Aalto-yliopisto
Kauppakorkeakoulu

Challenges in Adopting a Devops Approach to Software Development and Operations

MSc program in Information and Service Management

Maisterin tutkinnon tutkielma

Joonas Hamunen

2016

Tieto- ja palvelutalouden laitos
Aalto-yliopisto
Kauppakorkeakoulu

Aalto University
School of Business

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi

Abstract of master's thesis

Author Joonas Hamunen

Title of thesis Challenges in Adopting a Devops Approach to Software Development and Operations

Degree Master of Science in Economics and Business Administration

Degree programme Information and Service Economy

Thesis advisor(s) Matti Rossi

Year of approval 2016 **Number of pages** 69 **Language** English

Abstract

The constantly changing business needs and the requirement for faster time to market with software of present day has created a paradigm shift towards a 3rd generation Software Development philosophy called Devops. The lack of collaboration between IT Operations and Software Development as well as mismatch in configuration between development, testing and production environment has made deploying software releases slow and painful for many organizations. Different incentives between teams makes it difficult to work towards a common goal of bringing added value to customers.

A Devops approach to software development brings down the walls between the teams and align incentives through a collaborative culture, automation, lean principles, measurement practices and sharing. The benefits of Devops have been shown to be substantial with a significantly faster time to market and increased software stability. The organizational change is substantial which makes the challenges in adopting Devops an interesting topic to research. This thesis studies the challenges of Devops by interviewing nine experts who had been involved with Devops initiatives in their companies.

The qualitative study was conducted by semi-structured theme interviews on nine industry professionals who had hands-on experience with Devops implementations.

The findings were divided into four main challenge categories based on their topic. Due to the novelty of the approach, the concept of Devops for many is unknown or biased which hurts the overall implementation of practices. The lack of support in both management and organizational levels is a hindrance, since especially changing culture needs strong support and organizational buy-in in order to succeed. The toolset needed for Devops is particularly diverse and finding the fit, correct usage and attitudes towards that technology is challenging. Finally, when shifting to Devops that requires a certain level of lean principles and agility, aligning existing organizational processes such as the change management process to accommodate the new way of working was found challenging.

The implications of study yield four action points to help overcome the challenges found. Clearing misconceptions and spreading the knowledge of Devops helps overcome the lack of awareness challenge. Additionally, building commitment and trust in both management and team-levels getting Devops through the door in an organization. Establishing common ways of working and leading by example helps to overcome the challenge of fragmented technologies and reluctant attitudes towards it. Finally, ensuring the flexibility of the organization is key in order to prevent bottlenecks from forming in the delivery process.

Keywords Devops, Development and IT Operations, Agile, Lean, Continuous deployment

Acknowledgements

I thank all my interviewees for their availability and open attitude towards my research as well as my friend Edward Ford for opening some doors for me.

Table of Contents

List of Figures

List of Tables

1 Introduction

This section will give the reader a background overview of the topic, as well as the motivation for the research. It will shed light on the existing research gap in Devops research and also define the research questions the thesis will base on.

1.1 Background

In the age of high-availability monolithic IT systems and increasing demands springing from competition and business, there is pressure to develop new software features at an accelerating pace. In addition to rapid development, the systems require zero downtime and error-free operation at any given time. While technological advancement has been rapid, the quality of application development process has been lagging behind (Menzel 2015).

Staying ahead of competition puts an increasing pressure on software developers to produce new features at an increasing speed. New software development practices such as Agile Software Development encourage small, incremental changes which means developers need to deliver code changes frequently - often several times in one day.

On the other hand, operations personnel are in charge of taking these new features and improvements, deploying them into production and keeping them running indefinitely - at the same time making sure there are no system failures or outages caused by these deployments. Operations tend to naturally resist deployments since they know changes to the systems are a major cause of outages. In other words, developers want to release software more frequently; operations professionals want to protect the stability of the infrastructure. (Hussaini 2014).

If and when problems occur, IT Operations might not necessarily have enough knowledge about the inner workings of the deployment to see what is wrong on the other side. Development might not have paid much attention to the performance of the developed feature in an actual production system while designing it. This leads to software performance issues that are overcompensated by expensive hardware that is ultimately never used (Humble & Molesky 2011).

As a result, there exists a "Wall of Confusion" between development and operations personnel which manifests in throwing blame around, much to the irritation of management and the business, which obfuscates their ability to predict when releases are coming out and delivering business value as expected and required (Humble & Molesky 2011). In addition, this results in

stiffer competition, de-motivated teams and excessive time overruns of delivering changes (Hussaini 2014), ultimately leading to rising costs in both application development and deployment. As Patrick Debois (2011), one of the fathers of Devops puts it, "Delivering a project to production still feels like going to war".

Devops is an answer to these challenges. The word is an acronym for "development" + "operations" = "Devops". Devops is an umbrella concept that refers to a wide array of tools and practices to smooth out interaction between development and operations. Despite the term being loosely defined, the ideas behind the concept, however, run much deeper than that. From a slightly different point of view, Devops can also be seen as a collision between a trend of "Agile System Administration" or "Agile Infrastructure" and the aforementioned collaboration between Development and Operations throughout the entire application lifecycle (Agile Admin 2011).

The main idea is to bring down the Wall of Confusion to get developers and operations professionals to work together. What started out as a collaboration effort has manifested into a new kind of work profile: a Devops expert that can handle both the development of a functionality and the deployment of it. (Earnshaw, 2013).

From the business side, Devops is a business strategy that seeks to create an environment where the development and operations lanes are merged and cross-pollinated in order to maximize the outcomes of investments and to ensure that customers continuously get increased service quality and features in a manner that satisfies their needs. Additionally, it aims to reduce risk of IT deployments with tools that maximize predictability, visibility and flexibility. (Capgemini 2015).

Devops is not merely a philosophy but also demands an array of tools to automate processes and helps to facilitate collaborative change and integration. The principles are not entirely new. Many parts of Devops have been around long before the term ‘Devops’ was coined. (Eficode). The main components of this toolkit are release management and deployment tools (Azoff, 2011). Since Devops is not standardized to any given toolset, individual setups might vary considerably across organizations.

1.2 Motivation

This section outlines why it is useful to study Devops.

Devops is a relatively new term that first appeared in 2009. Many people argue that while the term is new, the concepts have been around for quite some time. While there is no clear-cut specification or standardization of Devops practices, the emerging philosophy has been noted as one of the rising tech trends in the first half of the decade.

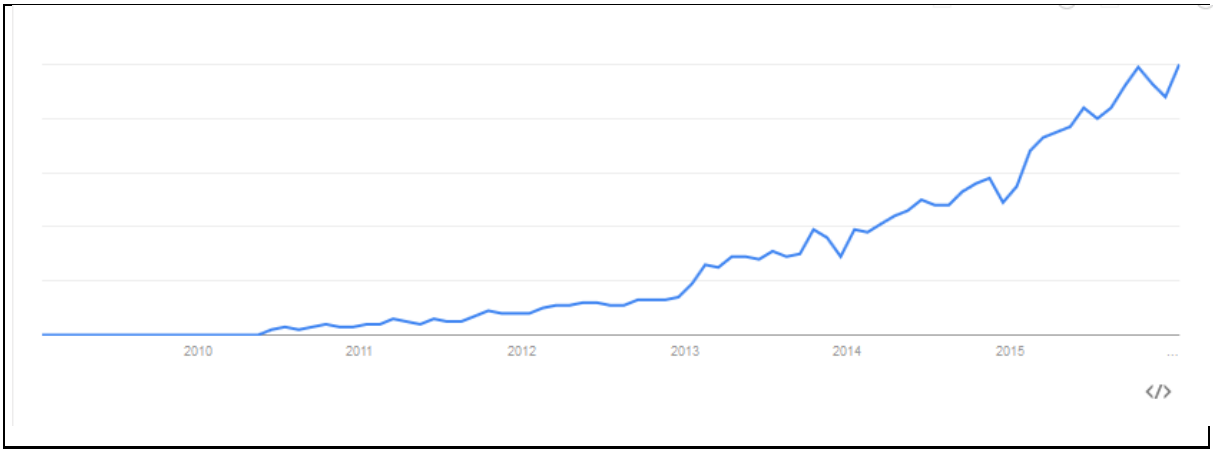

Figure 1: Google Trends graph for Devops.

On the 2015 Gartner Hype Cycle of Enterprise Architecture, Devops appears to be right on the top of the curve. The Cycle is a map that shows the adoption and expectations of emerging technology trends in enterprises. It is divided into five steps: Innovation Trigger, Peak of Inflated Expectations, Trough of Disillusionment, Slope of Enlightenment and Plateau of Productivity. In essence, emerging trends go through the hype cycle, gaining momentum and expectations as they go up the curve and hitting the peak point where it is considered a silver bullet. Investments are made. Next, when investments are not showing nearly as much return as expected, the trend comes down the curve and hits the Trough of Disillusionment. Only after expectations have settled to a realistic level and the organizations have been starting to learn how to apply the trend in their own specific contexts, the trend moves onward towards the real productivity gains. (Gartner 2015).

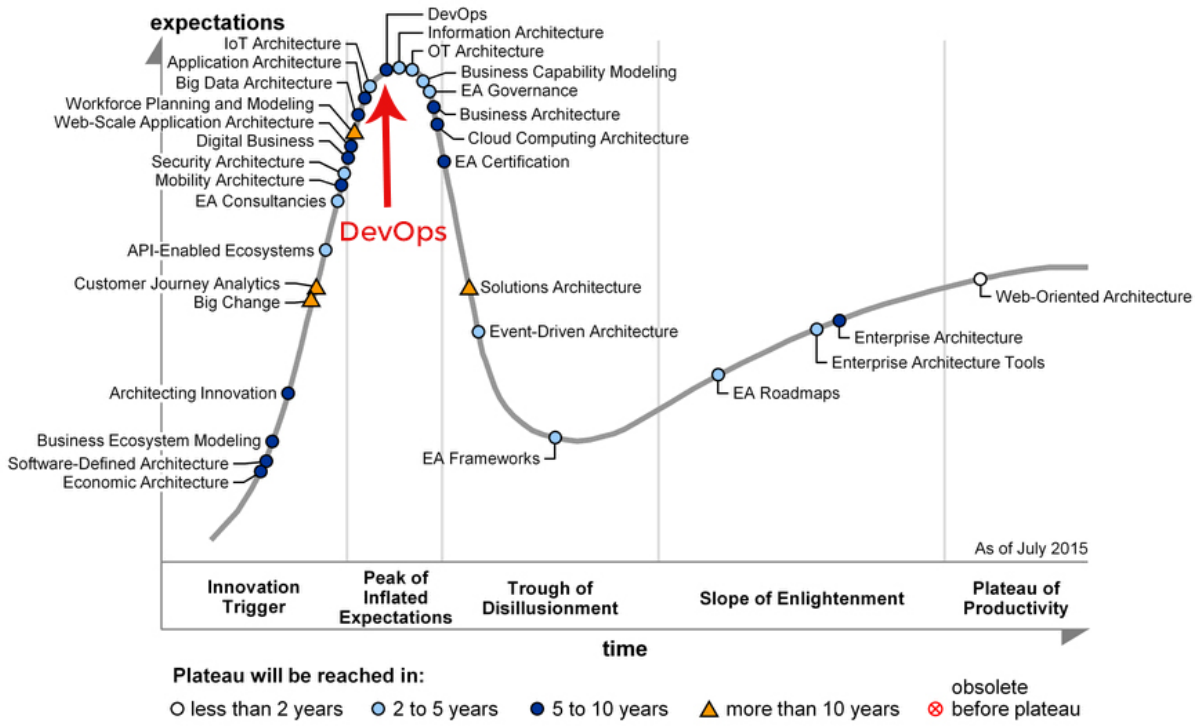

Figure 2: Gartner Hype Cycle of Enterprise Infrastructure.

A recent study by Rackspace (2014) shows 55% of the 700 IT decision makers had already adopted Devops and are looking for enhancement, while 31% of them plan to adopt Devops in the next 2 years. This adoption is among the largest in the side of technology for the initial implementation of tools.

Because Devops is now on the peak of inflated expectations, I find it very important to study the real-world implications and challenges of Devops to perhaps clarify some of the expectations and realities behind the approach.

As stated above, Devops is not only a philosophy. It requires a certain degree of tools and practices to work efficiently. These tools, for example Continuous Deployment and Release Management and Configuration Automation tools, are also evolving because of Devops. The rapid evolution creates a remarkable market for these tools and subsequently a lucrative market to enter as an enterprise application developer. Many of these tools have existed as a part of the regular toolset of IT development and operations already before Devops came about but the emphasis in support of Devops currently has started to transform how these tools are positioned and perceived in the marketplace (Azoff, 2011).

There are many benefits in embracing Devops in an organization. According to a CA Technologies (2014) research, organizations were able to reduce their development and maintenance costs by 18% after shifting to a Devops-style development lifecycle. Additionally, Devops was found to be the key reason behind increases in revenue and number of new clients. This translates to a 20% increase in business.

A State of Devops report (2013) recently quoted that failure rates have been cut down by half in their surveyed organizations. The same source also reported considerably faster release times: surveyed organizations were releasing code 30 times more frequently and 8000 times faster than before shifting to Devops.

In light of these figures it can be clearly seen that there exists real value in managing the development lifecycle in this manner.

1.3 Research Gap

There has been relatively little academic research on Devops considering the amount of Devops-related investments. Current literature is mainly based on describing the phenomenon and sharing best practices that are based on more or less subjective experiences of the authors. Moreover, the method is maintained by the community and can best be characterized as a set of best practices. The academic articles on the subject are extremely scarce, and the field lacks a common framework and even a quotable definition. Research circles around trying to describe the phenomenon. The solutions can be viewed through different perspectives and therefore can result in different implementations. According to a survey by Eficode, 66% of Finnish IT decision-makers see Devops as an interesting project for their company.

In light of these motivations, the key questions of this research are

1. *What are the key components of Devops?*
2. *What are the main challenges organizations face with Devops?*

Additionally, the research tries to answer the following sub-question:

3. *What are the ways to overcome these challenges?*

2 Overview of Development and Operations

This chapter's purpose is to give an overview of what is meant by Development and IT Operations respectively. The traditional functions of each area is described in order to gain understanding of how the functions change when shifting to Devops.

2.1 Software Development Lifecycle

A software development lifecycle (SDLC) adheres to important phases that are essential for developers, such as planning, analysis, design and implementation. Since software has become a key component in almost all aspects in life, the associated risks have grown considerably. To mitigate this risk and to structure the development process in a manner that it produces only anticipated outcomes, a number of software development models have been created: Waterfall, Spiral, V-model, Rapid prototyping, Incremental, Stabilize and Agile.

The Waterfall model is the oldest of the group, while Agile software development presents the modern trend of development. Devops can be seen as the most recent addition in a continuum presented in Figure 3. Some call Devops the 3rd generation software development method, a continuation from the 2nd generation Agile methods (Eficode).

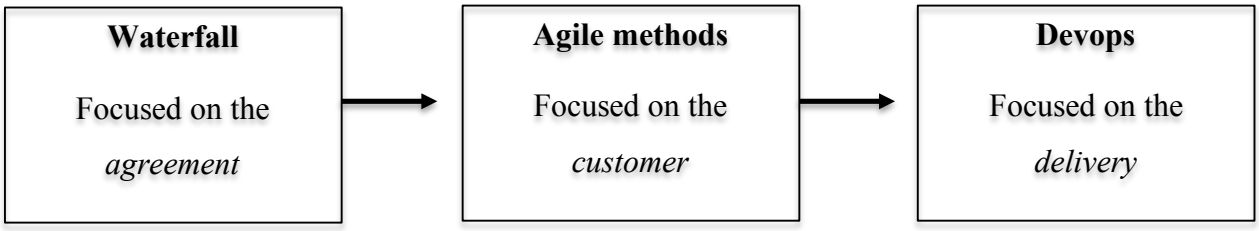

Figure 3: Devops: 3rd generation software development method. (Based on Eficode)

The next sections will focus on each end of the Waterfall – Agile spectrum. Starting with the traditional Waterfall model, the aim is to familiarize the reader with the rigid development processes that have been problematic for some IT organizations even nowadays. Next, Lean and Agile are discussed which are especially relevant from a Devops point of view. "Some consider Devops picking up where 'traditional agile' left off. After all, software doesn't bring value unless it is deployed in production; otherwise it's just inventory." (Debois 2011)

2.1.1 Traditional Software Development: Waterfall model

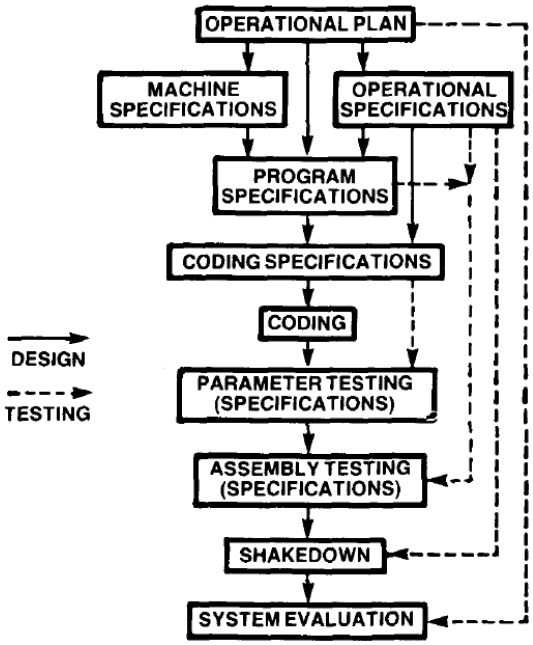

Figure 4: 9 sequential stages of software development by Benington (1956)

The Waterfall model is a sequential design process and is recognized as the oldest standardized way of doing software development. The first ideas were presented by Benington (1956) who identified 9 stages for software development that can be seen in Figure 4.

