucial to DevOps. The definitions and processes described by Dyck and others stress the need for companies to accelerate changes, something which can be achieved through automation release management processes. However, if small changes are not delivered on a frequent basis, changes to the entire development process will not be released in a timely fashion.

Kim (2014) describes the second wave in DevOps as involving constant feedback, that is, there is a need for information to be constantly delivered so that new problems can be prevented from arising; this means that in DevOps, there is a need for the constant flow of information so that the various levels in all their dimensions can influence each other.

The fact that both models do not deliver a proper framework for implementing and assessing the three dimensions of people, process, and technology means that further research needs to be carried out in order to generate and formulate an effective maturity model for DevOps.

The method used by Igor Stojanov , Oktay Turetken and Jos Trienekens (2015) during their creation of a maturity model for the scaled Agile framework (SAFe) (a set of organizational and workflow patterns to guide companies) is reported in their article , "A Maturity Model for Scaling Agile Development," and provides a sound starting point for anyone seeking to create a more nuanced and effective maturity model for DevOps. Although the SAFe maturity model created by Stojanov and others is a fixed model, the method they used will give guidance.

One example of a focus area maturity model is illustrated below; this model is based upon definitions of DevOps and the answers from the companies interviewed.

		Maturity Scale												
Focus Area		0	1	2	3	4	5	6	7	8	9	10	11	12
Dimension														
<b>Culture</b>	Operations and development are managed and are considered to form one team.													
	Each team member feels responsible for the complete lifecycle of the system.													
	Cross-functional training is promoted by management.													
<b>Automation</b>	Automation of the deployment process has occurred.													
	Automation of the deployment of infrastructure has taken place.													
<b>Lean</b>	The company focuses on eliminating waste.													
<b>Measurement</b>	Instrumentation of the applications to deliver monitoring insights has been put into place.													
	Measurement of the release frequency is taking place.													
	Measurement of the quality of delivery is taking place.													
	Measurement of the availability of the systems is occurring.													
<b>Sharing</b>	Development and operations celebrate their successes together.													
	Development and operations are fully cooperating and collaborating.													

Table 31: DevOps: A focus area maturity model

### 7.3 DevOps Success: Release frequency as the critical success indicator

NPS Scores				
	Company A	Company B	Company C	Company D
2013	Unknown	Unknown	12	Unknown
2014	Unknown	Unknown	17	Unknown
2015	-23	Unknown	33	Unknown
2016	-15	Unknown	36	-8
2017	-9	Unknown	53	-3
2018	-9	Unknown	57	0
Trust Index				
2015	4,2	4,1	4,3	4,4
2016	4,2	4,2	4,2	4,4
2017	4,4	4,3	4,3	4,5
2018	4,2	4,3	4,3	4,5
IOS App Rating				
2016	3,9	4,6	2,3	Unknown
2017	Unknown	4,1	3,4	Unknown
2018	4,2	4,6	4,6	Unknown
2019	4,6	4,6	4,7	Unknown
Year in which DevOps Was Implemented				
	2017	2012	2015	2019

Table 32: External Success Metrics

The data from the Apple App Store shows that an increase in release frequency delivers a better app rating. The ratings and the release frequency increases for the following companies occurred around the following dates:

- company A: 23<sup>rd</sup> of June 2018;
- company B: 1<sup>st</sup> of August 2018; and
- company C: 6<sup>th</sup> of September 2017.

Although this data is from the Apple App Store, it is not linked to information related to the Trust Monitor. No rise in trust can be proven and thus the trust rating in terms of mobile applications and the website has remained stable.

The interview revealed the following dates related to the companies' transition to DevOps:

- company A: 2017 to 2018;
- company B: 2012; and

- company C: 2015.

Company A clearly demonstrates that DevOps has had an effect on the mobile applications for IOS, at least in terms of the transition to DevOps; an increase in the release frequency correlates to the time periods listed.

The data on company B does not show that correlation: Based on the interviews, the transition to DevOps took place in 2012, while an increase in the release frequency in their mobile application started on the 1<sup>st</sup> of August 2018. Based on this information, there is no correlation between the transition to DevOps and an increase in the release frequency, but further research into this case is needed.

Company C stated in their interviews that they had implemented a new development platform for mobile applications in 2016, a development that resulted in a drop in their app ratings: that is congruent with the data shown in this thesis. It is possible that the new platform also had an effect on the frequency of releases. These facts could explain why this company reported that DevOps implementation of 2015 came only into effect in the second half of 2017.

