Transitioning towards continuous development within an established business organization

Atte Virtanen

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.
Espoo 17.11.2017

Thesis supervisor:

Doc. Kalevi Kilkki

Thesis advisors:

M.Sc. Tatu Sipilä

M.Sc. Antti Rajala

Aalto University
School of Electrical Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Atte Virtanen

Title: Transitioning towards continuous development within an established business organization

Date: 17.11.2017 Language: English Number of pages: 7+55

Department of Communication and Networking

Professorship: Network Economics

Supervisor: Doc. Kalevi Kilkki

Advisors: M.Sc. Antti Rajala, M.Sc. Tatu Sipilä

Software development today has rapidly developed into a significant part of business and its value creation chain. Increasingly more stakeholders within an organization are tied in with more frequent software releases. This has driven organizations to adapt to more flexible and continuous software development methods.

This Master's Thesis addresses the challenges, advantages and disadvantages associated in shifting an organization's software development culture towards that of continuous development. The specific type of continuous development within this research considers the new software development culture of DevOps. DevOps is seen as a fundamental change in the IT world today for the transition towards continuous software development, where dedication is given to the successful collaboration between development and operations.

The aim of this research is to discover the vastness of attempting to change an organizational culture for an improved and modern software development process for all stakeholders involved. Furthermore, this research attempts to provide the organization at hand with information on how and where to begin initiating the required changes. New cloud computing technologies have enabled development teams to become less dependent on companies' traditional IT departments.

The research is conducted via literature review and the data collected through interviews with employees of the organization attempting to shift towards continuous development. Further information is gathered through three case studies of other companies that have successfully undergone a transition towards continuous development and DevOps.

Keywords: devops, agile, lean, cloud computing, continuous development

AALTO YLIOPISTO | DIPLOMITYÖN
SÄHKÖTEKNIIKAN KORKEAKOULU | TIIVISTELMÄ

Tekijä: Atte Virtanen

Työn nimi: Jatkuvaan ohjelmistokehitykseen siirtyminen vakiintuneessa organisaatiossa

Päivämäärä: 17.11.2017 Kieli: Englanti Sivumäärä: 7+55

Tietoliikenne- ja tietoverkkotekniikan laitos

Professuuri: Tietoverkkotalous

Työn valvoja: Dos. Kalevi Kilkki

Työn ohjaaja: KTM Antti Rajala, DI Tatu Sipilä

Tänä päivänä ohjelmistokehityksestä on nopeasti muodostunut merkittävä osa liiketoimintaa ja sen lisäarvon tuottamista. Yhtiöiden sisällä yhä useampi sidosryhmä on osallisena yhä useammin toistuvissa ohjelmistojulkaisuissa. Tästä johtuen yritykset ovat joutuneet sopeutumaan joustaviin ja jatkuviin tapoihin kehittää ohjelmistoa.

Tämä diplomityö tutkii yhtiön jatkuvaan ohjelmistokehitykseen siirtymisen haasteita, hyötyjä ja haittoja. Jatkuvan ohjelmistokehityksen tyyppi, jota tässä työssä tutkitaan, on nimeltään DevOps. DevOps:ia pidetään keskeisenä muutoksena nykypäivän IT-alalla jatkuvaan kehitykseen siirtymisessä. Sen pääpiirteeksi koetaan sulava yhteistyö kehityksen ja ylläpidon välillä.

Tämän tutkimuksen tavoite on selvittää yrityksen ohjelmistokehityksen muuttamisen laajuuden ottamalla samalla sen kaikki sidosryhmät huomioon. Lisäksi tämä tutkimus pyrkii tuottamaan yritykselle, jolle tutkimus tehdään, lisätietoa siitä miten tarvittavat muutokset voidaan käynnistää ja toteuttaa. Uudet pilvipalvelut ovat lisänneet kehityksen autonomisia työskentelytapoja ja vähentäneet heidän riippuvuuksia perinteisen IT-osaston toiminnollisuuksista.

Tutkimus toteutetaan kirjallisuuskatsauksen sekä yrityksen eri sidosryhmien haastattelujen kautta. Lisätietoa kerätään esimerkkien avulla, joissa tutkitaan kolmen eri yrityksen menestyksekkäitä siirtymisiä jatkuvan ohjelmistokehityksen pariin.