The idea behind the model is that a carefully planned and documented application results in a higher quality application that is easier to code than an application that has constantly changing specifications and requirements. This tedious "code-it, fix-it, code-it-some-more-until-it's-quickly-not-maintainable" pattern was not acceptable. (Leffingwell 2007). Each of these stages are meant to be executed sequentially in contrast to a degree of parallelism present in modern development. Benington (1956) gives an example where the first 4 specification steps take 53% of engineering manpower, actual coding takes only 7% and testing takes the rest. The actual Waterfall model is usually credited to Winston Royce (1970), rather ironically, as he stated that "it has never worked on large software development efforts." Surprisingly, the model was widely accepted to use.

The Waterfall model is very strict about project scope. Once the specifications are made, the model is prohibitive about changing anything. As a result, the model is poor at responding to change. The foundation of the project is based on a fixed scope, and the costs and schedule are estimated and fixed thereafter. This results in what is called an "Iron Triangle Trap". (Leffingwell 2007). Since the scope cannot change, the software either needs to be delivered

in poor quality to meet the fixed budget and schedule or finally when either schedule and/or costs gets out of hand.

The sequential nature of Waterfall also means that the application has to be completed in terms of code before even a single test can be run (Hin 2006). In a complex application that has a lot of dependencies, tracing errors in this manner could prove to be very difficult.

2.1.2 Lean Software Development

Over the years the problems with the Waterfall model were realized and practitioners started to look for alternative ideas and methods from elsewhere.

The Lean way of thinking evolved from trends that started in manufacturing, such as Toyota Production System and Total Quality Management. Lean methodologies are designed to reduce waste and improve operational efficiency. Specifically, eliminating repetitive tasks that have no added value in the value chain became a goal. The Lean mindset can be applied to Software Development and IT Operations and is one of the key components in the CALMS model for Devops discussed later. It has also played a major part in the theoretical groundwork of Agile Software Development discussed in the next section.

Mary and Tom Poppendieck (2003) have formulated a set of principles for the application of Lean thinking into software development. Eliminating waste is the first principle. Waste is anything that does not add value to a product. Furthermore, the value is always perceived from the customer point-of-view. Waste might mean excess documentation, excess features that are not explicitly asked by a customer, and handing off development from one group to another. The key is delivering exactly what customers want. Excess development time is also considered waste. The second and third principles are Amplifying learning and Deciding as late as possible. In a software development context, it means iterative cycles of development and making definitive design decisions only when it is necessary. Learning is amplified by constant assessment and improvement. Deciding as late as possible is an options-based approach and is effective especially in situations that involve a level of uncertainty. In a dynamic market, better decisions are made based on facts rather than assumptions. In software terms, systems should include capabilities for change. (Poppendieck & Poppendieck, 2003).

Deliver as fast as possible is the fourth principle. Traditionally, a mistake-free approach has been favored over rapid development. However, Womack et al. (1990) argue that speed is not a cost factor. Swift delivery has a faster feedback loop and is better equipped to improve the product. This is one of the key components of Devops.

The fifth and sixth principles are Empowering the team and Building integrity in. Technical decisions should involve the people that are responsible for the execution of the plans. Because decisions are deferred to the last possible minute, there is no time for higher authorities to orchestrate the activities of workers. Integrity in software translates to reliable architecture, high usability and fitness for purpose, among maintainability, adaptability and extensibility. (Poppendieck & Poppendieck, 2003).

The final principle is seeing the whole. Integrity in complex systems requires a deep expertise in many diverse areas. From a Devops perspective, Development and Operations traditionally have siloed thinking that leads them to optimize only their respective ends of the development cycle rather than overall performance. This results in suboptimal end product or service.

Lean software development principles have been used to lay the groundwork for frameworks that are more agile for software development. Agile development can respond to changing requirements faster than the traditional models.

2.1.3 Agile Software Development

Agile Software Development springs from the so-called "Agile methods" that are a reaction to traditional ways of developing software: Waterfall, for instance. In response to heavyweight, documentation-driven development processes, Agile methods have increased in popularity since the beginning of the new millennium (Beck et al., 2001). Developers found that designing a software from a complete set "locked-in" requirements and documentation lead to frustration (Highsmith, 2002). According to Highsmith (2013) requirements "change at rates that swamp traditional methods". Customers and business owners were unable to state the full requirements of a system that was only going to be in production 2-3 years after the tedious and often lengthy process of "locking" the requirements down. This lead to people coming up with their own methods to respond to the traditional way of doing things.

Many of the ideas of Agile software development were already conceived as early as the 70's: Agile methods are partly based on Iterative Enhancement that was introduced by Basili and Turner (1975). Before Agile, the Waterfall model had evolved into Incremental techniques that broke the Waterfall model down into smaller pieces, namely allowing several functionalities to be worked on in parallel across the project. Incremental techniques brought some reductions to the development time but requirements were still frozen during the whole development cycle (Boehm, 1988). Iterative development and the subsequent Spiral Model also started to

introduce the risk management and changing requirements to the development process but failed to cut down on planning time and heavy documentation.

Finally, the capability of quick change was recognized. Schwaber, the developer of Scrum, realized that rather than stressing predictability, development needs to accept change in order to be Agile. Also, stressing the importance of creativity in managing complex software development problems rather than heavy documentation (Highsmith & Cockburn, 2001) was necessary and stressing techniques that would allow responding to changes as quickly as they arose (Turk et al., 2002). These ideas, coupled with Lean Manufacturing discussed earlier eventually lead to the Agile Manifesto.

The Agile Manifesto is an agreement made by seventeen representatives of the alternative software development movement that came together in 2001 to discuss new ways of developing software. The manifesto concentrates on individualism and interaction, working software over extensive documentation in addition to collaboration and responsive change (Beck et al., 2001). The emphasis on individualism was based on the fact that development was too caught up in its emphasis on the process. The manifesto embraces working software since it was the final product rather than large volumes of documentation that were never maintained. Collaboration is valued over contract negotiation since business value is only delivered after the project has been started (Abrahamsson et al. 2002). Finally, the Agile Manifesto highlights responding to change because "Requirement change was one of the most common causes of project failure" (Glass, 2001). The Agile way prioritizes features constantly in response to changing demands and even released features can be modified.

The aim of Agile in practice is to develop and release application functionalities in small proportional batches instead of releasing the finalized product all at once. According to Ambler (2002), there is no requirement for excessive planning ahead or documentation.

Instead, Agile strives to keep the development as simple and planning as adaptive as possible, focusing on continuous improvement and communication between people. People are the primary source of creativity and value. (Highsmith & Cockburn 2001).

Highsmith & Cockburn (2001) note that software is tested throughout the development cycle (in contrast to a separate testing phase later in the cycle). Customer participation also plays a key part in validating the constantly-released features. Concurrent testing and customer participation keeps the feedback loop tight and allows the teams to be better focused and to notice errors in the software sooner.

Agile software development is a philosophy rather than a standard set of practices. There are several detailed processes that apply Agile methods, the most popular of them being Scrum and Kanban.

The main idea of Scrum is based on sprint-oriented thinking. Sprints are development periods that typically last 1-4 weeks, however no longer than one month. During these sprints the features are coded, tested and integrated into the existing product. Daily sessions or "scrums" are held to track progress and to create a plan for the next 24 hours. The items that Scrum teams work on are arranged into a backlog of items or "stories" that are prioritized according to their perceived value at each point of time. Stories in Scrum refer to descriptions of features how they are perceived by the customer. (Scrum Guide; Mayer 2009).

At the start of the Scrum process, the Product Owner gathers input from customers and other stakeholders. These features are arranged into the product backlog. The Scrum team decides how much work is needed per feature and decides how much work it can do in the next sprint. Once the sprint is started, requirements and items in the sprint cannot be changed. The progress of the sprint is monitored closely with burndown charts and no changes can be introduced during an ongoing sprint. The teams have a special Scrum Master whose job is to facilitate the work of the team and to remove any obstacles in the way of their work. Scrum teams have very low hierarchy by design and the whole team is responsible for delivering the predefined outcome by the end of the sprint which may mean sudden reallocation of tasks at an individual level if its beneficial to the team. (Mayer, 2009). Scrum has been seen to increase productivity, induce higher quality and reduce lead times[1]. (Schwaber, 2004)

Kanban was developed in Japan by Taiichi Ohno after World War II for manufacturing purposes while working at Toyota. The main goals of Kanban were to keep up a high level of production and to continuously improve the process. (Ohno, 1988; Gross & McInnis, 2003). Kanban was first used in a software development process by David Anderson in 2006.

Kanban's goal is incremental change rather than radical change. Respecting the current process and the elements worth preserving in addition to embracing current roles, responsibilities and titles minimizes change resistance. Concurrently, agreeing to strive for continuous improvement constantly changes the process towards a more efficient one but takes a slow and gentle approach. (Anderson, 2010).

[1] The duration between the initiation of development of an item and the completion of it.

Kanban focuses on the flexibility of planning. The development team focuses only on one item at a time: the one that is prioritized first in the product backlog. This ensures that the team is providing maximum value back to the business. The product manager is free to re-prioritize work in the backlog if it is not currently actively worked on. There is no need for fixed-length iterations such as sprints used in Scrum. The key is to concentrate on the constant "flow" of value rather than fixed delivery dates. (Atlassian).

Kanban visualizes the workflow. At the very basic level, items of work are represented by cards. The cards are attached to a board with different columns for different stages of the process (the word kanban stands for "signal board" in Japanese). Typically, these columns are labeled Pending, Analysis, Development, Test, Deploy or something similar. Visualizing work in this manner makes it more tangible and therefore easier to focus on and prioritize (Anderson 2010).

Work-in-progress is limited to a set amount of items at a time. Kanban utilizes the Theory of Constraints where a bottleneck is found and protected with buffers (Goldratt 1990; Goldratt & Cox 2004). In Kanban, the buffers are the artificial card limits per each column. Even though items in other columns upstream might be finished, they cannot "pull in" new work from previous steps before some of their items are pulled forward. This highlights bottlenecks in the process and makes workers at other stages idle, allowing them to help completing the tasks at the bottleneck station. (Kanban blog).

Measuring the outputs of a process before making changes and mirroring the results to the ones measured after the change makes it easier to see how the process is evolving. Keeping the steps small and the feedback loop tight is essential for continuous improvement. It is necessary, however for the whole team to understand the process in the same way before changes can be made, which calls for open communication. (Everyday Kanban). Kanban is known to reduce lead-times and velocity of development, since focus is always crystal clear. Limiting work in progress, according to Little's Law, leads to decreased lead-times, affecting the number of software defects such as bugs in code and therefore improving the quality of the product.

The problems with the Waterfall model and their solutions using Agile development is summarized in Table 1.

Table 1: The problems with the Waterfall model and their Agile Solutions (based on Eficode).

Problem with the waterfall model	**Agile Development Solution**
Software is planned like buildings: before any actual work is done. Changing plans later is expensive or impossible.	Software is developed in small iterations and in cooperation with the customer throughout the project
Porjects often have a fixed agreement with regard to price and requirements. Negotiations are slow and based on guesswork.	The agreement made at the beginning of the project concerns common procedures and a prioritized list of requirements, not the final product.
Software is developed in stages where testing starts after the development is "ready".	Software is developed in small, independent iterations, the contents of which can be altered before an iteration begins.
Visibility into the development process is poor.	The customer sees the actual status of the project at regular intervals.
Changes that occur during the development process are not prepared for.	Changes are welcome and they are prioritized as parts of the future iterations.
The system is only tested at the end of the execution phase when making changes is expensive. Some of the necessary fixes may even be impossible to implement.	The functionality of the system is already tested during the execution phase to ensure that the requirements selected for an iteration may all be done from start to finish during the iteration.

In the next section, traditional IT operations function is described.

2.2 IT Operations

IT Operations is the area of IT responsible for managing the IT infrastructure of an organization. Gartner defines IT operations as "the people and management processes associated with IT service management to deliver the right set of services at the right quality and at competitive costs for customers." (Gartner IT Glossary) Furthermore, IT Operations Management is split into different components by the Information Technology Infrastructure Library (ITIL). According to the ITIL Service Operation, "IT operations management executes the daily operational activities needed to manage IT services and the supporting IT infrastructure" (Steinberg 2011). In other words, IT Operations performs a defined sequence of daily activities in order to keep IT going. A traditional operations role is measured by its ability to provide a reliable, optimized infrastructure. In effect, that means ensuring as little

change as possible in order to guarantee that network resources are available so users can be more productive.

The main areas of IT Operations are console management, job scheduling, backup and restore, print and output and maintenance activities. Console management refers to monitoring the IT infrastructure through various logging tools, reporting engines and IT controls. Job scheduling still plays an important role in IT organizations, where integrations, cleaning operations and other batch jobs have to be triggered manually at scheduled times. Backup and restore remains a key operation even though IT has been moving to central server control and management through various virtualization solutions and backup is therefore easier to perform. (Steinberg 2011). The role of printing and output has been shifting away from IT Operations to various print management solutions maintenance, but there are still plenty of IT Operations organizations that maintain printers and other output devices.

Infrastructure maintenance activities comprises of the two functions Application and Technical management. This means provisioning new servers, migrating them and deploying applications to them. Operations is usually organized to work in shifts, so it can perform assigned after-hours tasks.

Facilities management is in charge of the physical sites where the infrastructure is located: server rooms, data centers and disaster recovery sites. This involves cooling, power, humidity, etc. as well as actual network infrastructure and servers.

Depending on the size of the organization, IT Operations is usually organized in three ways. It can either be a department or a sector in large organizations. In mid-sized IT organizations, it can be a virtual team that consists of also Application and Technical management team members. Finally, in small-size IT organizations, Service Desk operators can perform IT Operations tasks during their normal shifts. More complex tasks are escalated to on-call engineers or technical account managers.

3 What is Devops?

This section will detail Devops at a more in-depth level. First, I will cover the short history of Devops and the main trends that lead to the inception of the philosophy. Then, I will take a more practical approach in describing the methodologies behind Devops. Finally, there is a section about the discovered benefits of the approach. The main goal of this section is to give an overview to support the succeeding research and to answer research question 1: What are the key components of Devops?

3.1 History of Devops

The Devops movement is so widespread that it is surprising to think that it is only a few years old. Devops has formed out of a fundamental need and it is based on a simple yet powerful philosophy: business works best when efforts are coordinated and when they are based on collaboration. As a result, its growth has been rapid.

Before the current evolution of Devops, there was no direct connection between development and operations. One of the early influencers of Devops was a Belgian consultant named Patrick Debois. Debois had assumed several IT roles in large organizations. In 2008, at an Agile Conference in Toronto he brought up the problems associated with the segregation of Development and Operations and the fact that projects were assumed to be always late and the final product underperforming. (Rapaport 2014) This talk lead to the forming of the Agile Systems Administration Group (Paul 2014).

Next year, the first Devops speech was given by Flickr employees titled "10+ Deploys per Day: Dev and Ops Cooperation at Flickr". This helped IT professionals look at the problem from a different point of view and showed the world what can be achieved with a highly transparent and integrated development-operations collaboration. Debois and a few interested professionals were inspired to organize a conference called Devopsdays. Simultaneously the name of the movement was coined to the portmanteau "Devops". Devopsdays soon became a regular global series of community-organized conferences and a major force driving the Devops community forward. Thereafter, the Twitter hashtag #Devops became an essential source of information. (Rapaport 2014).