There is a lack of data for company D, which experienced a major transition to DevOps in 2018. It would have been useful for the researcher to have had information about the company's app ratings, to see if this company experienced a lower release frequency and a lower app rating. Such data would have enabled a correlation to be established between the implementation of DevOps, a higher release frequency, and a better app rating; more research needs to be carried out on this company.

On the basis of the current maturity models, little difference in terms of how mature the implementation of DevOps is among the companies can be demonstrated. However, the interviews would seem to indicate that, in the minds of the interviewees, major changes should be apparent, especially given that these companies had implemented DevOps some years back.

The Apple IOS Store provided data that indicated that company A had begun a frequent release schedule in the middle of 2018, whereas company C started with a frequent release schedule in 2017. If automation and frequent releases are an important principle in DevOps, one would expect that a company that is scored as more mature in DevOps would have more frequent releases. Company B received the highest score and one would expect that an increase in release frequency would occur soon. However, this is not a valid point in terms of the efficacy of the current DevOps maturity model, and so this aspect should be further researched.

The app ratings are an external score and are, therefore, of interest to researchers. But the internal metrics are also important: that is, the number of incidents, the turnover of personnel, backed-out changes, etc., should be measured before and after the transition. That data should be part of further investigation.

## 8. Conclusions

The main question posed at the beginning of the research process was the following: How can companies be compared in terms of their success in implementing DevOps? An analysis of 4 different case studies yielded crucial information, as did interviews with various stakeholders, who were asked to describe their vision for the company: how they envisaged the implementation of DevOps taking place; and how they were measuring the entire process (specific metrics). In total, 12 interviews were conducted and were transcribed by the researcher and then approved by the interviewee. External sources were also consulted in search of validation. Based on the interviews, a timeline for the transition was defined; that timeline was used in comparison with the external sources in order to verify the success of DevOps.

### 8.1 Release engineering and app ratings

Based on the data, there was a link between release frequency and the rating in the IOS App Store. This is a positive effect. A release scheme of two weeks resulted in an increase of 0.5 points in the app rating. This correlation is seen in three different case studies.

So frequently release seems to have a positive effect on the ratings by users. Dyck and others (2015, p.3) have formulated the following definition of release engineering: “Release engineering is a software engineering discipline concerned with development, implementation, and improvement of processes to deploy high-quality software reliably and predictably”. If this definition is combined with a concomitant automation of deployment, the figures for release frequency would rise. The data obtained by the researcher demonstrates that this is what companies have achieved and has resulted in positive feedback.

The conclusions and observations made in this thesis should be investigated among a large set of companies.

### 8.2 Does DevOps matter?

The question, “Does IT matter” by Nicholas Carr created the question pursued by the researcher: “Does DevOps matter?”

All the evidence gathered, although this was limited to only four different cases, does **not** seem to prove that the implementation of DevOps delivers a successful outcome. The only external resource that indicated that some measure of success might have taken place was the information provided by the IOS app store. However, there was only one case where a link could be established between an increase in deployment frequencies and ratings and the implementation of DevOps. So, I can state with certainty that, “DevOps does not matter” in terms of the companies that were studied.

This could be due to the fact that the companies that were researched had already had a mature IT infrastructure and a history of usage of IT. This issue and why these findings arose need to be further researched.

The investigation and analysis also did not provide a justification or an explanation as to why these companies had implemented DevOps. Of course, it can be argued that the question had not been specifically included in the questionnaire. However, even though questions were included that asked for information about metrics, they did not lead to a discussion explaining the reasoning behind the transformation to DevOps. The interviewees did not appear to be aware of why the transition to DevOps was taking place, a state of affairs that contradicts the concepts behind DevOps, in that DevOps should be based on the principles of value management and on the assumptions that value should be delivered to customers and clients. Further research needs to be carried out as to why a transition to DevOps is recommended, in that there might be other reasons, other than that of business success, for transitioning to this set of practices.

Based on an investigation as to the reasons for implementing DevOps, the companies might deliver useful metrics from before and after the transition. So far, this is not happening. The companies have delivered very limited metrics before and after the implementation of DevOps.

### 8.3 Current DevOps maturity models are inadequate

The research carried out yielded only one scientific DevOps Maturity model, that of Mohamed (2015, 2016). This model focuses on automation only and therefore fails to address the important aspects raised by Humble and Molesky's five dimensions of DevOps implementation (2011). Vital information thus remains to be discovered, reported, and discussed.

The other maturity model that was used had been created by Micro Focus. Although it did touch upon the five dimensions described by Humble and Molesky, it failed to address and provide information on the dimension measurement as described by Lwakatare and others (2016).

Both models that were used during the course of this research were fixed maturity models, a fact that imposes huge restrictions, as pointed out by Steenbergen and others (2010, p.317) in the following comment: "Fixed Maturity Models are not geared towards expressing interdependencies between the processes making up the maturity levels".