Avainsanat: devops, agiili, ketterä, pilvilaskenta, jatkuva ohjelmistokehitys

Preface

I would like to thank my supervisor Kalevi Kilkki for the support during the creation of this research.

Furthermore, I would like to thank my advisors Antti Rajala and Tatu Sipilä for helping me research the topic at Kesko.

Thanks Mom. Thanks Dad.

Otaniemi, 13.11.2017

Atte Virtanen

Table of contents

Abbreviations

CD – Continuous Delivery

CI – Continuous Integration

DOD – Definition of Done

DevOps – Development and Operations

HPE – Hewlett Packard Enterprise

IAAS – Infrastructure as a service

IT – Information technology

PAAS – Platform as a service

RAD – Rapid application development

SAAS – Software as a service

TPS – Toyota Production System

1 Introduction

1.1 Research topic justification

The foundations of this research lie in the current environment of Kesko's software development and my personal responsibilities within the organization. Kesko provides a set of development tools, which their development teams are free to use. The organization currently does not employ its own software developers. Instead, it supports standardized tools and processes under which development occurs.

Maintenance and support for the tools from Kesko's perspective are my personal responsibilities within the company. Providing tools for the development teams has therefore granted me access and a wide perspective on all of the development conducted for Kesko. Not only am I invested in the IT department of Kesko, but am also in constant contact with the developers developing for Kesko and employees from the business side. As different vendors require and use different tools and processes, the need for creating functional development guidelines that correspond more adequately to an evolving digital domain becomes evident on a daily basis.

Kesko is striving to adapt to these changing circumstances and improve upon its own software development environment. Bridging the gap between business and development is at the forefront, as currently the challenges lie within this collaboration.

Furthermore, I am personally keen on discovering how change occurs within a large organization. Especially within an organization undergoing a transition towards continuous development, where the goal is to be able to release faster, more often and more agilely.

1.2 Company background

Kesko, one of the largest multi-industry corporations in Finland has been steadily taking advances into the digital domain with a vast range of new digital services. The company's strategy has been to increase sales, market share, customer satisfactory and customer value through the expansion of their digital services.

As a conventional sales company, Kesko now faces the challenge of evolving its ways of working as it attempts to tackle the hurdles associated with a large firm attempting to incorporate efficient digital development into its portfolio. Management, development, operations and many other roles need to align mutually in their goals, targets and cooperation in order to create a functioning and coordinated platform for their digital services. These hurdles include aspects from technical solutions to deeply rooted cultural practices, all of which can significantly hinder digital project management processes.

Kesko’s three main business operations are grocery trade, technical trade and car sales. Advancing these portfolios further into the digital domain requires skillful technical expertise in software development and efficient execution. Kesko’s scattered digital development model has revolved mainly around outsourcing its software development. The approach towards outsourcing software development is not due to change instantly, as Kesko’s business model lies in the sales of consumer goods. However, as the portfolio around its digital services expands, so does the need to evolve Kesko’s own digital development culture to match the fast-paced markets it is involved in.

Kesko has its target set at pursuing new methods for its working culture, mainly within its own IT department and its dependencies, in order to develop an environment, which can produce high quality digital services faster and more efficiently. The lack of a precise and unified blueprint on how to execute digital projects currently prevents development optimization on an organizational level. Different types of development methods such as waterfall, agile or lean have been implemented within the development projects. However, complete guidelines on how the organization wishes development to be conducted are missing.

Developing consumer applications such as the K-Ruoka mobile application or building new online stores for Kesko’s grocery and hardware trade have given rise to shifting development from the outdated “waterfall” model to that of continuous development. Projects that are set to increase market share and business value in the long term require constant development. Therefore, the organizational environment is required to be in a state of supporting such development. Transitioning a software product from one department to another, for example from development to operations, should no longer be the only option.

Technological advances in cloud services have been an additional catalyst for advancing the methods of conducting software development. With the rise of cloud computing focusing on delivering infrastructure, platforms and software as a service, the burdens of personally having to build infrastructure and catering to scalability needs have been mitigated. Developers and in particular Kesko are in a position to take advantage of these technologies for optimizing development.