In 2011, beginning with Vagrant, a tool to create and configure virtual deployment environments, the Devops community started building open-source tools to facilitate the philosophy from its technical requirements.

Two years after, Devops had started to formulate from a set of ideas and buzz to a more coherent, standardized practice. A plethora of seminal literature was released such as "What is Devops?" by Mike Loukides (2013), "Implementing Lean Software Development" (Poppendiek & Poppendiek, 2013) and even a business fiction novel called "The Phoenix Project" (Kim et al., 2013) that brings forward the problems of current IT organizations and the benefits of making the shift to Devops in an engaging and inspiring way.

What was first thought to be a simplistic idea of creating a communications channel between developers and Operations has been expanding to include the whole development lifecycle. This includes "automation from the beginning of the process through the deployment of the solution in a system of incremental builds" (Dodson 2013).

Currently, organizations have started reaping the benefits of this new approach, with several well-known organizations such as Netflix, Amazon and Target (Null 2015).

3.2 Trends Behind the Evolution of Devops

There are several trends that have contributed to and encouraged the rise of the Devops paradigm. In this section I will describe each paradigm in detail.

One of the reasons behind the ever-increasing requirements for software is the consumerization of IT. Consumerization of IT stands for describing the cycle of IT first emerging in the consumer market and then spreading to business or government environments. This is largely due to employees using consumer market technology and devices at home and then bringing them to the workplace. (Webopedia)

The field in consumer IT is changing quickly. For instance, Apple and Google need to keep pushing out new features in all their services, mobile operating systems and hardware. From major yearly updates to minor quarterly ones, consumers have grown accustomed to expect these releases. Bug fixes and other critical improvements are made even more quickly and deployed at once to millions upon millions of devices and software. If the providers are unable to keep up with this pace, consumers are quick to react and switch to competitors. Marketing process is also changing from driving consumer behavior to one responding to it. (Sussna 2013).

In business IT, these consumer expectations are also "consumerized" and they apply to software as well. This also requires the software development and application delivery process to be consumerized as well. However, with enterprise IT, there is a higher level of complexity,

moving parts and integrations involved which prevent lead times from improving naturally. This calls for embracing Devops practices. (Elgan 2013).

Another factor in the evolution of Devops is the evolution of IT infrastructure. IT infrastructure is described as the foundation of shared IT capabilities. The foundation provides the base capabilities for other business systems. These capabilities consist of internal technical expertise such as equipment, software, cabling and managerial expertise that are required to provide reliable services (McKay & Broadway 1989).

Traditionally, IT infrastructure has been viewed as a significant barrier or enabler in the practical options available for planning and changing business processes (Grover et al. 1993). IT infrastructure and related issues have consistently been identified as a key concern of Information Systems Management (Niederman et al. 1991; CSC Index 1994).

Furthermore, Grossman and Packer (1989) argue that IT infrastructure differs from applications in its purposes as a foundation also for future applications in addition to current business functionality, and in the way in which it must cope with the uncertainty of future needs. Therefore, the financing of IT infrastructure has traditionally been different from applications and their benefits have been difficult to quantify, requiring board-level or executive management approval (Weill 1993). According to a research, about 55% of the total IT budget goes towards technology, processes and human resources that comprises IT infrastructure (Weill 1991).

Major IT infrastructure components include hardware platforms, base software platforms, communications technology, client-server technology and other software components, common handling mechanisms for different data types and methods, standards and tools (Turnbull 1991; Darnton & Giacoletto 1992). These capabilities can be provided either in-house by the corporate IT function (Weill & Broadbent 1994) or by an outsourced service provider (PE International 1995).

Finally, providing the IT components as a service to business and application stakeholders requires a human infrastructure that has the knowledge and experience to provide the policies, architectures, planning, design, construction and operations capability necessary for a viable IT infrastructure (McKay & Brockway 1989; Keen 1995). The key elements of IT infrastructure are visualized in Figure 5.

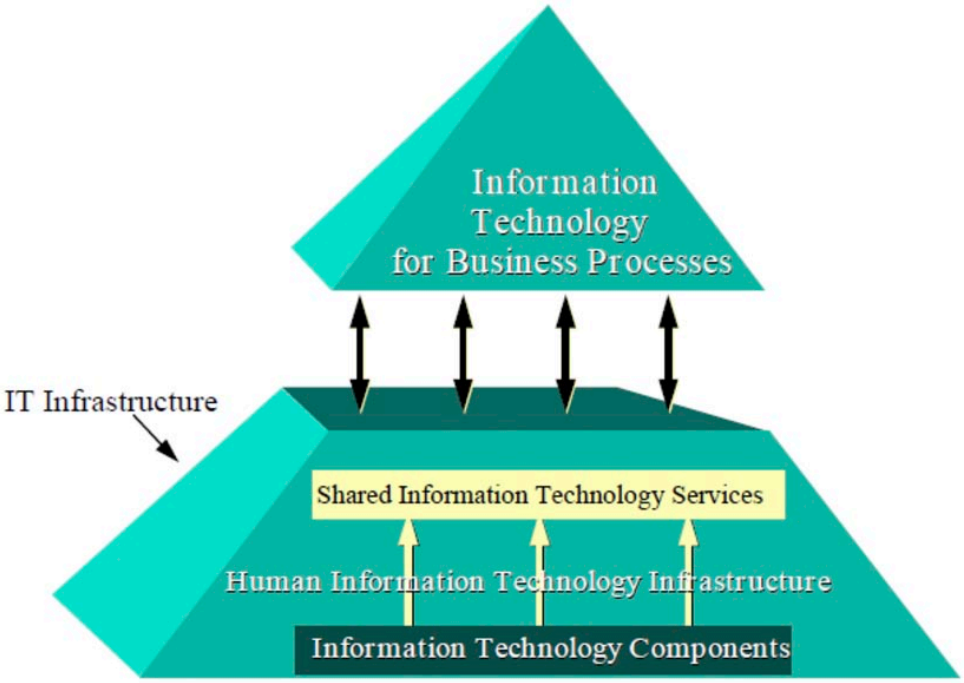

Figure 5: The elements of IT infrastructure (from Broadbent et al., 1996)

In the 21st century, IT infrastructure started to evolve into ubiquitous cloud infrastructure. Cloud computing is not a prerequisite of Devops but it is a major part of most Devops initiatives. "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." (NIST 2011)

Cloud enables servers to be accessed from anywhere over the Internet. Cloud infrastructure is ubiquitous by definition which means many providers introduce multi-location redundancy in case of site failures. As a result, a faster and more reliable service is realized. (Buyya et al., 2009).

When IT infrastructure is provisioned in the cloud, IT operations can provision new servers in a matter of minutes to host an application or a service. It has implications to agility and flexibility since traditionally server hardware needed to be purchased, set up in a physical location and configured separately. To accommodate peak times, additional hardware needed to be procured which resulted in servers that were idle most of the time and it was impossible to use this extra capacity for other workloads. (Cervone 2010).

Cloud computing is also much cheaper than traditional hardware resources. Smaller and newer organizations with simple needs often get the most dramatic and immediate results from public cloud services. They avoid "sunk costs" in existing infrastructure and have fewer internal applications to integrate with cloud platforms.

The emergence of cloud computing is related to service economy. In service economy the nature of software has changed from being customer-operated products to vendor-operated services. Furthermore, payment is based on consumption and utility rather than one-off purchases. (Buyya et al., 2009).

The main services cloud computing can offer are Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). They differ in the amount of abstraction they offer. Infrastructure as Service, such as Amazon AWS, provides servers that can be configured in the same manner as traditional server hardware. Platform as a Service, such as Heroku, provides a platform to run an application on. There are usually several PaaS applications running on a PaaS server which is configured and managed by the service provider in its entirety. The configuration of servers is therefore abstracted away from the customer. Finally, in the other end of the spectrum lies Software as a Service that abstracts away also the application platform. Sussna (2013) concludes that cloud vendors are playing an increasing

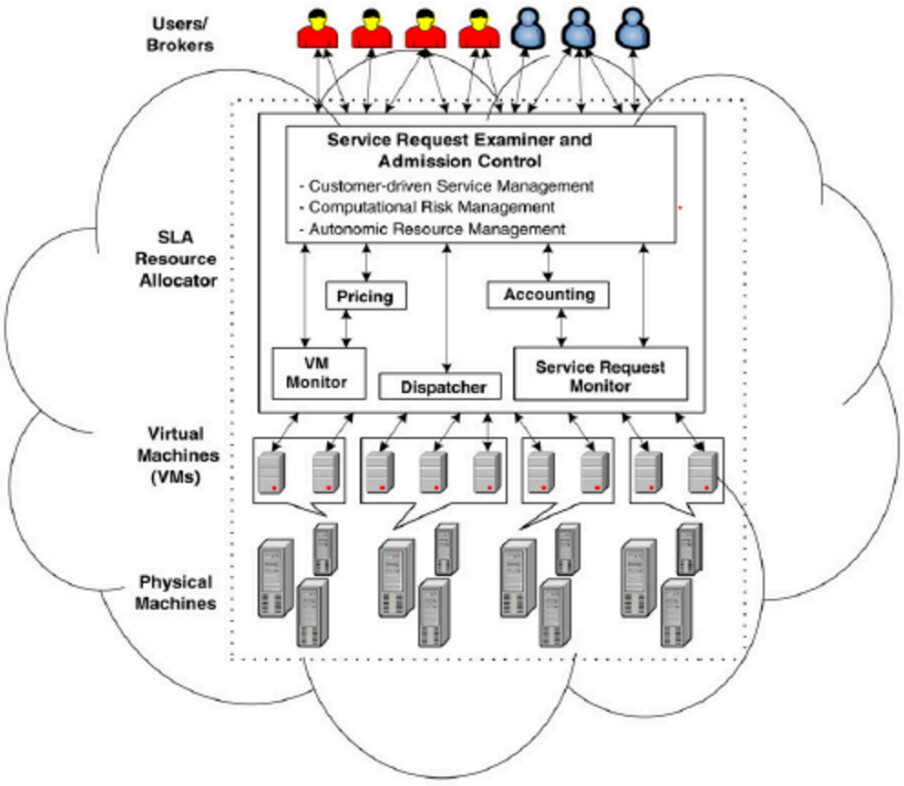

Figure 6: The elements of Cloud Services (from Buyya et al., 2011)

role within Enterprise IT, freeing up internal IT to focus on value added services. Figure 6 exhibits the elements of cloud services.

A major component of cloud computing is server virtualization. Rather than offering a physical server for each deployment, the resources of one physical machine can be split into multiple "virtual server" operating systems running simultaneously on the host. Subsequently, economies of scale are realized in addition to better utilization of resources. Services seldom require even the most minimum amount of resources that a physical machine purchased in present day can provide.

In addition to better utilization of resources, virtual servers are "sandboxed", meaning that their configuration has no effect on other virtual servers running on that instance or even the host, for that matter. Virtual server images can also be migrated to other physical instances in order to scale up, prevent failure if the physical machine goes down or to ensure continuous deployment. The physicality of virtual servers is diminished. As a result, no one needs to know the exact location of the machine running the software any more. (Cervone 2010).

Virtual servers are a key component in the automated deployment process used in Devops. Changes to a system pass through a pre-defined deployment pipeline involving a complex framework of different tools ranging from version control [2] through automated testing frameworks to configuration management tools. (Humble & Molesky, 2011). The idea behind deployment automation is to reduce user errors in each of the phases and to increase deployment speed and visibility. Everyone who has access to the pipeline can immediately see which step a release is currently in. The feedback loop is very tight: the developer is immediately notified when the deployment passes through each phase or if there are problems with the build. The automated deployment pipeline is shown in Figure 7.

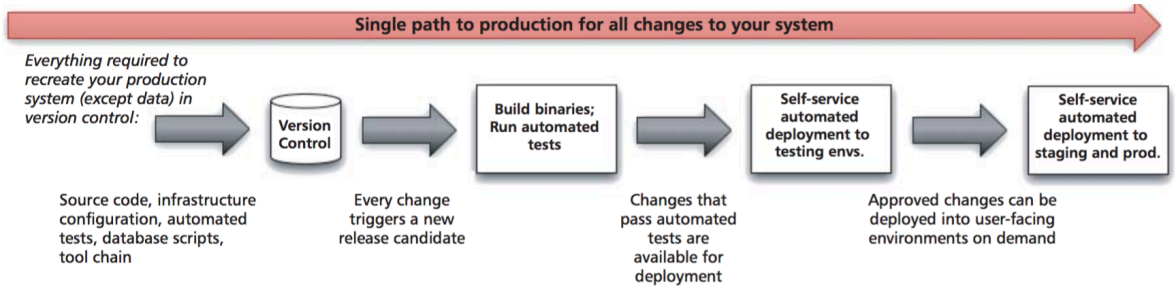

Figure 7: The automated deployment pipeline (from Humble & Molesky, 2011)

[2] Version control, eg. Git, is a ”system that records changes to a file or set of files over time so that you can recall specific versions later”. (Chacon & Straub, 2014)

Automated testing is an essential component of deployment automation and releasing quality software in rapid increments. Test automation reduces the need for manual testing and speeds up the testing process significantly. Additionally, it ensures testing is carried out exactly the same way every time, minimizing the risk of user errors and oversights.

Another common component in many automated deployment pipelines is Infrastructure as Code. Infrastructure as Code (IaC) is a method for managing IT infrastructure following the same principles as software development. This involves version control, testing and incremental change.

Infrastructure as Code automates the provision and configuration of application servers and middleware using pre-defined automation logic. IaC automations are expected to be repeatable by design, so that the infrastructure can be brought up to a desired state from any arbitrary state. This process is facilitated by IaC tools such as Chef and Puppet.

The key notion of IaC is idempotence. Idempotence means that tasks can be executed multiple times always yielding the same results. IaC removes human error and makes sure that all machines in an infrastructure are configured how they are intended and all scripts are executed in the same exact order, every time. (Hummer et al., 2013). This differs from infrastructure automation which just involves replicating steps on multiple servers.

IaC is easy to write and debug: similar environments can be brought up even on the laptop of the developer. Version control allows tracking changes to configuration over time with the possibility to roll back quickly to the last working configuration. The are caveats present in IaC as well. It takes an extended amount of time to plan the configuration. Additionally, bad configurations are duplicated everywhere once they are executed. Finally, there is a requirement that all configurations made to an environment must be present on the configuration template, otherwise the environments will differ and produce unexpected results. (Sitakange, 2014).

3.3 Challenges that Devops attempts to overcome

Devops came into existence because of certain problems in the traditional way of managing development and operations. This section will give an overview of the main challenges that sparked the evolution of Devops.

First, I will describe problems with traditional IT project settings. In an IT-project it is not unheard of to have an individual or individuals who fail to document and share their code and decisions with the rest of the team. In doing so, this person creates a hero cult and this "hero" is the only one who knows how to make a software work and to apply changes to it. Emphasizing titles is problematic, since titles usually correspond to team members' duration working for a company. Seniority does not correlate with development skills, let alone interpersonal skills. Shadow responsibilities, such as tending to pet projects and other ambiguous areas, manifest as a mismatch between role descriptions and what is actually done resulting in problems with planning and coordination. Finally, favoring plan over planning isolates the intended results from the project goals. Alterations to the original plan are rarely made or even authorized since the goal has changed to following the plan rather than delivering value-adding software in a timely manner. This holds especially true in the Waterfall model discussed earlier. (Hütterman 2012).

Organizational and cultural barriers are one of the most definitive challenges Devops hopes to address. Having development and operations siloed in their respective teams or "some loosely coupled working groups" is a major impediment in achieving a unified goal of releasing value-added software quickly. In addition to separate teams, dev and ops might not share a common language in communicating their ideas and challenges. This leads to misunderstandings, confusion and extra work on either sides as well as on the business side. The final barrier is fear: negative attitudes toward sharing knowledge and a common goal coincides with old habits and fear of losing power, reputation or influence. (Hütterman 2012).

Several researchers mention the misalignment of goals and incentives as one of the predicaments of Devops (Hütterman 2012; Humble & Molesky 2011; Hussaini 2014). While Agile methodologies have done a lot to increase collaboration and transparency within the development group and quality assurance, operations are traditionally still left out of the equation.

Agile teams have goals of delivering an increasing number of features at an increasing speed. Business depends on them to respond to fast changing business and technology needs. Because of this intimate relationship, Development is often expected to bring in as much change as possible. (Hussaini 2014).