It was clear that the research led to the conclusion that the models did not provide standards that enabled discrimination/differentiation between elements of performance to be assessed. Company A and Company D, both of which began DevOps implementation only recently, seemed to achieve the same level of maturity as company C, which had implemented DevOps 7 previously.

For all of the above reasons, further research into DevOps maturity models needs to be carried out and new models, ones that enable more nuanced and multi-dimensional assessments and interpretations to be carried out, must be developed. Only then can we confirm that DevOps works and why. Only then can an argument promoting the success of DevOps be made, urging companies to embrace this transformative set of practices and policies.

## 9. Comments and recommendations for future research

In this section the research should be evaluated and guidance for further research should be stated. This section will be divided into two main sections:

- Threats to validity
- Future work

### 9.1 Limitations to the validity of the research

#### 9.1.1 The research was limited to only one market space

Although the number of cases appears small, the number of interviews was quite extensive for the Dutch financial market. In two major external sources, that data was limited to only three cases: Although the market was comparable for these companies, it is important to remember that they were operating in only one market. Perhaps in different markets, different results as to DevOps success might emerge.

#### 9.1.2 The research was limited to companies in transition to DevOps

All of the companies selected had decided to transition to DevOps. It would have been optimal for the researcher to have studied at least one company that had not transitioned to DevOps, thus enabling more specific comparisons to have been made about the success of DevOps.

#### 9.1.3 All companies have an IT legacy

All companies in this research had an IT legacy, that is, they had a huge installed base of IT services before they transitioned to DevOps. That fact could have affected the implementation of DevOps.

#### 9.1.4 Limited internal metrics for DevOps

All of the companies in this research supplied internal metrics only in relation to the release frequency before and after the transition. All of them demonstrated an increase in the release frequency. The other metrics they supplied were extremely limited; it is thus possible that DevOps could have had a substantial effect on some internal metrics that were not obviously perceptible. Thus it is possible that DevOps had had a huge effect upon some internal metrics that are currently not reflected in the results, thus providing companies with a valid reason to transition to DevOps.

## 9.2 Future work

Some of the limitations in terms of this investigation and analysis need to be addressed in the future, so that the effectiveness and success of DevOps implementation can be more thoroughly studied and proven. I would recommend that future research:

- be expanded and that it analyze and investigate multiple markets;
- scrutinize companies that explicitly have chosen not to transition to DevOps;
- examine “DevOps-native” companies, that is, those companies, such as Fintech start-ups, that have implemented DevOps from the very beginning; and
- analyze how data related to the internal metrics assessing DevOps success can be better measured.

If the steps described above are undertaken, the validity of the research that has been carried out in relation to this thesis will be supported. In addition, it is critical that more research into maturity models be carried out so that a better and more effective model can be created.

## 10. References

- ABN AMRO. Annual Report (2016).  
[https://www.abnamro.com/en/images/Documents/050\\_Investor\\_Relations/FinancialDisclosures/2016/ABN\\_AMRO\\_Group\\_Annual\\_Report\\_2016.pdf](https://www.abnamro.com/en/images/Documents/050_Investor_Relations/FinancialDisclosures/2016/ABN_AMRO_Group_Annual_Report_2016.pdf). Accessed on 27-07-2019.
- ABN AMRO. Annual Report (2018).  
[https://www.abnamro.com/nl/images/Documents/050\\_Investor\\_Relations/FinancialDisclosures/2018/ABN\\_AMRO\\_Group\\_Annual-Report\\_2018.pdf](https://www.abnamro.com/nl/images/Documents/050_Investor_Relations/FinancialDisclosures/2018/ABN_AMRO_Group_Annual-Report_2018.pdf). Accessed on 27-07-2019.
- Andreesen, M., (2011). Why Software is eating the world. *Wall Street Journal*.  
<http://www.wsj.com/articles/SB10001424053111903480904576512250915629460>.  
Accessed on 20-08-2018.
- Abrahamsson, P., Conbon K., & Wang X (2009). Lots Done, More To Do: The Current State of Agile Systems Development Research. *European Journal of Information Systems*, 2009. (18): 281-284: doi:10.1057/ejis.2009.27.
- Agile Manifesto (2019). Manifesto for Agile Software development. <https://agilemanifesto.org/>  
Accessed at 26<sup>th</sup> of October
- Atlassian (2019). DevOps Assessment. <https://www.atlassian.com/devops/maturity-model>.  
Accessed on 17-04-2019.
- Bass L., Weber I., & Zhu I. (2015). *DevOps. A Software Architect's Perspective*. Old Tappan, New Jersey, USA. Pearson Education. Print ISBN: 978-0134049847.
- Banken.nl (2018). Mobiel bankieren apps spelen steeds meer in op wensen van de consumenten. <https://www.banken.nl/nieuws/21000/mobiel-bankieren-apps-spelen-steeds-meer-in-op-wensen-van-consumenten>. Accessed on 27-07-2019.
- Boehm B. (1976). Software Engineering. *Journal IEEE Transactions on Computers*, volume 25 issue 12, December 1976, 1226-1241: doi>[10.1109/TC.1976.1674590](https://doi.org/10.1109/TC.1976.1674590).
- Breno B. Nicolau de França, Helvio Jeronimo, Junior, and Guilherme Horta Travassos (2016). Characterizing DevOps by Hearing Multiple Voices. *Proceedings of the 30th Brazilian Symposium on Software Engineering (SBES '16)*. Eduardo Santanda de Almeida (Ed.). ACM, New York, New York, USA, 53-62. doi: <https://doi.org/10.1145/2973839.2973845>.
- Carr N. (2003). IT Doesn't Matter. *Harvard Business Review*, May 2003: <https://hbr.org/2003/05/it-doesnt-matter>. Accessed on 20-08-2018
- Carr N. (2004). *Does IT Matter?*. Boston, Harvard Business School Press. ISBN 9781591394440.
- Chen L. Continuous Delivery: Huge Benefits, but Challenges Too. *IEEE Software*, Volume 32, no. 2, 50-54, March-April 2015. doi: 10.1109/MS.2015.27.