Kesko’s development structure consists of an organizational IT department supporting outsourced software development conducted by collaborating vendors. Improving this divide between the information technology department and software development is at the forefront of Kesko's plans. DevOps, the cooperation of development and operations, has been suggested as a potential model for Kesko's development guidelines.

This thesis sets out to investigate the problems in Kesko's software development process and to compare them to the practices related to DevOps. The aim is to gain an understanding into the aspects of how Kesko could implement DevOps and whether or not DevOps could solve problems currently present in software development. The research questions and scope presented in the following chapters define the goals of this research in more detail.

1.3 Research problem and questions

As Kesko is striving to improve upon its software development model and transition towards developing more continuously than before, the questions which this research attempts to answer are as follows:

RQ1: What are the problems in Kesko's software development process?

In order to create an analysis of how Kesko could change its software development process, the current problems need to be uncovered. This question is answered through interviews with employees working in Kesko's software development.

RQ2: What points can Kesko focus on in transitioning towards continuous development?

As a non-software development company the process of transitioning towards continuous development is not rapid. Furthermore, the creation of a new model is a tedious process during which the best practices for Kesko are developed. Therefore, this research attempts to answer the question of what Kesko can do now and what takes longer to establish.

RQ3: Could DevOps be adaptable?

After analyzing the current situation of the company's development model and making proposals on what it can implement now and in the future, the final question of this paper answers the question whether or not the culture of DevOps is suitable for Kesko to transition towards continuous development.

1.4 Research scope

The nature of DevOps and its implementation on an organizational scale require vast inputs from all aspects of the company. The working culture of DevOps affects everyone, from management to every team and employee. Tools, guidelines, new social norms and skills must all be developed and established from the ground up. The process of developing an organization's own DevOps culture does not have a finite ending, but instead focuses on continuously improving upon the current conditions. This does not happen overnight.

Due to the nature of DevOps being a newly considered model for Kesko, the scope of this thesis is limited to the key aspects of DevOps. The goal of this research is to identify the technological and cultural readiness needs at Kesko for deploying and building their own continuous software development. This thesis provides an insight into the current problems at Kesko, the required points of focus for transitioning from a conventional "waterfall" development model to an agile or lean method of working; specifically, the deployment of DevOps as a potential model. Furthermore, this paper aims to aid readers new to the topic to familiarize themselves with the relevant concepts and steps for DevOps and transitioning towards continuous development.

This thesis attempts to present the theories and aspects behind DevOps and place them into the context of Kesko. Companies and organizations differ vastly from each other, meaning that each firm that has successfully implemented DevOps has done so within their own limitations. DevOps can mean significantly contrasting things within different organizations, mainly due to contrasting business goals. Hence, this research aims to discover whether the best practices of DevOps are deployable at Kesko.

Finally, the purpose of this paper is not to create an immediately deployable model of DevOps for Kesko, but instead present the necessary aspects for being able to adapt it. A roadmap for the ideal state or a completed model of Kesko's DevOps culture does not suit the scope of this research. Instead, it aims at describing the key points of DevOps, reflecting them onto Kesko's current software development environment and propose points of emphasis for a potential Kesko DevOps model. However, this thesis does function as an introduction on the scale of implementing a revised working model, especially for employees and decision makers involved in the transitioning process.

1.5 Research structure

As its first chapter, the structure of this thesis begins with the presentation of the necessary background information on the topic at hand. Information on the author's relevance to the subject and the company in question are presented, closely followed by the research questions studied within this work.

The structure continues with chapter 2, which introduces the research methods used to conduct this study. Literature was utilized to discover the theorem relative to the topic. Data collecting was conducted via interviews with employees from the company who are associated with the topic.

Chapter 3 presents the theorem on software development, DevOps and cloud-computing capabilities.

Chapter 4 lists three example cases of how other companies have transferred to continuous development.

Chapter 5 posts the data and results gathered during the interviews. This chapter is followed by chapter 6, which analyses the results and proposes methods on how to begin developing a more modern software development model.

Chapter 7 discusses key points on what could be the ideal situation of the company in the future. The final chapter, chapter 8, summarizes the research conducted in this work.

The references are listed in chapter 9, which is conclusively followed by the appendix listing the questions asked during the interviews.