On the other hand, Operations is tasked with delivering these features into production with additional nonfunctional requirements of service stability. These requirements often conflict with delivery requirements (Hütterman 2012).

Stability is defined as "a resilient system that keeps processing transactions, even if transient impulses (rapid shocks to the system), persistent stresses (force applied to the system over an extended period), or component failures disrupt normal processing (Nygard, 2007).

Traditionally, it has been the responsibility of Operations to ensure software availability in production. Its success is often measured with metrics that calculate server uptimes, software availability, security measures, capacity and response times. Together these attribute to Service Level Agreements (SLA's) that express the users' expectations that all of the software features are fully available. (Hütterman 2012).

It is established that changes to production systems are the main cause of system failures (Humble & Molesky 2011; Hussaini 2014). When downtime is experienced, operations are blamed for the outage directly, resulting in operations shying away from changes to the production system. The main goal of operations is stable and working software with high availability. Therefore, SLAs and delivery requirements have traditionally had a tendency to conflict (Hütterman 2012).

Another issue described in this context is described as the "Blame game" (Hütterman 2012), or "A Toxic Blame Storm" (Turnbull 2010), a mistrust and juxtaposition between development and operations. The conflicts are often due to time pressures and they manifest in several ways.

A common scenario goes accordingly: developers hand in a release that worked well in an isolated test environment, however failing to run in production. Operations blame developers for bad design and developers blame operations for poor implementation and out-of-date infrastructure (Turnbull 2010). This cycle continues until the root cause is found out to be a mismatch between configurations or some minor detail that neither party knew about (Hütterman 2012).

The reason behind this conflict lies in the fact that developers never have to run the systems they build. Therefore, they do not understand the tradeoffs in scalability, reliability, performance and quality. Due to a lack of influence over the design of the application, operations tend to overcompensate the aforementioned by acquiring more expensive hardware that ultimately is never used. (Humble & Molesky 2011).

As mentioned before, Agile Software Development embraces frequent and small deployments. Furthermore, Agile promotes collaboration but this does not normally extend to the Operations people. Frequent deployment puts pressure on the Operations side, widening the gap even more (Debois 2011). The advantages of Agile processes are often "nullified because of the obstacles to collaboration, processes, and tools that are built up in front of operations". Software is released only in test environments and the releases queuing to production grows. Operations are reluctant to deploy in such a pace, often referring to previous bad experiences of hasty deployment. There is often a heavy mismatch between completed functions and planned release dates. This mismatch is also influenced by horizontal optimization where the operations have a higher priority in optimizing the infrastructure than delivering releases more frequently. (Hütterman 2012).

The lack of contemporary feedback is also an issue. One of the principles of Lean Software Development is eliminating waste. It is a common notion that features that are not directly needed to get the customers' current work done *should not be implemented* (Poppendieck & Poppendieck 2007). Subsequently, the knowledge of whether or not to implement a feature needs a working feedback loop between development and the customer. Humble and Molesky (2011) argue that there is a lack of contemporary feedback to base the feature development decisions on - introducing decisions based on out-of-date or incomplete data. This lack of information is a result of long lead times associated with non-Devops organizations.

Finally, the overwhelming complexity of systems makes it difficult to determine what systems are redundant. Decommissioning a system that still plays a part in the inter-related and fragile network of systems can have catastrophic consequences. Therefore, to play it safe, systems are kept running thus increasing the cost of maintaining systems. (Humble & Molesky 2011).

Table 2 summarizes the problems with agile development and exhibits a Devops solution to them.

Table 2: Problems with Agile development and their Devops solution (based on Eficode)

Problem with agile development	**Devops solution**
Delivery of new features to the customer is often delayed.	Devops tools are used to test and release new features as they are completed.
Completed software components are not compatible with each other.	Open interfaces and test automation make it possible to divide development into independent yet compatible parts.

Quality of the product is not ensured properly prior to release.	Devops tools and practices help automating quality assurance and reduce the need for repetitive manual work.
New features break old functions.	The quality of existing functions is ensured quickly and automatically after each change.
Budget goals and deadlines are missed.	The tools and procedures of Devops increase the transparency and predictability of the development work.
Developer teams and IT operations crews are not cooperating.	Developer teams and IT operations crews agree upon responsibilities together. Their goals are unified.

3.4 The key areas of Devops

This section will detail the key areas of Devops. Devops is no longer considered as a trendy subject of those working with the latest development tools. Present thinking goes beyond technology into processes and people, and there are many different approaches to bringing them together (Riley, 2014).

Humble and Molesky (2011) purpose that Devops consists of four dimensions: Culture, Automation, Measurement and Sharing. These four dimensions each target the specific problems discussed in the previous section, the main goal being the alignment of incentives of all stakeholders, particularly Development, Quality Assurance (QA) and Operations personnel. Furthermore, several sources add Lean to the definition as well (Riley, 2014; van Ommeren et al.). Collectively this has been dubbed as the "CALMS" model: Culture, Automation, Lean, Measurement and Sharing respectively. This model defines the most essential points-of-view of Devops. These areas also describe Devops as a "flow" (Riley 2014).

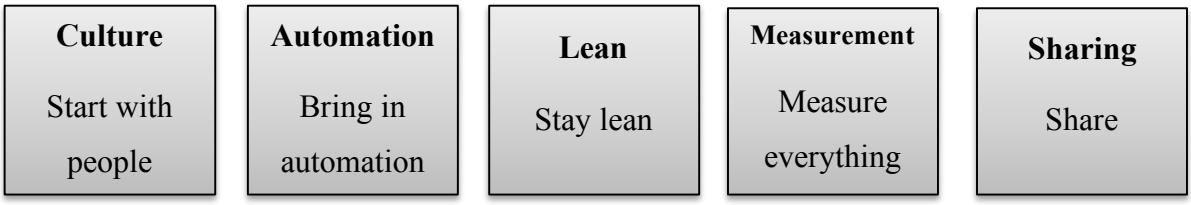

Figure 8: The CALMS model and its flow (based on Riley 2014).

3.4.1 Culture

The adoption of Devops approaches begins from culture. While the tangible parts of Devops largely consist of new processes and tools for continuous deployment and integration,

"Devops" is merely just another buzzword if the organizational culture doesn't embrace all the things that come in the package (Wilsenach 2015).

An essential part of Devops is changing the organizational culture from a collection of silos into an openly collaborative way of working. It entails involving Operations personnel in the design and transition process of an application. Furthermore, they should attend necessary planning meetings, retrospectives and showcases of project teams in order to share their insights and knowledge already early on in the process. Humble and Molesky (2011) note that rotating through operations teams is necessary for developers and they should be equally available for root cause analysis and remedy in case of incidents in production.

Shifting from project- and responsibility-oriented teams into product teams is suggested by Humble and Molesky (2011). In essence, it means that teams are responsible for the entire life cycle of an application or service. The development team easily becomes disinterested in the operation and maintenance of a system if they employ a strict handover process to another team that will look after the system. Rather than having development that throws a deployable package "over the wall" to the operations team, operating the software becomes the duty of the team (Wilsenach 2015). Looking after the system becomes the product team's pain. Doing so forces the team to take interest in optimizing the software code to run as smoothly as possible in production, as well as streamlining the testing and deployment phases of the product through automation and improved monitoring mechanisms for instance.

The product team might consist of developers, operators and quality assurance personnel. The often-used dogma "You build it, you run it" applies especially in this context. This involves building quality into the product via automated tests. In addition to shifting the composition of talent in teams the responsibilities also change. The traditional operations team no longer carries the responsibility of keeping all things running: rather it merely provides the necessary infrastructure solutions for teams to run their software on. (Humble 2012).

Co-locating development and operations staff will help them to work together. Handovers and sign-off processes discourage sharing between individuals and also contributes to the "blame game" mentioned earlier. In contrast, viewing system successes (and failures) as a team effort effectively reduces the culture of blame. (Wilsenach 2015).

Devops culture blurs the line between the roles of developers and operations staff. In some cases, the blurring eliminates the distinction altogether. The extreme manifestation of this

paradigm is “NoOps”, a model where the developer role has taken over all operations responsibilities through effective automatization and IaC or PaaS (CenturyLink 2013).

A common anti-pattern is the introduction of a Devops role or a separate “Devops team”. This kind of approach has an adverse effect on the silos Devops aims to break down as well as the spreading of culture and practices inside the organization. (Humble 2012).

While development and operations talent consolidate to form product teams and new tools are introduced to the workflow of these teams, the need for control is also diminished. In order for the team to perform effectively, it has to be able to make their own decisions without the extra overhead of a convoluted change management process that is common in large enterprises that promote ITIL for instance. Shifting to autonomous teams involves building trust between managers and team executives as well as changing the way risk is managed. (Wilsenach 2015). New tools help facilitate these changes. Introducing version control to infrastructure, for instance, increases transparency and auditability in changes. There is no need to associate changes to infrastructure to change management tickets anymore since every change and the accountable individual is visible in version control.

3.4.2 Automation

In order to build the new kind of culture into an organization, processes need to change. According to Humble & Molesky (2011), it is made possible by various Devops tools and Infrastructure as Code covered in the previous section. The goal behind the automation practices in Devops lies in achieving low lead times and rapid feedback. This implies using a deployment pipeline that covers all changes that are to be made to a system by any team. As mentioned earlier, the pipeline models the entire process of developing, building, testing and deploying a system.

This "single path to production" is not just application code, rather it encompasses everything including the infrastructure (as code), different environments, database schemas, reference data and configuration. The pipeline acts as one of the gatekeepers into production - most of the time there is a wide coverage of automated tests that are executed against each change to catch any potential bugs, errors or conflicts in the change. Once the change passes all tests it moves onward to the next stage in the deployment pipeline. When the fitness-for-release has been established, actual deployment is a just a button-push away if automated provision of different environments is enabled. The upside of automated provision is clear: any environment (at any

point of time) can be recreated should emergency rollback be necessary. (Humble & Molesky 2011).

3.4.3 Lean

As described earlier, Lean methodologies are a major building block of Devops. Limiting Work-in-progress (WIP) and cutting down batch sizes is handled through Agile Development: no extra work is done unless it is included in the sprint or if the Kanban columns are limited. Furthermore, automation reduces waste efficiently and adds respect of people since no manual repetitive tasks need to be done any more, making work more enjoyable. Test automation builds quality in, delivers fast and optimizes the whole.

3.4.4 Measurement

Humble and Molesky (2011) define measurement in Devops as "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 metrics in the delivery process such as lead time and the effect of releases on system stability. Measurement affects the way people work and visualization of these metrics is important so people have a clear view of how well the team is doing at any given point of time. Metrics also have an influence on discovering the bottlenecks in the process – an important step in keeping the process lean.

Another point of view to measurement is application monitoring in production. Traditionally in Waterfall-type of delivery models, application monitoring is only built downstream at the end of the development process, a step that is performed by operations to ensure application stability. A crucial approach in Devops is to build monitoring capabilities in parallel with the application – this is a requirement of the "you build it, you run it" –type of thinking. This might entail building health endpoints for easy status queries of a service etc. As applications are broken up to even smaller units (i.e. microservices), built-in monitoring mechanisms that can be automatically triggered play an ever-increasing role in measuring application stability. (Ursin 2016).

3.4.5 Sharing

The final component of the CALMS model is Sharing. A transparent culture with effective mechanisms of spreading knowledge across teams is essential in bringing down the walls between Development and Operations. Humble and Molesky (2011) mention celebrating successful releases together and sharing tools and techniques as effective sharing mechanisms.

Co-location, and face-to-face meetings with other teams further enhance social aspect of Devops.

Finally, giving developers time to experiment with new things fuels the discovery process and encourages people to share these findings with the rest of the organization benefiting the organization as a whole (Puppet Labs, 2015).

Like most components of the CALMS-model, sharing is also facilitated with new collaboration tools. In addition to improved auditability, introducing version control to everything and sharing the code repositories across teams helps keeping all teams on track in terms of what techniques and technologies are being used, what code commits are made etc.

Patrick Debois, (2011) one of the "fathers" of Devops, also notes that shared workflow is a key element. When an organization has a deployment pipeline that is visible to everybody, all team members know what deployments are in which stages in the workflow. In other tools chat, wikis and newsletters are also mentioned.

Already mentioned in the Culture section of CALMS, shared responsibility is a key element in motivating people to break down silos. Shifting operational responsibility to the developer side and reorganizing teams into product teams fosters collaboration. Previously, the pains of deployment were experienced solely by operations. In Devops, the pain is shared by the whole team and it is in everyone's interest to make deployments as painless as possible. Instead of just having operations on call in case of emergencies, developers also carry pagers and are alerted if something goes wrong (Debois 2011). Having a shared monitoring system for all application instances further increases transparency and encourages other teams to review what is going on in the infrastructure as a whole.

Finally, the introduction of Infrastructure as Code makes it possible to share existing setups with all teams. Having all teams commit to shared practices, unified operating systems and languages makes life easier for everyone. Using Docker as a tool allows the further unification of systems so all server instances can have the same setup without limiting the choice of technology when developing an application.

3.5 Benefits of Devops

There are clear benefits that can be drawn from the different components of Devops. This section will give an outline of the most important aspects of Devops benefits.

The main benefit of Devops is reduced lead time and enhanced deployment frequency. Lead time is defined as "Time required for changes to go from "code committed" to code successfully running in production." Additionally, deployment frequency stands for "How frequently the organization deploys code." (Puppet Labs 2015). Applying Lean methodologies to cut down batch sizes and the use of automatization in the delivery process are key contributors in these metrics. According to the 2015 State of Devops report (Puppet Labs 2015), high-performing IT organizations have 200 times faster lead time than low-performing organizations. Also, by minimizing the bottlenecks and reducing the pains of deployment, high performing organizations deploy 30 times more frequently than low-performing ones. In this context, high-performing organizations use Devops approaches.

More deploys means faster time-to-market and an enhanced feedback loop. With an enhanced feedback loop, organizations have a better idea of customer reactions to features and consequently, better means of continuous improvement.

Another benefit of Devops is an enhanced change success rate. With a well set-up continuous development pipeline in place, organizations have seen a reduction of pains in the deployment phase. Change success rate is defined as the percentage of changes that succeed when rolled out into production. The 2015 State of Devops research states high-performing organizations have 60 times higher rate of success in deployments than low-performing ones. (Puppet Labs 2015).

The key to realizing this benefit is the use of version control, infrastructure as code, automated testing and common build mechanisms for different environments to prevent errors in code and differences in configuration. A high change success rate attributes to employee satisfaction as well as a lower level of burnouts in staff when unplanned, meaningless work and related stress decreases.

Devops also contributes to stability enhancements. Stability in this context stands for error-free operation of a system and the ability to keep processing requests. Enhanced stability is realized by building quality into the system by utilizing automated tests, building applications with testability and deployability in mind and creating a culture for continuous improvement.

Stability is often measured with availability. As an example of enhanced stability, Fonecta's systems went from 95 % availability to 99.95% availability after adopting Devops approaches (Halkosaari 2016).

Another metric that is used to quantify performance is the Mean Time to Recover (MTTR). It stands for the time needed to recover from a service incident. With version control and automated deployment in place, rolling back to previous versions is easier as mentioned before. High-performing organizations using Devops can have 168 times faster recovery time than low-performing organizations that do not (Puppet Labs, 2015).

Finally, Devops reduces costs. A survey by CA Technologies (2013) found that 46% of respondents had seen a reduction in spend on Development and Operations, with 45% of respondents expecting to see a reduction in the near future. On a side note, only 17% of the survey regarded cost reduction as a driver for Devops adoption.

A path diagram of Devops and its benefits is shown in Figure 9.

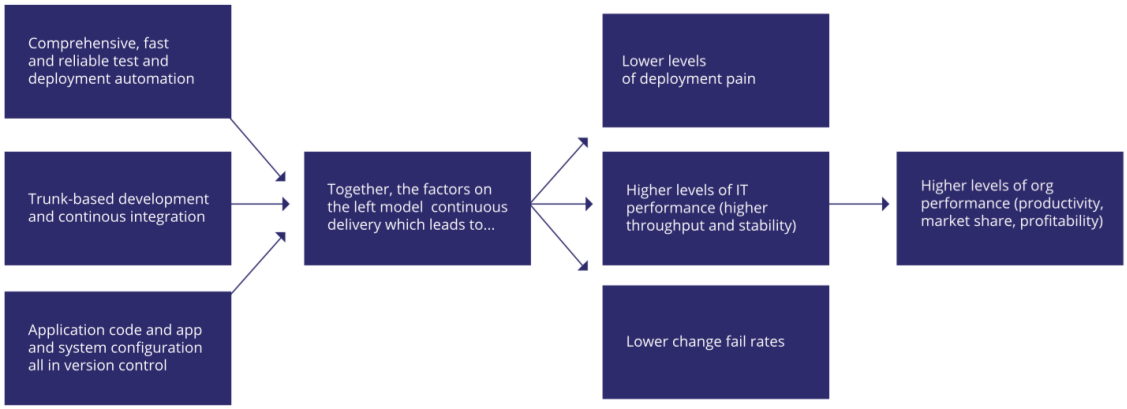

Figure 9: Path diagram showing relationships between Devops and its benefits (from: Eficode)

4 Research Design

The chapter will detail how the research is designed. First, the overall research strategy and approaches taken in the study are explained. The second part describes how data collection was carried out and the reasons for selecting the approach. Finally, in the last subsection the quality of the research is discussed.