- Clerc V. & Niessink F. (2004). *IT Service CMM, A Pocket Guide*. Van Haren Publishing. Zaltbommel. ISBN 9789077212356.
- CMMI Institute (2019). Introducing CMMI V2.0. <https://cmmiinstitute.com/cmmi> accessed on 15-10-2019
- DASA (2017). DASA DevOps Principles: <http://www.devopsagileskills.org/>. Accessed on 21-2-2017.
- Davis J. & Daniels R. (2016). *Effective DevOps*. O'Reilly Media Inc, Sebastopol. ISBN 978-1491926307.
- Dyck A., Penners R., & Lichter H. (2015). Towards Definitions for Release Engineering and DevOps. RELENG 2015. Florence. doi: <https://doi.org/10.1109/RELENG.2015.10>.
- Edwards, D. (2012). The (Short) History of DevOps. <https://www.youtube.com/watch?v=o7-luYS0iSE>.
- Erich F., Amrit C., & Daneva M. (2014). *Report: DevOps Literature Review*. University of Twente: [http://floriserich.nl/wordpress/wp-content/uploads/2017/06/erich2014\\_devops\\_report.pdf](http://floriserich.nl/wordpress/wp-content/uploads/2017/06/erich2014_devops_report.pdf).
- Erich F., Amrit C., & Daneva M. (2014). *Cooperation Between Software Development and Operations: A Literature Review*. ESEM 2014. Torino: doi: [10.1145/2652524.2652598](https://doi.org/10.1145/2652524.2652598).
- Erich F., Amrit C., & Daneva M. (2017). A qualitative study of Devops usage in practice. *Journal of Software Evolution and Process*, June 2017: <https://doi.org/10.1002/smr.1885>.
- Gartner (2019). Gartner Glossary. <https://www.gartner.com/en/information-technology/glossary/devops> Accessed on 15-10-2019
- Freeform Dynamics (2015). Exploiting the software advantage: Lessons from Digital Disrupters: October 2015: [https://freeformdynamics.com/wp-content/uploads/2017/10/15-10\\_Exploiting\\_the\\_Software\\_Advantage.pdf](https://freeformdynamics.com/wp-content/uploads/2017/10/15-10_Exploiting_the_Software_Advantage.pdf).
- Humble J. & Farley D. (2011). *Continuous Delivery: Reliable Software Release Through Build, Test and Deployment Automation*. Pearson Education. Boston, USA. ISBN 978-0321601919.
- Humble J. & Molesky J. (2011). Why Enterprises Must Adopt DevOps to Enable Continuous Delivery. *Cutter IT Journal*, August 2011, Volume 24 (8). <https://www.cutter.com/article/why-enterprises-must-adopt-devops-enable-continuous-delivery-416516>.
- IMD (2019). Digital Vortex 2019. <https://www.imd.org/contentassets/d4b328f064c844cd864a79369ba8405a/digital-vortex.pdf>. Accessed on: 12-11-2019
- Jones, S., Noppen, J., and Lettice, F. (2016). Management Challenges for DevOps Adoption within UK SMEs. *Proceedings of the 2nd International Workshop on Quality-Aware DevOps (QUDOS 2016)*. ACM, New York, NY, USA, 7-11. doi: <http://dx.doi.org/10.1145/2945408.2945410>.
- Jonker J., & Pennink B. (2004). *De kern van methodologie*. Koninklijke Van Gorcum, Assen. ISBN 9789023235415.