4.1 Research Strategy

This research is an empirical study of a relatively new approach to software development and delivery in the field of IT. Uusitalo (1991) states that an empirical study has to focus on a real world phenomenon and information needs to be acquired with a systematic method. Due to a scarce amount of prior research in the field, the empirical data collection for this thesis was carried out by a systemic qualitative method. The thesis consists of rich insights to the topic with an aim to describe and explain the phenomenon. This supports the view of Miles and Huberman (1994). Further, the goal of qualitative research is to describe a phenomenon or an event with the goal to understand a certain activity or give a interpretation to a phenomenon that is meaningful in a theoretic point of view.(Eskola and Suoranta 1996).

The sampling data is chosen by judgement sampling. According to Hirsjärvi et al. (2009), using judgement sampling data is an important feature in qualitative research. By hand-picking the suitable interviewees, the best understanding of the phenomenon can be reached and the challenging areas of Devops uncovered. One of the contributions of qualitative study is describing a phenomenon in depth rather than making extensive, generalized assumptions based on statistical analysis of data (Koskinen et al., 2005). Therefore, based on the amount of the available data, the thesis favors this research strategy.

Theme interviews were chosen as the method of data collection for this research. In order to zoom in to potential challenges in the Devops approach without making any assumptions beforehand, the data collection method needed to be flexible. According to Tuomi and Sarajärvi (2006), interviews are the recommended method of data acquisition where flexibility is important.

Additionally, the value of interviews also lies in the ability to ask for clarification and to find out more about specific issues mentioned. This holds especially true for themes that have not been previously explored. (Hirsjärvi et al., 2009).

Theme interviews are adaptive in the sense they support making follow-up questions and modifications if an interesting topic comes up. The aim of this research was to find out and understand potential challenges in the field of Devops. Therefore, targeting questions and predefined discussion topics were decided with the overall aim of the research in mind (Tuomi & Sarajärvi, 2006).

Finally, an abductive approach was chosen to carry out the interviews. According to Dubois & Gadde (2002), abductive approach stands for first exploring and analyzing the existing theory followed by the empirical phase of the research. The abductive approach was necessary for this study to first find out the main components Devops so all necessary areas could be covered in the interviews.

4.2 Data Collection

The data collection was carried out by semi-structured theme interviews with nine Finnish IT-professionals with previous in-depth experience of Devops initiatives. The interviews were structured by the CALMS-model described in the theory part of this thesis. Each section of CALMS was supported by targeting questions. However, conversations were allowed to develop freely in any direction in order to learn more about the different aspects of Devops as well as to uncover new sources of challenges and points-of-view.

4.2.1 Interviews

The study consists of nine semi-structured theme interviews with industry experts. The experts mainly come from organizations that have seen a Devops-transition and many of these interviewees were key sponsors of these transitions. Additionally, two of the experts come from companies that work with Devops transitions as a business. While the number of interviewees is low, they have extensive knowledge on the subject and some of them can be considered key players in the field of Devops in Finland. Gomm et al. (2009) state the quality of the study is not influenced by statistical significance. Taking a deep scope, rather than scratching the surface, allows to concentrate on the reasons and the ways to overcome challenges in Devops implementations, as supported by Dubois and Gadde (2002).

The data collection interviews were carried out between February 19th and March 8th of 2016. The interviewees were contacted through LinkedIn and e-mail and they were carried out at company premises or their nearby restaurants. On two occasions, two people were

interviewed simultaneously. The length of the interviews varied between 47 minutes and 1 hour 7 minutes. The average length of the interviews was 59 minutes. The interviews were held in Finnish and were recorded on the permission of the interviewees. The interview questions were based on the key areas of Devops, aiming to guide the interview. Most of the questions asked to describe a Devops transition process while a minority of them asked specifically to name challenges in related areas.

A summary of the interviews is displayed in Table 3.

The interviews were transcribed on March 14-March 20. The transcribed content was analyzed and given a title based on the topics mentioned in the interviews. The categories were coded thematically using Microsoft Excel. Further, the findings were grouped and categorized by axial coding into separate categories based on their similarity. The table of the categories is displayed in chapter six. The individual nuances are not included in the table due to space limitations. However, most of them emerge in the analysis section.

Table 3: Summary of the interviews

Date	Name	Title	Company	Location	Duration
19.2.2016	Lauri Halkosaari	CTO	Schibster Media / Tori.fi	Helsinki	1h 2min
22.2.2016	Juha Lehtosalo & Kalle Ylä-Anttila	Portfolio Manager, Software Architect	YLE	Helsinki	1h 2min
22.2.2016	Erno Aapa	Swiss Army Knife	Sharper Shape	Helsinki	1h 0min
23.2.2016	Niilo Ursin & Tomi Vainio	CTO, Senior Production Manager	Alma Talent / Kauppalehti	Helsinki	1h 07min
2.3.2016	Pasi Katajainen	Head of Technology	Nordcloud	Helsinki	51 min
8.3.2016	Joakim Sandström	Software Architect	Aktia	Helsinki	47 min
8.3.2016	Marko Klemetti	CTO	Eficode	Helsinki	1h 07min

4.2.2 Interviewees

A total of nine people were interviewed in the data collection phase. The interviewees were picked by first identifying Devops initiatives in Finland (Commitcom, 2014). Next after getting responses from the first interviewees, I got recommendations from their personal networks for additional people. All interviewees are top professionals in the Finnish Devops-scene and have been deeply involved with Devops transformations in their respective companies.

The remainder of this section will give an overview of the interviewees and their involvement with Devops.

Erno Aapa

Erno Aapa is the founder of Devops Finland, a community of 750+ members attending Devops-related meetups and promoting discussion about the approach. Originally a developer, Aapa has experience in Devops consulting to companies such as Rovio and Elisa. Currently Aapa is working at Sharper Shape with focus on management and team leadership.

Lauri Halkosaari

Lauri Halkosaari is the CTO/CIO of Schibster Media / Tori.fi. With a backround in full-stack development, web services and project management, Halkosaari was leading the Devops transformation at Fonecta before going to Schibster.

Pasi Katajainen

Pasi Katajainen is the Head of Technology at Nordcloud, a leading cloud architecture and Devops consultancy company in Finland. Originally a developer, Katajainen has experience at Nokia's HERE-unit in managing its cloud transformation where he got acquainted with challenges of Continuous Integration and Devops generally. Nowadays working closely with Devops-workshop-based solutions.

Marko Klemetti

Marko Klemetti is the CTO of Eficode. Eficode is one of the leading Devops consultancy service providers in Finland. Klemetti has been involved in many Devops implementations starting from the early days of Devops.

Juha Lehtosalo

Juha Lehtosalo is Portfolio Manager at YLE, the Finnish national radio. His area of responsibility is the development of YLE.fi web service. Lehtosalo has played a key role in the Devops transformation at YLE web services.

Joakim Sandström

Sandström is an application architect in Aktia. Aktia is a large bank in Finland. Sandström became involved with Devops through the banking renewal project in Aktia, that aims to renew all basic banking services with the use of 3rd generation software development.

Niilo Ursin

Niilo Ursin is the CTO of Alma Talent / Kauppalehti. With both developer and operations background, his team is responsible for the technology behind digital content. Ursin has been of the key drivers in Devops transformation at Kauppalehti.

Tomi Vainio

Tomi Vainio is Senior Production manager at Alma Talent / Kauppalehti. Before his 14-year career at Alma Talent, Vaino has also worked at Elisa and Sonera. Along with CTO Niilo Ursin, Vainio has driving the Devops initiative at Alma Talent / Kauppalehti.

Kalle Ylä-Anttila

Kalle Ylä-Anttila is Application Architect at YLE, the Finnish national radio. With 9 years of experience at YLE, his responsibilities lie in leading the operations-team and Ylä-Anttila has been deeply involved with the Devops-transformation at YLE.fi.

4.3 Quality of the Research

The quality of the research refers to validity and reliability of the research. Validity describes how well the research is mirroring the claimed research objectives. Reliability is referred to the repeatability of the results in methodology research. Tuomi and Sarajärvi (2006) claim that there are no specified rules to measure validity and reliability. Measuring validity and reliability in quantitative study is considered easier than in qualitative study since all findings are quantified and the research can be repeated using different sets of data. Using numbers to measure validity and reliability makes it easy to test the research (Hirsjärvi et al. 2009; Uusitalo 1991).

Qualitative research complicates the measuring of reliability and validity. According to Hirsjärvi (2009), the reliability and validity should still be addressed in all types of research. In order to increase research reliability and validity, a detailed description of how the research is conducted should be present. Koskinen et al. (2005) also argue that concentrating on a limited number of observations in qualitative research enhances reliability. Taking these points into account, this research follows the aforementioned guidelines to improve the quality of the study.

5 Analysis

This section analyzes the information derived from the interviews. The thesis uses a data driven analysis for the purpose of creating a theoretic body of the qualitative data that is acquired by means of theoretical research. Data driven analysis can be divided into three phases (Miles & Huberman 1984). This research leans on these three phases in processing the acquired data.

1) Reduction of the data (simplification)

2) Clustering of the data (categorization)

3) Abstraction of the data (creating theoretic concepts)

In order to reduce the data into manageable entities, I first identified the broad concepts of different challenges and linked similar stories from different individuals together. I created an Excel-spreadsheet to manage the data and prioritized the challenges according to the frequency they were mentioned in interviews.

Next, the data was clustered into different categories. There were 16 sub-challenges identified. Further grouping resulted in four general-level challenges that are listed below:

Challenge 1: Lack of Awareness in Devops

Challenge 2: Lack of support for Devops

Challenge 3: Implementing Devops technology

Challenge 4: Adapting organizational processes to Devops

The following section describes each channel in detail.

5.1 Lack of Awareness in Devops

As anticipated, the lack of clear definition of Devops brings forth confusion. The concept of Devops is not well understood, since the concept has not yet sufficiently matured. Also since Devops is such a hot word currently, a certain amount of resistance to 'buzzwords' was also identified in the interviews. The lack of awareness inside an organization creates bottlenecks and people-dependencies.

5.1.1 Maturity of the concept

One of the main challenges in bringing Devops to an organization was found to be the actual concept of Devops. The lack of definition for Devops was found to be confusing in six

of nine interviews. This creates problems in getting organizations aboard to start embracing the approach and selling Devops-consultancy services to customers.

According to Ylä-Anttila, a common misconception about Devops is that it is a "product" or a "toolset" that "one can buy straight from IBM and be done with it". Mentioned both by Ylä-Anttila and Halkosaari, some configuration management software companies seem to be selling their products as a "Devops suite" that further promotes the confusion. As I mentioned earlier, the market for tools that can facilitate the automation part of Devops is growing rapidly and these companies are taking full advantage of the buzzword Devops. Furthermore, Halkosaari mentioned that consulting companies are also trying to capitalize on the buzzword by offering Devops as a package to customers whereas the real advantage of Devops is organizational and process change.

Another common misconception was that Devops is either a "Devops guy or a Devops team" (Katajainen) or a "separate operations team that is doing some coding" (Ylä-Anttila). Neither is it a role or a title (Aapa). The problem is that everybody is using Devops the way they see it and how their organization has been inspired by it. Aapa stated that the usage of the term is also dependent on the size of the organization: for startups Devops might entail radically different things than for large enterprises. For instance, startups are usually born cloud-native and are operating on a pure PaaS infrastructure, for example Heroku [3], which is closer to a NoOps-model where all operations are automated and handled by the service provider. In contrast, large enterprises might operate on a mixture of self-hosted and IaaS infrastructure that requires more configuration management, coding infrastructure and automation. To conclude, Devops for startups might mean pure development with no operations while Devops for enterprises might mean the whole product development process going even further over the boundaries of both development and operations to involve product owners as well as quality assurance.

On the other hand, Halkosaari brought forth an alternative point of view: it can also be seen as a great freedom and strength for Devops that it is not centrally directed by any foundation in contrast to Scrum foundation or ITIL and the British Military. Devops gets its trends and best practices directly from the community and interested industry professionals – everything is kept lean and therefore true to the approach.

[3] Heroku is a leading PaaS (Platform-as-a-Service) provider that offers an automated platform to host applications without the developer having to worry about the underlying infrastructure.

5.1.2 Allergy to buzzwords

There are several interviewees who see a problem with the Devops-buzzword. Partly due to the fact that Devops is currently such a hot word and partly due to its ambiguity and "wild" usage, the value of the word has decreased and Devops is now met with a certain kind of reluctance. Aapa mentioned that the "allergy" can be seen in people that are naturally against trend phenomena, people who think "it is too mainstream, I am against it even though I do not know what it is really about". This is important to mention because it affects how to sell the idea and how it is received on the other end.

Many of the interviewees suggest a strategy of dropping the umbrella term 'Devops' and talking about reducing lead time and introducing automation instead, while others say the "hotness" of Devops actually works to its advantage or that the reluctance can be used to educate the other party.

5.1.3 Lack of Awareness

Devops is a new concept for many people. As mentioned before, it is a challenge to grasp the whole concept of Devops. According to Vainio, for management, it might not be even necessary to understand Devops but for the people inside the IT organization where the change is taking place, the concept needs to be shared and communicated in order for everyone to work together with the level of collaboration Devops requires for it to work.

The overall lack of awareness about Devops is viewed as a challenge by several interviewees. Even though the majority of the organization might already be Devops-oriented, there are always areas in the organization where Devops has not reached yet. Not limited to the concept of Devops, the components such as Continuous Delivery or Continuous Integration may be unknown.

In an organization that is moving fast with these technologies, training all necessary people can be easily overlooked. Formal training is not enough. Klemetti noted that sponsors for Devops are usually the key people that share knowledge inside and across teams. Often they do not have higher position in a team and therefore they are a bit more difficult to find. Moreover, managers might not even know that these people are key resources in the teams for sharing and educating their colleagues. If the specific knowledge needed to use the tools is not readily available, the people dependencies required to operate them become bottlenecks and risks for continuity, as mentioned by Aapa.

Finally, sharing and spreading the knowledge of Devops is considered easy in the start when large changes are happening in parts of the organization. Once an equilibrium is reached and business is going as usual, sharing and spreading common practices may easily be forgotten. Sandström claimed that implementing a Devops-approach in new teams usually re-ignites active sharing but promoting sharing as a routine can be difficult.

5.2 Lack of Support for Devops

Another central challenge identified was the lack of support for Devops. The lack of support may show on several levels: on the management level, on the team level or as an overall lack of trust.

5.2.1 Lack of management support

Lack of management support is considered one challenge in Devops. Because Devops has to do with lots of changes to the ways different teams are working on a daily basis, a high commitment and support at the top level of the organization is necessary to change the company culture. Klemetti says that The role of the managers is to "untie the knots executive personnel cannot untie themselves", to "break down the walls" and allocate more time for communication and sharing.

According to Klemetti, there are three profiles of managers that each reflect their attitudes towards Devops: tech-savvy, open to ideas and teflon. Tech-savvy managers are the easiest ones to convince. First of all, they usually have heard of Devops and are open to discuss it. If the argumentation is made correctly, the ideas will usually go through without problems. For instance, Ursin described the management at Alma-Talent tech-savvy and therefore IT could get their initiatives to develop infrastructure even at the cost of a longer overall project lifecycle approved easily.

Managers open to ideas are usually willing to discuss an idea but they are more concerned about the financial side of the changes. It might be challenging for a development organization to translate the improvements into cost reductions and other financial gains. Therefore, argumentation must be well thought out for it to pass. Fortunately, Devops can be tried out with small initial commitments. Finally, managers with teflon profiles are the most difficult to deal with. In many cases, these managers are not even aware of the issues a Devops initiative is trying to solve. Thinking that the organization is doing just fine with current methods is quite common, or the notion that current systems are so complex they cannot be developed in the

proposed way. Klemetti argued that it might end up in a sort of "guerilla warfare" with the teflon managers, training people to do things differently one person at a time without any help from the management. These kinds of endeavors may still end up in failure, Klemetti said, giving an example of an anonymous company where the main sponsor for Devops was lost when the management saw the person as a burden and decided to let the person go, even though the person contributed to Devops in that organization significantly.