- Jonker J. & Pennink B. (2009). *The Essence of Research Methodology*. Springer, Heidelberg. ISBN 9783540716587.
- Kavis, M. (2017). The Four Stages of DevOps Maturity. <https://www.forbes.com/sites/mikekavis/2017/11/17/the-four-stages-of-devops-maturity/#b52e0a12f625>. Accessed on 14-07-2019.
- Kim G. (2019). Gene Kim. <https://itrevolution.com/faculty/gene-kim/> Accessed on 30-10-2019
- Kim G., Behr K., & Spafford G. (2014). *The Phoenix Project*. IT Revolution Press, Portland. ISBN ISBN 978-0988262591.
- Kim G., Humble J., Dubois P., & Willis J. (2016). *The DevOps Handbook*. IT Revolution Press, Portland. ISBN 9781942788003.
- Kniberg H. and Ivarson A. (2012). Scaling Agile @ Spotify. <https://creativeheldstab.com/wp-content/uploads/2014/09/scaling-agile-spotify-11.pdf> accessed on 30-10-2019
- Lwakatare L.E., Kuvaja P., Oivo M. (2015). Dimensions of DevOps. In: *Agile Processes in Software Engineering and Extreme Programming*, (eds): Lassenius C., Dingsøyr T., and Paasivaara M. XP 2015. *Lecture Notes in Business Information Processing*, Volume 212. Springer Cham: [https://doi.org/10.1007/978-3-319-18612-2\\_19](https://doi.org/10.1007/978-3-319-18612-2_19).
- Lwakatare L.E., Kuvaja P. & Oivo M. (2016). *An Exploratory Study of DevOps: Extending the Dimensions of DevOps with Practices*. ICSEA 2016. Rome. [https://www.researchgate.net/profile/Luigi\\_Lavazza/publication/307576316\\_ICSEA\\_2016\\_The\\_Eleventh\\_International\\_Conference\\_on\\_Software\\_Engineering\\_Advances/links/57c9a36a08ae3ac722af8728.pdf#page=105](https://www.researchgate.net/profile/Luigi_Lavazza/publication/307576316_ICSEA_2016_The_Eleventh_International_Conference_on_Software_Engineering_Advances/links/57c9a36a08ae3ac722af8728.pdf#page=105)
- Mansor Z., Yahya S., & Habibah Arshad N. (2013). Empirical Study of Cost Management Success Determinants in Agile Based Software Development Project: A Rash Measurement Model Analysis. *Social and Behavioral Sciences* 107 (2013), 129–135. <https://doi.org/10.1016/j.sbspro.2013.12.408>.
- McCarthy, M.A., Herger, L. M., Khan, S. M., and Belgodere, B. M. Composable DevOps: Automated Ontology-Based DevOps Maturity Analysis. *2015 IEEE International Conference on Services Computing*, New York, NY, 2015, pp. 600-607. doi: 10.1109/SCC.2015.87.
- McChrystal S. (2015). *Team of Teams: New Rules of Engagement for a Complex World*. Portfolio/Penguin, New York, New York. ISBN 978-1591847489.
- Micro Focus (2019). What is DevOps. <https://www.microfocus.com/en-us/what-is/devops> Accessed on 30-10-2019