Table 4: Manager profiles as described by Klemetti

Manager profile	**Description**	**Strategy to gain support**
Tech-savvy	Prior knowledge of Devops, open to new technologies	Reasonable argumentation
Open to ideas	No prior knowledge of Devops, open to new ideas	Argumentation based on financial gains
Teflon	No prior knowledge, no realization of issues	"Guerilla warfare"

The lack of "real" support from management is also an issue once the decision to adopt Devops has already been made. Klemetti stated that more often than not, management does not understand the scope of the change. Simply throwing money at Devops does not lead to much, rather there are several aspects that the management needs to address. First of all, according to Katajainen, change is often mistakenly viewed as a R&D change where, for instance, an automation pipeline is set up. Klemetti explained that the change in culture needs to be facilitated by empowering different boxes with different communication tools and practices and seeing this as a value-adding measure: not just wasted time away from "actual" development work. Along with culture, staff needs to be properly trained to new processes and tools and it needs to be determined whether any 3rd party consultants are needed in any part of the change process. Moreover, the nature of product development changes from iterative work to continuous development and therefore, for example feature specifications need to generated accordingly.

5.2.2 Lack of team-level support

Lack of support can also be seen on the team level. The main reasons for lack of support lie in the change of working methods, change of roles and the organizational readiness to

communicate. The change of working methods is perhaps the most profound change in Devops. As mentioned in the literature review, developers take more responsibility in running software, operations take more responsibility in building coherent, preferably code-driven infrastructure and quality assurance can focus on actual quality and not manual testing. As many tasks are automatized, the fear of a machine replacing jobs is a common one that raises opposition on the executive level.

There are roughly 3 types of issues related to this: lack of trust, lack of skills and lack of readiness to communicate. Lack of trust can be found anywhere where work gets shifted away from people. Aapa explained that in some cases, especially when development is tasked with coding infrastructure, operations do not trust that development have the required skills to do this adequately, for instance setting up proper security measures etc. Moreover, Sandström added that people might be reluctant to let automation handle tasks that would require administrative rights – doing these tasks manually through change management processes ensure that it has been approved by the required people and not by an automated sequence.

Lack of skills is a very common issue when dealing with new tools and processes. Katajainen and Klemetti both realized an issue where there might be some people in the organization, especially operations and QA people that are intimidated by these new tools and skill requirements and instead of striving to learn them, they initially choose to oppose change. The key to solving this issue is to communicate that the change is necessary and once the skills have been acquired, the job will actually be more enjoyable. At the same time the members of staff have increased their market value by learning new skills. Among the interviewees, the fear of operations or QA staff losing their jobs was not encountered, although Devops can be sold to an organization as a means to reduce the need of manual work.

Finally, Halkosaari pointed out that inertia at the team level may rise from outside development and operations, namely from stakeholders who rely on traditional means of communication such as ITIL and release management, change advisory board (CAB) meetings and steering committees. While many of these practices prevail in a Devops organization, the nature of communication changes from a structured set of practices to free and transparent ones, with the aid of new tools and processes. Sandström added that the organizational structure might also be an impediment to free collaboration.

5.2.3 Lack of trust

Somewhat related to the challenges with managerial support, Lehtosalo and Ylä-Anttila identified a challenge more widely related to Devops adoption: earning and keeping up trust with stakeholders who are not involved with day-to-day operations.

The challenge is the clearest at the very top level of the organization. In the traditional model, the services are developed in steering committees and CAB meetings and other agreements where deployment dates are set and approved. However, Ylä-Anttila noted that in a Devops approach the batch sizes are smaller and releases more frequent so many of these meetings can only deal with more abstract, higher level changes. This shifts responsibility more towards the development team and the teams are getting more and more autonomous. In the process, a lot of the information and documents produced by the development teams to bring to these meetings are not produced anymore (such as weekly reports mentioned by Ylä-Anttila).

Without these agreements and reports, all stakeholders do not have a clear view of the day-to-day operations which may lead to situations where they are doubting whether the development teams "know what they are doing" according to Lehtosalo. This requires a certain level of trust between stakeholders and development teams without having to "constantly remind everyone that they are professionals and letting them do their work".

Another related challenge is contracting external operations resources to work in teams that are doing Devops. Previously work was contracted on a transactional basis where a contractor would offer a certain service based on agreements such as SLA or deliverable based on a specification document. What Lehtosalo noticed at YLE was that it became significantly more difficult to contract people to work in teams, and managing these contracts were more difficult because they were now based on trust and not hard deliverables such as service availability or some feature in a product. The performance of these resources could not be determined in the same manner as in the agreement-based model.

5.3 Implementing Devops technology

The key technological breakthroughs in Devops is the creation of an automated continuous delivery pipeline. This new way of delivering software has a profound impact on the processes in development, quality assurance and operations. "Automatizing inefficient processes leads to automatizing inefficiency." (Jyrki Kasvi) Sandström used the quote to point out that setting up automation technology alone is not sufficient. While automation promises benefits in lead

times, application stability etc., implementing this technology and processes is not entirely challenge-free.

5.3.1 Automated testing

7 of 9 interviewees found challenges with building automated testing inside the development process. Many of these challenges are people- and skill-related, while others are more related to the way testing is organized in the new approach. In pre-Devops models, testing is usually handled downstream, that is, after features have been developed and handed off to the next person. Devops automation requires tests to be written simultaneously as a part of the development process. As a result, the need for manual click-based testing is reduced, testing time is minimized and possible breaking errors can be noticed more easily since tests are run more often and in the exact same manner each time.

One challenge mentioned by several interviewees was the attitudes of developers towards writing automated tests. First of all, Katajainen and Vainio both pointed out that even though new practices regarding testing have been agreed upon, there are developers that have a hard time embracing these practices and continue to develop without writing tests. According to Katajainen, writing tests is not considered fun nor a part of developers' role. It might even be considered waste. Ylä-Anttila added that many still assume testing is someone else's responsibility. Finally, developers might not have adequate skills to do testing properly. For instance, the lack of knowledge and know-how of different testing frameworks is quite common but somewhat surprisingly there have been cases where some developers did not know automated testing frameworks even existed.

Aapa named a few measures that can be taken to combat these reluctant attitudes towards testing. Leading by example is effective: writing a few tests that the team can take a look at makes the team understand what is expected of them. Additionally, when the team is making pull requests[4], the person who makes the approval should not do so before test coverage is adequate. By making it clear that no changes go through to production without the agreed level of tests forces developers to make writing tests a part of their development routine. In doing so, it is also made clear that the responsibility of testing lies on the developer, and not the quality assurance people whose job is not to click through the application every time there is a change ready to be delivered.

[4] A pull request is used to request a finished code commit to be merged into the common code branch of a product

Another aspect is defining the added value of writing automated tests. Since writing tests is also a financial issue it is often a challenge to find the correct level of test coverage. As Vainio put it, tinkering with tests is not their core business. Ursin reminded that all software is not equal: critical functionalities such as functionalities that handle monetary transactions should be tested thoroughly while others, such as components for internal use should only have a rudimentary coverage. It is up to the developer to understand the required level. In some cases, specifically in the banking industry, Sandström claimed it might be cheaper to have a manual tester do the job rather than spending too much development time in complex test cases.

5.3.2 Automation tool challenges

Five out of nine interviewees identified challenges with automation tools, namely determining which tools are right for the project and the maturity of automation tools. The challenges with automation tools were not considered technical - the configuration of these tools to work in a proper continuous deployment pipeline was considered a laborious task but not a challenge per se.

Determining the suitable tools was considered a challenge since the industry is young and there are numerous different options. Katajainen highlighted the main decision points: will the company go with open source tools that are easier to replace or will they commit and pay for these tools? Additionally, should the tools be hosted locally in some environment or should they be cloud-based? Are there restrictions to which tools an organization can use? For instance, if an organization is already using a test framework, is it "Devops-friendly" and if not, can it be replaced?

The other challenge related to tools is their maturity. Since many of them are quite new, they do not always work as intended, they do not support all needed environments or they might become outdated too fast. Unfortunately, the most common solution seems to be waiting for new versions to come out or trying to find workarounds using other tools. Aapa said it is not necessarily a bad thing: building a toolset out of various pieces that are easily replaceable also prevents vendor lock. However, the downside there is the ever-increasing skill requirements for different software.

5.3.3 The type of the application

The final set of challenges in automation is related to the type of the application that is the target of automatization. Three interviewees of nine identified this challenge. The most common problem is that the application architecture is not at all suitable for virtualization.

Klemetti mentioned mainframe computers that are still quite common in banking- and insurance industry, while Halkosaari mentioned the type of architecture that has negative scalability – this can be databases that only have a limited amount of socket connections etc.

One response to these kind of architectural limitations is splitting down the application into small units, sometimes called microservices, and virtualizing as many of them as possible. This leaves only the core fraction of the service to run on the legacy hardware, and other parts of the service can now be developed with a Devops-approach. While all of this might be possible, many interviewees (eg. Halkosaari) reminded that all transitions of this magnitude should only be considered if the business case is well-established. Klemetti noted that in some cases, it might be wiser to use the allocated automation money on developing a new, modern system from scratch.

5.3.4 Fragmentation of tools and practices

Another related set of challenges lie in common practices both inside and across teams. First of all, the use of various technologies and tools across teams is considered a challenge. Ylä-Anttila and Vainio claimed that having different environments for continuous integration and not having a common deployment pipeline is an anti-pattern for the very foundation Devops is built on.

There are several ways of overcoming this. Keeping all application code in version control and available for everybody to audit and review makes all application development transparent and prevents reinventing the wheel effectively. Moreover, people from all teams can contribute to the code if they need a specific functionality implemented. This is especially true with infrastructure. Aapa explained that rather than making a different version of the infrastructure to use with a specific application, the enhancements made are available for all other applications that use the same infrastructure template as well. Subsequently, it prevents fragmentation and people dependencies.

Sometimes it is necessary to use different technologies for application development. With the help of containerization, different software components written in different languages can function on the same infrastructure. Ylä-Anttila and Halkosaari both mentioned the use of Docker which is a platform containerization software to address this issue. As a result, the programming language can be chosen somewhat freely by the developer. In other situations where the use of containerization is limited (especially with legacy systems), building common Application Programming Interfaces (API's) around the special software helps isolate it from

the common architecture while still providing the same kind of programmability as other common software do.

Additionally, the use of different communication channels across teams is to be avoided. All but one interviewee said their IT organization has started using Slack to counter this issue. In addition to basic chat functionalities, it's beneficial to be able to follow and subscribe to conversations based on their topic, which promotes cross-team communication.

5.3.5 Finding the right scope for monitoring

With the help of Devops and new tools such as the cloud-based New Relic, it becomes easier to monitor different services in a consolidated view. However, when more and more operations duties are shifted towards development, "No-Ops" being at the far end of the spectrum, it becomes also increasingly important to monitor the right things.

Many of the systems are better-equipped to raise alerts when a defect is found but this has introduced a problem with prioritization: with an increasing number of alerts it is often difficult to find which ones are critical and which ones need attention. Vainio and Sandström noted that receiving hundreds of different alerts daily from different sources is sub-optimal and costly to inspect. Therefore, setting ground rules for incident severity prioritization, used channels and which people are notified is necessary to overcome this challenge.

5.4 Adapting organizational processes to Devops

Devops promises speed and flexibility in the delivery of software. The speed and flexibility might not be fully realized, however, if the organizational processes do not accommodate it. The processes have to do with Devops initiation, software development mode, change management processes, metrics and team-related challenges.

5.4.1 Starting with the correct scope

A challenge that was frequently mentioned was the notion of starting with the correct scope in regard to Devops transition. As explained in the literature review, Devops calls for the virtualization of IT infrastructure. For existing applications, it means the server stack needs to be converted to code that is able to provision the functional equivalents of previous hardware servers in order to run the application. For new applications the process is much easier: starting with a clean slate the infrastructure can be configured without any limitations using the latest

available technologies and industry standards. Furthermore, the monetary investment is a significantly cheaper than in a traditional setting since no new hardware is usually needed.

While the application may be at the core of Devops, Klemetti reminded that the process and practice changes related also to software development, culture and IT governance have such high significance that trying to start off with too much can become overwhelming.

Katajainen elaborated that starting off with several applications may result into reinventing the wheel when solving similar problems among projects. Additionally, solving the same problems in an uncoordinated manner can also lead to different solutions that cannot be later unified. A small scope proof of concept ensures that the automation is set up in a way that it can be reused later in other projects and that the governance model works the way it is intended before any of it is rolled out to the rest of the organization. Also Halkosaari noted that a proof of concept works: using only a fraction of the cost of a traditional software project, a successful Devops-oriented initiative can easily gain traction in the organization since it is possible to realize early wins and thanks to new measuring practices, the results can also be quantified.

Klemetti named four factors affecting how much an organization can do at the start of their Devops-initiative. The most important factor is effective agreements. If, for instance, the organization's Operations is outsourced to an external vendor, there is little an organization can do to existing applications before their current contract expires or is renegotiated. The second factor is the type and scope of IT-infrastructure. There are still many companies that use mainframe computers or other special hardware as key components in their infrastructure that are not easily virtualized. Thirdly, application architecture also affects the outcome: using software technologies that are easily containerized makes them very flexible for Devops approaches. On the other hand, using complex application architecture that have a lot of dependencies in various places might make the job more difficult. Finally, Klemetti named company culture as the last factor. The level of communication in the organization is a major component. The key question is: Does the organizational culture make it possible for developers, quality assurance, IT operations and those responsible for product specifications to come together and develop the product or service? Are they even in the same country? As stated above, taking a small-scale approach to Devops with early wins decreases the risk considerably and buys time for the rest of the organization to adjust to the new ways of working without having to make a considerable leap of faith.

5.4.2 The mode of Software Development

There was a very common challenge associated with Scrum as an Agile software development method. The main challenge with Agile, according to Klemetti, is that many organizations are developing the software in sprints without the goal of releasing the features immediately when they're finished. "You can do agile forever without ever releasing your software. Agile itself does not translate to added value for customers", Klemetti explained. Moreover, Aapa claimed that doing Scrum by the book is too rigid for the purposes of Devops. The model where a release package is created at the end of each (2-week) sprint does not fit well into the Devops-approach where the goal is to release once a feature is completed, often several times a day. This is the reason why Kanban fits into the Devops approach better.

Many interviewees explained that their organization started out with Scrum by the book. However, according to Lean principles many teams were allowed to modify and cut down on the ceremonies that were considered waste. After the process, many organizations found themselves doing "something similar to Kanban" (Vainio). Teams got to decide autonomously the best ways of working after having experimented what works in Scrum. For instance, at Alma Talent, the daily Scrum check-up meeting was considered useful but not necessary to meet face-to-face every day. It was therefore moved over to the chat-tool Slack instead. Aapa noted that a good feature of Scrum is that it is better-equipped to guide overall product development on a longer time frame, however there are other tools that can be used to tackle that.

Other challenging aspects with software development methods were identified when working with 3rd party vendors. Working methods between the two parties may conflict: the 3rd party might not be able to deliver features reasonably at the same pace the procuring organization expects it to. Also specification changes in a market-driven, volatile environment may prove to be challenging for the vendor to make if they work in a non-agile mode that does not allow specifications to change.

For instance, some of YLE's development is procured from third parties and the agreements were traditionally made on delivery-basis. According to Lehtosalo, this means that YLE was paying vendors to develop features and these features were considered as investments, which in turn had pressure to increase YLE's value. In a continuous delivery setting, however, batch sizes are smaller and delivery is much more incremental, it was challenging to determine the price and value for each small increment. Therefore, the agreements were renegotiated to

procure development work instead of features which also had major implications on budgeting: instead of release-based budgeting, budgets had to be converted into accommodating a reasonable amount of continuous development.

5.4.3 Change Management Processes

The other major challenge with Devops-style continuous delivery is how it fits into the organizations current change and release management processes. Very often enterprises follow for example ITIL processes by the book. The processes are not necessarily designed to handle the amount of changes and the velocity of development Devops can bring to an organization.

Klemetti offered a point of view where ITIL is especially good for "writing in stone" who to blame when something goes wrong with a change. The approvers are usually held responsible. However, Devops changes the situation and brings responsibility back to those that actually develop the software. Aapa stressed that it is extremely important that the developer is the one who takes the code into production. Not wanting to break anything, developers are forced build quality and integrity into their code. The reasons are transparent: every change can be tracked at the source code level – who changed and what (in the Git version control system a view that shows line-by-line code changes is actually called "Blame") in the delivery pipeline. There is no longer a need for rigid change processes.

Ylä-Anttila described their solution to the challenge: making own adjustments to the process to better serve the product development is necessary. Therefore, at YLE they introduced "Domain CAB" –thinking where there is a person close to the team that is responsible for change management related to their specific domain. As long as the domain CAB-person has a holistic view of what is going on with their respective service the approval chain does not extend further. For larger releases such as a new version a more formal process is in place but other than that changes are mostly business as usual. Aapa's views are in line with Ylä-Anttila's: the change ticket is informative, describing the change that was already deployed to production. According to Lehtosalo they have solved the problem in a manner that there is one change ticket that "basically says we're going to make changes all the time. So far the arrangement has worked out fine with the management, but if something goes wrong then we're back at the discussion whether these models should be based on trust or more formal agreements", Lehtosalo noted.

All interviewees that saw change management as a challenge agree that ITIL, for instance, is still considered a well-thought library of principles but the prescriptive nature of some practices

does not support Devops in all cases. Therefore, local adjustments and flexibility is necessary to overcome the challenge.

5.4.4 Adopting new metrics

Finding common metrics for both development and operations as well as quality assurance was considered challenging. Traditional metrics such as uptime and the amount of tasks completed in a Scrum sprint, for instance, are still viable but the focus is shifting. The "Devops" metrics such as lead time, code quality and overall system health can be measured with new kinds of tools. Once the metrics are in place, it may take time for the numbers to be comparable with other applications. According to Klemetti, there are managers who might be over-enthusiastic with these new metrics. When these metrics are first put in place for an application that has been in development for several years, it is natural for these metrics, such as code quality, to be below average. The challenge is finding the balance between improving the score of the metrics and keeping up a rapid pace in development. Over-emphasizing the metrics can have an adverse effect on the development since a lot of time goes to refactoring code and making the application perform better from a metrics-point of view.

5.4.5 Team challenges

When development and operations are brought together to form Devops, the idea is that the amount of collaboration and communication between the two increases. Several aspects, such as location, the time spent together with other teams and product teams have some associated challenges.

Several interviewees recognized that having development and operations in separate locations is challenging for Devops. Unfortunately, an organization structure usually mirrors the way software is built (Conway's law) which is an impediment to open communication. The same goes with other areas of the product development as well. Klemetti mentioned a worst case scenario where some organizations still send their product specifications over from another building as printed documents without meeting with development in order to explain it all. In cases where co-location is not an option, Katajainen suggested visiting other teams at least once a day. Klemetti said that the only logical solution is the simple one: Co-locate development and operations.

While co-location serves as an informal medium of communication, there is a need for other, arranged meetings with teams as well. Ursin, Vainio and Ylä-Anttila all mention demo days where teams come together to show other teams what they have been working on. Moreover,

hackathons (intense coding periods where something functional is built from scratch) work towards educating people about different tools and technologies as well as getting to work with different kinds of people from different teams.

6 Findings and Discussion

The purpose of this section is to summarize the findings relevant to the research questions:

1. *What are the key components of Devops?*
2. *What are the main challenges organizations face with Devops?*

The secondary research question attempts to describe how these challenges can be overcome:

3. *What are the ways to overcome these challenges?*

The main function of the literature review in chapters 2 and 3 was to answer research question 1. The second and third questions was based on the qualitative research analyzed in chapter 5. Further, chapter 7 presents implications that contribute to answering question 3.

There were 16 notable challenges mentioned in the interviews. The four main challenges that emerged from these findings are: Lack of Devops Awareness, Lack of Support, Implementing Devops Technology, Adapting organizational processes to Devops

The following table summarizes the findings for each main challenge.

Table 5: Summary of challenges found in the study

CHALLENGE	SUB-CHALLENGES	REFERENCE	FREQUENCY
Lack of Awareness in Devops	Maturity of the Concept	Halkosaari, Lehtosalo, Ylä-Anttila, Aapa, Sandström, Katajainen	6
	Allergy to Buzzwords	Halkosaari, Aapa, Ursin, Sandström, Klemetti, Katajainen	6
	Lack of Awareness	Halkosaari, Ylä-Anttila, Aapa, Klemetti, Katajainen	5
Lack of Support	Lack of management support	Halkosaari, Lehtosalo, Vainio, Klemetti, Katajainen	5
	Lack of team-level support	Halkosaari, Ylä-Anttila, Aapa, Vainio, Sandström, Klemetti, Katajainen	7
	Lack of trust	Lehtosalo, Ylä-Anttila, Aapa	3
Implementing Devops Technology	Automated testing	Ylä-Anttila, Aapa, Ursin, Vainio, Sandström, Klemetti, Katajainen	7

	Automation tool challenges	Aapa, Ursin, Sandström, Klemetti, Katajainen	5
	The type of application	Halkosaari, Vainio, Klemetti, Katajainen	4
	Fragmentation of tools and practices	Ylä-Anttila, Aapa, Ursin, Vainio, Sandström	5
	Finding the right scope for monitoring	Ylä-Anttila, Vainio, Sandström	3
Adapting organizational processes to Devops	Starting with the correct scope	Halkosaari, Ylä-Anttila, Vainio, Sandström, Klemetti, Katajainen	6
	Mode of Software Development	Halkosaari, Aapa, Ursin, Vainio, Sandström, Klemetti, Katajainen	7
	Change Management Process	Halkosaari, Lehtosalo, Ylä-Anttila, Aapa, Klemetti	5
	Adopting new metrics	Ylä-Anttila, Katajainen, Klemetti	3
	Team challenges	Halkosaari, Aapa, Vainio, Katajainen	4

The first main challenge of Devops is the lack of awareness. The novelty of Devops as a concept results in ambiguity, confusion and the word meaning different things to different people. Moreover, Because Devops is a hot buzzword at the moment, people can grow reluctance towards it which hurts organizational buy-in and the sales of Devops-related consultancy services. Additionally, Devops is not adequately known throughout the IT organization and training these people is often overlooked. Joe Hendren agrees in his blog that Devops has a marketing problem: “Clearly, the concept of Devops has a marketing problem, and it's making the hurdle to entry for new participants in the Devops community needlessly high.” (Hendren). Another blogger, Topher Marie (2014) notes that an organization’s business drivers change the definition of Devops in their respective organization. In resonance with many interviewees, Hendren advises to “Leave buzzwords at home”. However, Spikelab notes that labeling phenomena with buzzwords is necessary to cross the Chasm in the Technology Adoption Lifeycle where the majority of adopters are risk averse and require enough momentum and bandwagon effect to adopt a technology. Therefore, “being able to identify a

set of good practices and tools with a word like Devops is a necessity if we want to hope for larger adoption in the industry" (Spikelab). Spreading Devops to all parts of the organization needs proper management effort, where training and the identification of key sponsors is imperative for success.

The second main challenge of Devops is the lack of support. The subchallenges consist of lack of management support, lack of team-level support and the lack of trust. The main problems with management are "teflon" managers mentioned by Klemetti where Devops sponsors have to exercise "guerilla warfare" inside the IT organization, training individuals one person at a time to embrace the new ways of working. Many times management does not understand the scope of change: Devops is viewed as an R&D project instead of a major organizational shift (Moss-Bolaños, 2016). The 2013 State of Devops report (Puppet Labs, 2013) is in line with these findings: According to the report, 48% of the respondents stated that the value of Devops is not understood outside their group.

On a team level, the change of working methods and roles leads to inertia where automation that replaces manual tasks is not trusted. Joy Ma (2014) from New Relic also lists these challenges in her blog "5 Things You Need to Know When Implementing Devops". Both the interviewees and Ma suggest that showing concrete ways of making daily jobs easier and more meaningful via Devops is the key of overcoming this challenge.

Additionally, the readiness to communicate without fixed CAB meetings and change management processes can be an issue. Finally, without these fixed managerial tools the clear view of day to day operations is obstructed and needs more trust. The lack of trust between management and teams is considered an impediment.

The 2015 State of Devops report (Puppet Labs, 2015) states that IT managers play a critical role in any Devops transformation. Building trust and enabling the teams to fulfill strategic objectives is the key to Devops success. Ensuring that work is not wasted and investing in developing capabilities inside teams is necessary to overcome this challenge. Both the earlier 2013 report and the e-book by New Relic (2014) suggests that building open channels of communication between teams is essential to Devops success.

The third main challenge identified is Implementing Devops Technology. With automated testing, it was noted that there lies an initial challenge of team member buy-in. The attitudes behind automated testing need to be changed and training to write tests promoted. This can be

done with leading by example and setting the correct kind of approval gates to control and ensure that tests are indeed written in the required way.

Organizations have challenges choosing the correct tools, due to the related risks of application maturity. When choosing tools for automation, the risks of changing them often need to be taken into account. Therefore, the use of a setup where tools can be replaced and interchanged is recommended to overcome the challenge.

The type of application was found to be an issue since not all applications are suitable for virtualization. When either legacy applications or applications using special, non-common technologies are fitted into Devops, the use of microservices-architecture and containerization is essential. Finally, when establishing a culture for sharing and collaboration, extra attention needs to paid to common practices and tools. Riley (2014b) also identifies the challenge with fragmented tools when the original adopter leaves and more importantly in a governance aspect. To overcome the challenge, there is a need for common tools, rules and practices with both infrastructure and communications. Riley (2014b) also mentions tool management and the use of shared services. Using common infrastructure code, common API's and abstraction via the use of containers in order to use the same infrastructure for all applications regardless the technology used is necessary to prevent a fragmented landscape.

Finally, monitoring scope was also an issue to some interviewees. They speak of "alert fatigue", also mentioned by Bass et al. (2015). Aligning practices used in monitoring across different applications combats 'alert fatigue'.

The fourth and final main challenge relates to Adapting organizational practices to Devops. Firstly, establishing the correct scope when starting with a Devops initiative can be challenging, since there are no practices in place. The risk with doing too much results in reinventing the wheel with different applications in an organization structure that is not yet equipped to handle a large scale Devops transition. This view is supported by a blog post by Moss-Bolaños (2016), who suggests creating a tiger team with a focus to show early results in order to facilitate management buy-in.

The mode of software development is also considered a challenge. Doing Agile development without the goal of releasing features as soon as they are ready does not create value for customers. This is especially a problem where the goal is to release only when a Scrum sprint ends. Also, working with outside vendors in different mode of development is also problematic. These views are supported by a blog post "Devops and Kanban – a Match Made in Heaven"

(Upguard), where taking the "time box" out of the equation is deemed beneficial for Devops because it enables the team to work on a feature exactly the amount of time required with the required amount of resources. Additionally, reducing ceremonies that are considered wasteful and renegotiating contracts with 3rd parties to fit the development mode of the team are ways to overcome this challenge.

Heavy change management and approval processes such as those suggested by ITIL are considered challenging in a fast-moving development environment if they are followed by the book. Levitan (2016) notes in his blog post that these should be aligned carefully with the development velocity in order to prevent bottlenecks in delivery. Supporting the views of the interviewees, the processes need flexibility in order to make Devops work (Techbeacon, 2015). Further agreeing with the findings of this research an article by Forbes (Bloomberg, 2015) quotes that "many organizations have been able to modify ITIL to work within the Devops context, but it's unclear whether such rework will provide much value long-term. 'ITIL isn't part of the success patterns for the fastest moving, most innovative organizations,'" Finally, Chris Jackson, CTO of Rackspace sums this challenge up: *"We have seen a trend proliferated by ITIL and post-Enron legislation that led to the operations teams becoming mired in managing risk and compliance. For them it wasn't possible to think about doing things quickly. As a result, operations could find themselves bypassed by developers who wanted, and probably needed, to take risks and speed up deployment so they started to use shadow IT. Ultimately they were no longer aligned and working to completely different outcomes."* (Rackspace 2014)

Adopting new metrics in all aspects of product development was further considered a challenge. When these metrics have been found, there needs to be a balance where the metrics, such as quality of code, can be improved without the cost of slowing down the speed of development.

Finally, the challenges related to collaboration between teams were mentioned. When development and operations are brought together to collaborate in Devops, it is considered challenging if these teams are physically located in different premises. Additionally, the scarcity of random and free encounters with people from other teams constricts the flow of ideas and co-operation. To overcome these challenges, the interviewees stated that co-location is somewhat a necessity, and arranged meetings such as demo days help to "cross-pollinate" between teams to foster the kind of exchange that is required by Devops.

7 Conclusions

7.1 Research Summary

The constantly changing business needs and the requirement for faster time to market with software of present day has created a paradigm shift towards a 3rd generation Software Development philosophy called Devops. The lack of collaboration between IT Operations and Software Development as well as mismatch in configuration between development, testing and production environment has made deploying software releases slow and painful for many organizations. Different incentives between teams makes it difficult to work towards a common goal of bringing added value to customers.

A Devops approach to software development brings down the walls between the teams and align incentives through a collaborative culture, automation, lean principles, measurement practices and sharing. The benefits of Devops have been shown to be substantial with a significantly faster time to market and increased software stability. The organizational change is substantial which makes the challenges in adopting Devops an interesting topic to research. This thesis studied the challenges of Devops by interviewing nine experts who had been involved with Devops initiatives in their companies.

The findings were divided into four main challenge categories based on their topic. Due to the novelty of the approach, the concept of Devops for many is unknown or biased which hurts the overall implementation of practices. The lack of support in both management and organizational levels is a hindrance, since especially changing culture needs strong support and organizational buy-in in order to succeed. The toolset needed for Devops is particularly diverse and finding the fit, correct usage and attitudes towards that technology is challenging. Finally, when shifting to Devops that requires a certain level of lean principles and agility, aligning existing organizational processes such as the change management process to accommodate the new way of working was found challenging.

7.2 Implications for practice

This part of the thesis provides a foundation to evaluate the implications of the results and conclusions related to the challenges of adopting Devops. There are four categories to address each of the main challenges and provide means to overcome the challenges.

1. **CLEARING MISCONCEPTIONS AND SPREADING THE KNOWLEDGE**
2. **BUILDING COMMITMENT AND TRUST**
3. **ESTABLISHING COMMON WAYS OF WORKING AND LEADING BY EXAMPLE**
4. **ENSURING THE FLEXIBILITY OF THE ORGANIZATION**

7.2.1 Clearing misconceptions and spreading the knowledge

The main challenge behind Devops is the novelty of the concept. The findings related to the challenge encourage clearing any confusion related to Devops early on. It is necessary to highlight the gravity of the change: Devops is not merely a product or a toolset. Neither is it a team or a role. The entire IT organization should adapt to the new culture and processes that follow. Another takeaway of the study suggests dropping the term Devops at first where reluctance and biases are met. It is easier to talk about the benefits of eg. collaboration and automation before mentioning Devops. Finally, training and educating the entire IT organization across the board early on helps to combat the awareness challenge. Finding the key sponsors inside teams is also important since these people are very efficient in spreading knowledge in teams.

7.2.2 Building commitment and trust

An organizational change can rarely happen without the full commitment of management and teams. In getting management aboard, the findings conclude that identifying the type of management profiles, whether they are tech-savvy, open to ideas or teflon, helps choosing a strategy to onboard management. It is also a question of attitude: Devops needs to be communicated as a major change and not just a R&D endeavor. Similar to management, teams need to be engaged as well. The change of roles and new skill requirements should be communicated as a factor of increased job satisfaction as well as an increase to employee market value, considering that there is an increasing demand of “Devops skills” in the job

market. Finally, promoting trust and autonomy both horizontally across teams and vertically with management is important because many of the agreement-based models such as rigid change management processes and CAB meetings are not necessarily valid anymore. Devops needs organizational flexibility and velocity in decision-making.

7.2.3 Establishing common ways of working and leading by example

The main challenge with Devops technology is applying it correctly across the organization without fragmentation or varying practices and ways of working in teams. Highlighting the responsibility of single developers in playing by the rules (eg. writing automated tests) and leading by example was found efficient in establishing common principles. To further prevent fragmentation, being as transparent as possible helps to prevent reinventing the wheel with most technologies. This entails source code repositories and version control visible to everybody, with the possibility of everyone making changes to the code as well as common infrastructure and shared communication channels such as Slack, which enables topic-based conversation across teams. Where special technology is required, such as legacy systems, the application can be split up into separate services and further virtualizing them. Abstracting technology layers as far as possible is recommended in order to maintain a common architecture, with the help of containerization tools such as Docker. The challenge with the maturity of "Devops tools" such as configuration management tools can be overcome by keeping the tools interchangeable in order to prevent vendor-lock.