- Micro Focus (2019). Datasheet – DevOps Solution Discovery Workshop.  
[https://www.microfocus.com/media/datasheet/devops\\_solution\\_discovery\\_workshop\\_ds.pdf](https://www.microfocus.com/media/datasheet/devops_solution_discovery_workshop_ds.pdf) Accessed on 17-04-2019
- Mohamed S. (2015). DevOps shifting software engineering strategy-value based perspective. *IOSR Journal of Computer Engineering* (volume 17, issue 2: 51-57).  
[http://scholar.msa.edu.eg:93/sites/default/files/dr\\_samer\\_ibrahim/files/devops\\_shifting\\_software\\_engineering\\_strategy-value\\_based\\_perspective-iosr\\_jce.pdf](http://scholar.msa.edu.eg:93/sites/default/files/dr_samer_ibrahim/files/devops_shifting_software_engineering_strategy-value_based_perspective-iosr_jce.pdf).
- Mohamed S. (2016). DevOps Maturity Calculator DOMC – Value-Oriented Approach. *International Journal of Engineering Research & Science*, volume 2, issue 2: 25-35.  
[https://s3.amazonaws.com/academia.edu.documents/43309208/IJOER-FEB-2016-18.pdf?response-content-disposition=inline%3B%20filename%3DDevOps\\_Maturity\\_Calculator\\_DOMC\\_-\\_Value\\_o.pdf&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-credential=AKIAIWOWYYGZ2Y53UL3A%2F20191123%2Fus-east-1%2Fs3%2Faws4\\_request&X-Amz-Date=20191123T122114Z&X-Amz-Expires=3600&X-Amz-SignedHeaders=host&X-Amz-Signature=74e311887ae34e223335eb1c3b24cac1968fcfe271c4dbd30c91beef55d510b8](https://s3.amazonaws.com/academia.edu.documents/43309208/IJOER-FEB-2016-18.pdf?response-content-disposition=inline%3B%20filename%3DDevOps_Maturity_Calculator_DOMC_-_Value_o.pdf&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-credential=AKIAIWOWYYGZ2Y53UL3A%2F20191123%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20191123T122114Z&X-Amz-Expires=3600&X-Amz-SignedHeaders=host&X-Amz-Signature=74e311887ae34e223335eb1c3b24cac1968fcfe271c4dbd30c91beef55d510b8)
- De Nederlandse Vereniging van Banken (2015). Vertouwensmonitor.  
<https://www.nvb.nl/media/1101/vertouwensmonitor-banken-2015.pdf>  
 Accessed on 01-08-2019.
- De Nederlandse Vereniging van Banken (2016). Vertouwensmonitor.  
<https://www.nvb.nl/media/1101/vertouwensmonitor-banken-2016.pdf>. Accessed on 01-08-2019.
- De Nederlandse Vereniging van Banken (2017). Vertouwensmonitor.  
<https://www.nvb.nl/media/1101/vertouwensmonitor-banken-2017.pdf>. Accessed on 01-08-2019.
- De Nederlandse Vereniging van Banken (2018). Vertouwensmonitor.  
<https://www.nvb.nl/media/1101/vertouwensmonitor-banken-2018.pdf>.  
 Accessed on 01-08-2019.
- O'Reilly (2009). Velocity Conference 2009.  
<https://conferences.oreilly.com/velocity/velocity2009> Accessed on 30-10-2019
- Paulk M. (2009). A History of the Capability Maturity Model for Software, *Software Quality Professional*, vol. 12 no 1: pages 5-19.  
<https://pdfs.semanticscholar.org/6fb0/c324e08698a9e364693151605a74982b487a.pdf>
- Paulk M., Weber C., Curtis B., & Chrissis M. (1995).. *The Capability Maturity Model* Boston, Addison-Wesley. ISBN 9780201546644.
- Peuraniemi T. (2014). Review: DevOps, value-driven principles, methodologies and tools. SEMINAR NO. 58314308. Helsinki available at

[https://www.researchgate.net/profile/Juergen\\_Muench/publication/273064010\\_Data\\_and\\_Valuedriven\\_Software\\_Engineering\\_with\\_Deep\\_Customer\\_Insight/links/54f6ab6f0cf27d8ed71e6023/Data-and-Value-Driven-Software-Engineering-with-Deep-Customer-Insight.pdf#page=46](https://www.researchgate.net/profile/Juergen_Muench/publication/273064010_Data_and_Valuedriven_Software_Engineering_with_Deep_Customer_Insight/links/54f6ab6f0cf27d8ed71e6023/Data-and-Value-Driven-Software-Engineering-with-Deep-Customer-Insight.pdf#page=46)

Phillips M., & Shrum S. (2010). Process improvement for all: What to expect from CMMI version 1.3. *The Journal of Defense Software Engineering*. January 2010, 10-14.

<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.170.3408&rep=rep1&type=pdf>.

Pöppelbuß, J., & Röglinger, M. (2011). What Makes a Useful Maturity Model? A Framework of General Design Principles for Maturity Models and its Demonstration in Business Process Management. *ECIS 2011 Proceedings* 28. <https://aisel.aisnet.org/ecis2011/28>.

Rabobank (2015). Jaarverslag.

[https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=2ahUKEWjGrNb1bTjAhWM5KQKHYYW9AXIQFjABegQIBhAC&url=https%3A%2F%2Fwww.rabobank.com%2Fnl%2Fimages%2Frabobank-jaarverslag-2015.pdf&usg=AOvVaw2CpAWzGu1\\_w3nY-nyJtD\\_5](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=2ahUKEWjGrNb1bTjAhWM5KQKHYYW9AXIQFjABegQIBhAC&url=https%3A%2F%2Fwww.rabobank.com%2Fnl%2Fimages%2Frabobank-jaarverslag-2015.pdf&usg=AOvVaw2CpAWzGu1_w3nY-nyJtD_5). Accessed on 27-07-2019.