7.2.4 Ensuring the flexibility of the organization

As mentioned several times before, Devops needs a certain amount of flexibility in an organization to succeed. The findings of this study suggest that starting a Devops initiative with a quick proof of concept is essential to Devops success. Not only does it create traction in the rest of the organization by being able to show early wins, decisions can be made quickly and flexibly, leaving the rest of the organization to follow when the "coast is clear". Flexibility is needed also in the Software development mode the company uses: overcoming this challenge entails ensuring that the development mode fits into the Continuous Delivery approach with a goal of releasing features as soon as they are ready (instead of waiting for the sprint to finish). With change management flexibility is needed: a rigid RFC process with CAB meetings for instance, might need some adaptation to get the most out of Devops. For example, the Domain CAB –mentality used at YLE is a way to overcome the challenge. Finally, adding informal time spent together with teams and co-location improves communication.

7.3 Limitations and suggestions for further research

The study is limited to only nine industry experts. Although the interviewees all have considerable experience in this field, the amount of interviewees is quite small. Moreover, all the interviews were done in a Finnish context. A large part of Devops is connected to company culture. Therefore, the Finnish company culture might influence culture in the targeted companies and might have resulted in somewhat different implications if the study had been conducted in a different country. The lack of academic research in the field of Devops had an effect on the literature review and therefore many of the sources used in especially the literature review of this study were non-academic.

Although the results of this study are not directly generalizable, they do provide implications of what the main challenges of adapting a Devops approach to Software Development and Operations might be. In addition, the means to overcome these challenges are based on the experiences of the interviewees.

For further research I suggest expanding the amount of respondents and carrying the research out as a quantitative analysis. A larger research material could give a better overall picture of the challenges and finding out which of them are the most common ones. Moreover, a study carried out at a later point of time might give a more mature picture of Devops.

7.4 Own reflections

A new concept such as Devops is always met with a certain amount of skepticism in the beginning. For me, Devops seemed like a great suggestion for improvement of rigid processes in an organization, although grasping the concept in its entirety took a considerable amount of time. It wasn't until the very final interview with Klemetti when I thought I knew enough about Devops to actually write a thesis about it. Bearing this in mind, it's no wonder that understanding Devops also proved to be one of the main challenges in the findings.

References

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). Agile software development methods. Review and analysis. Espoo: VTT.

Ambler, S. (2002). Agile/Lean Documentation: Strategies for Agile Software Development. Online. Available at: http://agilemodeling.com/essays/agileDocumentation.htm [11.04.2016]

Anderson, D. (2010). Kanban Successful Evolutionary Change for your Technology Business. Sequim, Washington: Blue Hole Press.

Atlassian, A brief introduction to Kanban. Online. Available at: https://www.atlassian.com/agile/kanban [11.04.2016]

Azoff, M (2011) Devops: Advances in Release Management and Automation. Ovum. Online. Available at: http://electric-cloud.co.jp/wp-content/uploads/2014/06/EC-IAR_Ovum-DevOps.pdf [09.04.2016]

Basil, V. R., & Turner, A. J. (1975). Iterative enhancement: A practical technique for software development. Software Engineering, IEEE Transactions, (4), 390-396.

Bass, L., Weber, I., & Zhu, L. (2015). DevOps: A Software Architect's Perspective. Boston: Addison-Wesley Professional.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M. & Kern, J. (2001). Manifesto for agile software development.

Benington, H. D. (1983). Production of large computer programs. IEEE Annals of the History of Computing, (4), 350-361.

Bloomberg, J. (2015) DevOps and ITIL: Friends Or Enemies? Forbes article. Online. Available at: http://www.forbes.com/sites/jasonbloomberg/2015/11/13/devops-and-itil-friends-or-enemies [04.04.2016]

Boehm, B. W. (1988). A spiral model of software development and enhancement. Computer, 21(5), 61-72.

Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation computer systems, 25(6), 599-616.

CA Technologies (2014) TechInsights Report: What Smart Businesses Know About DevOps. Online. Available at: http://www.ca.com/us/~/media/Files/whitepapers/techinsights-report-what-smart-businesses-know-about-devops.pdf [09.04.2016]

CenturyLink (2013) What is NoOps Anyhow. Online. Available at: https://www.ctl.io/developers/blog/post/what-is-noops-anyhow [09.04.2016]

Cervone, H. F. (2010). An overview of virtual and cloud computing. OCLC Systems & Services: International digital library perspectives, 26(3), 162-165.

Chacon, S., & Straub, B. (2014). Pro git. New York City: Apress.

Cockburn, A. (2002). Agile software development. Boston: Addison-Wesley.

Commitcom (2014), Aktia käynnistää sovelluskehityksen DevOps-menetelmällä. Online. Available at: http://www.commitcom.fi/asiakkailta/?NewsAction=ShowNewsItem&ItemId=12043 [02.02.2016]

CSC Index. (1994) Critical Issues of Information Systems Management for 1994. Boston: CSC Index

Darnton, G., and Giacoletto, S. (1992) Information and IT Infrastructures. Information in the Enterprise: It's More Than Technology. Salem, Massachusetts: Digital Press, 273-294.

Debois, P. (2011). Devops: A software revolution in the making. Journal of Information Technology Management, 24(8), 3-39.

Dodson, J. (2013) A Brief and Incomplete History of Devops. Sonatype blog. Online. Available at: http://blog.sonatype.com/2013/07/a-brief-and-incomplete-history-of-devops [11.04.2016]

Dubois, A., & Gadde, L. (2002). Systematic combining: An abductive approach to case research. Journal Of Business Research, 55 (7), 553-560.

Earnshaw, A (2013) What is a Devops Engineer. Online. Available at https://puppet.com/blog/what-a-devops-engineer [09.04.2016]

Eficode, Devops Quick Guide. Online. Available at http://devops-guide.instapage.com/ [20.04.2016]

Elgan, M (2013) DevOps Is Just the (Necessary) Consumerization of Software Development. CA Technologies Blog. Online. Available at: http://54.174.189.58/2013/09/27/devops-is-just-the-necessary-consumerization-of-software-development/ [11.04.2016]

Eskola, J., & Suoranta, J. (1991). Johdatus laadulliseen tutkimukseen. Rovaniemi: Lapin Yliopisto.

Everyday Kanban, What is Kanban? Online. Available at: http://www.everydaykanban.com/what-is-kanban/ [11.04.2016]

Gartner (2015). Gartner Hype Cycle for Enterprise Architecture 2015. Online. Available at: https://www.gartner.com/doc/3103125/hype-cycle-enterprise-architecture-[01.02.2016]

Gartner IT Glossary, IT Operations. Online. Available at: http://www.gartner.com/it-glossary/it-operations [11.04.2016]

Glass, R. (2001) Agile Versus Traditional: Make Love, Not War. Cutter IT Journal, 12-18.

Goldratt, E., & Cox, J. (2004). The goal. Great Barrington, MA: North River Press.

Goldratt, E. (1990). What is this thing called theory of constraints and how should it be implemented?. Croton-on-Hudson, N.Y.: North River Press.

Gomm, R., Hammersley, M., & Foster, P. (2009). Case study method. 234-259. Thousand Oaks: SAGE.

Gross, J., & McInnis, K. (2003). Kanban made simple. New York: AMACOM.

Grossman, R. B., & Packer, M. B. (1989). "Betting the Business": Strategic Programs to Rebuild Core Information Systems. Office Technology and People, 5(4), 235-243.

Grover, V. J. T. C. T., Teng, J. T. C., & Fiedler, K. D. (1993). Information technology enabled business process redesign: an integrated planning framework. Omega, 21(4), 433-447.

Hendren, J. The problem with Defining Devops. Upguard blog. Online. Available at: https://www.upguard.com/blog/the-problem-with-defining-devops [03.04.2016]

Highsmith, J., & Cockburn, A. (2001). Agile software development: The business of innovation. Computer, 34(9), 120-127.

Hirsjärvi, S., Remes, P., & Sajavaara, P. (2009). Tutki ja kirjoita (13th ed.). Helsinki: Tammi.

Hummer, W., Rosenberg, F., Oliveira, F., & Eilam, T. (2013). Testing idempotence for infrastructure as code. Middleware 2013, 368-388. Berlin Heidelberg: Springer.

Menzel, G (2015) DevOps - The Future of Application Lifecycle Automation. A Capgemini Architecture Whitepaper – 2nd Edition

Halkosaari, L. (2016) Interview at Schibster. Helsinki 19.2.2016.

Humble, J. (2012) There is No Such Thing as a "Devops Team". Thoughtworks blog. Online. Available at: https://www.thoughtworks.com/insights/blog/there-no-such-thing-devops-team [09.04.2016]

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

Hüttermann, M. (2012). DevOps for developers. New York City: Apress.

Highsmith, J. A. (2002). Agile software development ecosystems (Vol. 13). Boston: Addison-Wesley Professional.

Highsmith, J. (2013). Adaptive software development: a collaborative approach to managing complex systems. Boston: Addison-Wesley.

Hin, W. (2006, November). Future implementation and integration of agile methods in software development and testing. Innovations in Information Technology, 1-5. IEEE.

Humble, J., & Molesky, J. (2011). Why enterprises must adopt devops to enable continuous delivery. Cutter IT Journal, 24(8), 6.

Kanban blog. What is Kanban? Online. Available at: http://kanbanblog.com/explained/ [11.04.2016]

Keen, P. G. W. (1995) Every Manager's Guide to Information Technology, Second Edition. Boston: Harvard Business School Press.

Koskinen, I., Alasuutari, P., & Peltonen, T. (2005). Laadulliset menetelmät kauppatieteissä. Tampere: Vastapaino.

Leffingwell, D. (2007). Scaling software agility. Upper Saddle River, NJ: Addison-Wesley.

Levitan, J. (2016) Overcoming the challenges to Devops. TriNimbus blog. Online. Available at: http://www.trinimbus.com/blog/overcoming-the-challenges-to-Devops/ [01.04.2016]

Ma, J. (2014) 5 Things You Need to Know When Implementing DevOps. New Relic blog. Online. Available at: https://blog.newrelic.com/2014/10/20/Devops-ebook-challenges/ [09.04.2016]

Marie, T. (2014) What is DevOps? Even the Thought Leaders Can't Agree! Devops Zone. Online. Available at: https://dzone.com/articles/what-devops-even-thought [09.04.2016]

Mayer, T. (2009). Scrum: A New Way of Thinking. Agile Anarchy. Online. Available at: https://agileanarchy.wordpress.com/scrum-a-new-way-of-thinking/ [09.04.2016]

McKay, D. B., Brockway D.W. (1989). Building I/T Infrastructure for the 1990s. Stage by Stage (Nolan Norton and Company), 9(3), 1-11.

Miles, M., & Huberman, A. (1994). Qualitative data analysis an expanded sourcebook, 352. Thousand Oaks: SAGE.

Moss-Bolaños, A. (2016) Top Organizational Challenges For Implementing DevOps and How To Overcome Them. Xenialabs Blog. Online. Available at: http://blog.xebialabs.com/2016/01/20/top-organizational-challenges-implementing-Devops-overcome/ [04-04-2016]

New Relic (2014) Kickstarting DevOps. New Relic. Online. Available at: https://blog.logentries.com/2014/10/6-challenges-facing-Devops-and-operations-teams-in-2015/ [09.04.2016]

Niederman, F.; Brancheau, J. C.; and Wetherbe, J. C. (1991) Information Systems Management Issues for the 1990's. MIS Quarterly, 15(4), 475-495.

NIST (2011). The NIST Definition of Cloud Computing. Recommendations of the National Institute of Standards and Technology. Gaithersburg: Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology.

Null, C. (2015) 10 Companies Killing it at Devops. Techbeacon blog. Online. Available at: http://techbeacon.com/10-companies-killing-it-devops [11.04.2016]

Nygard, M. (2007). Release it!: design and deploy production-ready software. Pragmatic Bookshelf. Boston: Addison-Wesley.

Ohno, T. (1988). Toyota production system. Cambridge, Mass.: Productivity Press.

Paul, F. (2014) The Incredible True Story of How DevOps Got Its Name. New Relic Blog. Online. Available at: https://blog.newrelic.com/2014/05/16/devops-name/ [09.04.2016]

PE International. (1995) Justifying Infrastructure Investment. IT Management Programme Report. Egham, United Kingdom: Centre for Management Research, PE International.

Poppendieck, M., & Poppendieck, T. (2003). Lean Software Development: An Agile Toolkit: An Agile Toolkit. Boston: Addison-Wesley.

Puppet Labs (2013). State of Devops 2013 Report. IT Revolution Press. Online. Available at: http://info.puppetlabs.com/2013-state-of-devops-report [11.04.2016]

Puppet Labs (2015). State of Devops 2015 Report. IT Revolution Press. Online. Available at: https://puppet.com/resources/white-paper/2015-state-of-devops-report. [11.04.2016]

Rapaport, R (2014), A Short History of DevOps. Online. Available at: http://rewrite.ca.com/us/articles/devops/a-short-history-of-devops.html [11.04.2016]

Riley, C. (2014a) How to Keep CALMS and Release More! Logentries blog. Online. Available at: https://blog.logentries.com/2014/10/how-to-keep-calms-and-release-more/ [09.04.2016]

Riley, C (2014b) 6 Challenges Facing DevOps and Operations Teams in 2015. Logentries blog. Online. Available at: https://blog.logentries.com/2014/10/6-challenges-facing-Devops-and-operations-teams-in-2015/ [12.04.2016]

Royce, W. W. (1970, August). Managing the development of large software systems. Proceedings of IEEE WESCON, 26(8), 1-9.

Schwaber, K. (2004). Agile project management with Scrum. Redmond, Washington: Microsoft Press.

Sitakange, J. (2014) Infrastructure as Code: A Reason to Smile. Thoughtworks blog. Online. Available at: https://www.thoughtworks.com/insights/blog/infrastructure-code-reason-smile, [11.04.2016]

Spikelab. The word Devops and a marketing problem. Online. Available at: http://www.spikelab.org/blog/the-word-devops-and-a-marketing-problem.html [09.04.2016]

Steinberg R., (2011) ITIL Service Operation. London: TSO.

Sussna, J. (2013) Cloud and DevOps: A Marriage Made in Heaven. Online. Available at: http://www.infoq.com/articles/cloud-and-devops [11.04.2016]

Techbeacon (2015). Is DevOps at odds with change management? Techbeacon blog. Online. Available at: http://techbeacon.com/devops-odds-change-management [04.04.2016]

The Agile Admin (2011). What is Devops. Online. Available at https://theagileadmin.com/what-is-Devops/ [09.04.2016]

Tuomi, J., & Sarajärvi, A. (2006). Laadullinen tutkimus ja sisällönanalyysi. Jyväskylä: Tammi.

Turk, D., France, R., Rumpe, B. (2002) Agile Software Processes: Principles, Assumptions and Limitations. Technical Report. Colorado State University.

Turnbull, P. D. (1991) Effective Investments in Information Infrastructures. Information and Software Technology, 33(3), 191-199.

Turnbull, J. (2010) What Devops Means To Me… Online. Available at: http://kartar.net/2010/02/what-devops-means-to-me.../ [11.04.2016]

Upguard. DevOps and Kanban - Match Made in Heaven. Upguard blog. Available at: https://www.upguard.com/blog/Devops-kanban-match-heaven, [11.04.2016]

Ursin, N. (2016) Interview at Alma Talent. Helsinki 23.2.2016.

Uusitalo, H. (1991). Tiede, tutkimus ja tutkielma. Porvoo: WSOY.

van Ommeren, E., van Doorn, M., Dial, J., & van Herpen, D. DESIGN TO DISRUPT.

Webopedia, Consumeration of IT. Online. Available at: http://www.webopedia.com/TERM/C/consumerization_of_it.html [11.04.2016]

Weill, P. (1993) The Role and Value of Information Technology Infrastructure: Some Empirical Observations, Strategic Information Technology Management: Perspectives on Organizational Growth and Competitive Advantage. Middleton, Pennsylvania: Idea Group Publishing.

Weill, P. (1991) The information technology payoff: implications for investment appraisal. Australian Accounting Review, 1(1), 2-11.

Weill, P., & Broadbent, M. (1994) Infrastructure Goes Industry Specific. MIS(July), 35-39.

Wilsenach, R. (2015) DevOpsCulture. Online. Available at: http://martinfowler.com/bliki/DevOpsCulture.html [09.04.2016]

Womack, J. P., Jones, D. T., & Roos, D. (1990). Machine that changed the world. New York City: Simon and Schuster