Rabobank (2017). Infographic.

<https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=2ahUKEWjqtLp1LTjAhWQ5KQKHYZWAWgQFjAAegQIBBAC&url=https%3A%2F%2Fwww.rabobank.com%2Fnl%2Fimages%2F09-infographic-rabobank-in-2017-nl.pdf&usg=AOvVaw0AR1ir1cnsQjO7Po4LyNuk>. Accessed on 27-07-2019.

Rabobank (2018). Infographic.

<https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=2ahUKEWjCs5KY1bTjAhWCy6QKHbnEAY8QFjABegQIAxAC&url=https%3A%2F%2Fwww.rabobank.com%2Fnl%2Fimages%2F04-infographic-rabobank-in-2018-nl.pdf&usg=AOvVaw1g9BxGI3YC4Z2xc4EcmfpR>. Accessed on 27-07-2019.

Ramtin J., Bin Ali N., Petersen K., & Tanveer B. (2016). What is Devops? A systematic mapping study on definitions and practices. XP '16 Workshops. *Proceedings of the Scientific Workshop Proceedings of XP2016*. Article No. 12. doi: [10.1145/2962695.2962707](https://doi.org/10.1145/2962695.2962707).

Rakesh, R. and Mirosław, S. (2016). First International Workshop on Emerging Trends in DevOps and Infrastructure. In *Proceedings of the Scientific Workshop Proceedings of XP2016* (XP '16 Workshops). ACM, New York, NY, USA, Article 11, doi:<https://doi.org/10.1145/2962695.2962706>.

Rapaport R. (2014). A short history of DevOps. <http://www.ca.com/us/rewrite/articles/devops/a-short-history-of-devops.html>. Accessed on 30-10-2016.

Ries E. (2008), What is customer development?.

<http://www.startuplessonslearned.com/2008/11/what-is-customer-development.html> Accessed on 30-11-2019

- Riungu-Kalliosaari L., Mäkinen S., Lwakatare L.E., Tiihonen J., Männistö T. (2016) DevOps Adoption Benefits and Challenges in Practice: A Case Study. In: Abrahamsson P., Jedlitschka A., Nguyen Duc A., Felderer M., Amasaki S., Mikkonen T. (eds) Product-Focused Software Process Improvement. PROFES 2016. Lecture Notes in Computer Science, vol 10027. Springer Cham. Doi: [https://doi.org/10.1007/978-3-319-49094-6\\_44](https://doi.org/10.1007/978-3-319-49094-6_44)
- Siakas K., & Siakas E. (2007). The Agile Professional Culture: A Source of Agile Quality. *Software Process Improvement and Practice Journal*, Volume 12, issue 6, pages 597 – 610. <https://doi.org/10.1002/spip.344>.
- Sharma S., & Coyne B. (2015). *Devops for Dummies*. Hoboken, Wiley. Available at: <https://www.ibm.com/ibm/devops/us/en/resources/dummiesbooks/>.
- Saunders M., Lewis P., & Thornhill A. (2016). *Research Methods for Business Students*. Pearson Education, Essex. ISBN 9781292016627.
- Soni, M. (2015). End to End Automation on Cloud with Build Pipeline: The Case for DevOps in Insurance Industry, Continuous Integration, Continuous Testing, and Continuous Delivery. 2015 IEEE, International Conference on Cloud Computing in Emerging Markets (CCEM), Bangalore, 2015, pp. 85-89. doi: 10.1109/CCEM.2015.29.
- Steenbergen van M., Bos R., Brinkkemper S., Weerd van de, I., and Bekkers W. (2010). The Design of Focus Area Maturity Models. In: Winter R., Zhao J.L., Aier S. (eds): *Global Perspectives on Design Science Research*. DESRIST 2010. Lecture Notes in Computer Science, Volume 6105. Springer, Berlin, Heidelberg. doi: [https://doi.org/10.1007/978-3-642-13335-0\\_22](https://doi.org/10.1007/978-3-642-13335-0_22).
- Stojanov I., Turetken O., and Trienekens, J. J. M. (2015). A Maturity Model for Scaling Agile Development. *41st Euromicro Conference on Software Engineering and Advanced Applications*, Funchal, pp. 446-453. doi: 10.1109/SEAA.2015.29.
- Survuz (2019). IT Service CMM. <http://www.survuz.com/frameworks/it-service-cmm-it-service-capability-maturity-model#.Xd98fW5Fx9O> Accessed on 1-12-2019
- Swartout P. (2012), *Continuous Delivery and DevOps: A Quickstart Guide*. Packt Publishing. Birmingham. ISBN 978-1849693684.
- The Open Group (2019). IT4IT Forum. <https://www.opengroup.org/it4it> Accessed on 1-12-2019
- Valentic B. (2014), Known Errors repetitio est mater studiorum? Not in this case <https://advisera.com/20000academy/blog/2014/02/04/known-errors-repetitio-est-mater-studiorum-case/> Accessed on 1-12-2019
- Velasquez N., Kim G., Kersten N., & Humble J. (2014). *2014 State of DevOps Report*, Puppet Labs. [https://www.researchgate.net/publication/263198947\\_2014\\_State\\_of\\_DevOps\\_Report](https://www.researchgate.net/publication/263198947_2014_State_of_DevOps_Report)

- Volksbank (2016). Jaarverslag 2016. <https://www.devолksbank.nl/investor-relations/jaarverslagen>. Accessed on 27-07-2019.
- Volksbank (2018). Jaarverslag 2018. <https://www.devолksbank.nl/investor-relations/jaarverslagen>. Accessed on 27-07-2019.
- Willis J. (2010). What DevOps Means to Me. <https://blog.chef.io/2010/07/16/what-devops-means-to-me/>. Accessed on 30-10-2019.
- Willis J. (2012). The Convergence of DevOps. <https://itrevolution.com/the-convergence-of-devops>. Accessed on 30-10-2019.
- Wikipedia (2019). Deontic logic. [https://en.wikipedia.org/wiki/Deontic\\_logic](https://en.wikipedia.org/wiki/Deontic_logic) Accessed on 20-11-2019
- Wikipedia (2019). Net Promoter. [https://en.wikipedia.org/wiki/Net\\_Promoter](https://en.wikipedia.org/wiki/Net_Promoter) Accessed on 21-11-2019
- Yin R. (2014). *Case Study Design*. Sage Publications Ltd, London. ISBN 9781452242569.
- Yokoi T., Wade M., & Macaulay J. (2019). Digital Vortex 2019. <https://www.imd.org/research-knowledge/reports/digitalvortex2019/>. Accessed on 18-07-2019.

---

## Appendix A: Questions/Questionnaire for the Interviews

---

### *Personal introduction*

1. What is your role? Tell me a little bit about your role and your educational background.
2. How many years of experience do you have with software development?
3. How many years of experience do you have with IT Operations?

### *Background: Software development and life cycle now*

4. Are you familiar with the Software Development Life Cycle (SDLC)?
5. From your perspective, what role does SDLC in play your organization? (What are steps from idea to production?)
6. How do you see your role in the SDLC?

### *Definitions of DevOps*

7. Are you familiar with the concept of DevOps?
8. What is your definition of DevOps? What are the most important aspects of DevOps in your estimation?
9. Do you have experience with DevOps? If so, for how long?
10. Are you using DevOps at the current time? How did you start applying DevOps?
11. How do you see your role in DevOps and the company's transition to it?

### *Current state of the process: After the transition to DevOps*

12. What does your SDLC process look like now (name 5 to 10 steps)?
13. What are the challenges your organization faces in the current setup (after the transition to DevOps)?
14. Can you describe the relationship between development and operations after the transition?
15. How do you share knowledge between development and IT operations?

*Describe your transition to DevOps? (IMPORTANT: Ask for 5-10 concrete steps, e.g. give them paper, whiteboard)*

16. Do you believe there is such a thing as DevOps maturity? If so, can you describe the steps to maturity?

### *Earlier stages of the process: Before the transition to DevOps*

17. What did your SDLC process look like before the transition to DevOps (name 5-10 steps)?
18. What were the challenges your organization faced in the previous setup (before transition to DevOps)?

19. What were the challenges you faced in the previous setup (before the transition to DevOps)?
20. How did you share knowledge between development and IT operations?
21. Can you describe the relationship between development and operations before the transition?

*Measuring impact*

22. How do you measure success/value provided by your software products (concrete KPIs)?
23. What is your turnaround time from idea into software?
24. How frequently do you deploy? How frequently do you deploy into production?
25. How do you measure the quality of the software in production? Did this improve (ratings, no. of incidents after a change/upgrade)?
26. Did the implementation of DevOps have an impact on that success? If yes, how (have app ratings improved since you adopted DevOps)? Can you provide statistics/numbers)?
27. What is the waiting time inside your software development lifecycle?
28. What is the current turnaround rate of IT personnel compared to the time before the transition (leaving the company)?

*Future state of the process: Trends*

29. How do you see your organization evolving?
30. How do you see your own role evolving?