2 Research methods

This chapter describes the research methods of this thesis. It begins with the introduction of the literature related to this research and is followed by presenting the data collection process in the form of conducting interviews within the organization.

The approach for researching this topic commences with the investigation into the theories of software development and the theories of DevOps in particular. In order to place these theories into the context of Kesko, interviews were chosen as the most suitable form of data collection. The methods are justified by this research being qualitative in its nature.

For referencing purposes, the research is completed by analyzing how other companies have either fully implemented DevOps or transitioned from a traditional development model to a continuous development model. These data collecting methods were used in analyzing Kesko's current situation and drawing comparisons to the theorem and practical examples. Conclusively, this data was used to create results for Kesko's points of focus on transitioning towards DevOps.

2.1 Literature review

The literature review presents different software development models, such as the waterfall, agile and lean frameworks. Their general ideas and reasons for their application strive to introduce the reader into the different ways software development is managed.

The software development models are followed by introducing DevOps, which is the continuous development model Kesko wishes to adopt. DevOps is presented in detail, as it is the focus of this research.

Further introductions are made into the automation processes of software development, as in the form of presenting the theories of the continuous deployment pipeline and cloud-computing possibilities. These chapters are justified by modern software development relying on these technological aspects.

The literature review concludes with information on common organizational challenges caused by organizational culture and structure. These theories are closely linked to a transition towards a different development model, as changes within large organizations are complex.

2.2 Interviews

2.2.1 Interviewees and company context

The empirical study of this research is conducted through the analysis of 11 interviews, lasting roughly an hour each, with people related to Kesko's IT department. The interviewees were selected from different branches, teams, vendors, positions and operations to attain a comprehensive coverage of Kesko's digital domain and a current understanding on what it means to establish a new culture in Kesko's own software development operations.

The group consisted of three employees from the business branch, five employees from the IT department and three software developers. Hence, not all participants in the interview process were directly from Kesko's IT department, but all had connections to it.

The business aspect of interviewees encompassed the three main business branches: the grocery trade department, hardware trade and supporting IT functions. Such a distinct division based on each industry was avoided for the IT perspective due to the complexity of one common IT department being involved in all digital development projects at Kesko. Instead, the interviewees were chosen based on the nature of their work: the group was comprised of project managers, IT architects, operations, testing and IT leadership.

The developer aspect of interviewees involved people from different software companies. All interviewed developers were working on diverse projects with altering methods and tools. This selection provided a wide range of information on the study at hand. The goal of selecting developers from different partnering companies was identical as with the other groups: gaining different perspectives from different corners of digital development within Kesko.

2.2.2 The purpose of the interviews

The interviews and their respective questions were devised with the purpose of gaining knowledge into the current state of Kesko's software development model. The primary part of the questionnaire focused on learning from the interviewee what he or she believes DevOps is and discovering whether people involved in the process of software development had any previous knowledge or understanding of DevOps. These questions differed between interviewees as the questions were chosen according to the interviewee's role within the organization. The subsequent part presented the interviewee with questions relating to the digital project environment at Kesko and the recognizable problems within it. The closing set of questions considered aspects of what the future of Kesko's software development model could withhold.

By reflecting the ideas and theorems behind software development, DevOps and Kesko on the interviewee, the questionnaire had three specific goals on which to construct this research:

- Obtain information on whether or not Kesko already has elements of the DevOps model built into its own process
- Understand Kesko's development environment
- Discover how much of DevOps could potentially be implemented at Kesko

The questionnaire is posted in the appendix.

3 Software development

Software development is conducted through different models, with each model carrying its unique advantages and disadvantages. The most traditional method of developing software is the “Waterfall” model, whereas today more modern approaches are agile and lean. Others such as the Rapid application development, or RAD, model [1] exist as well, however this chapter focuses on the more traditional approach and agile as they are the primary approaches relevant for Kesko. Additionally this chapter presents DevOps and cloud services in more detail.

Waterfall, lean and agile are frameworks for software development. These frameworks consider the methods and processes of how teams and projects conduct software development. They describe the dynamics within teams for creating software. DevOps on the other hand is not a framework for software development, but instead a culture of tying the software development frameworks into an efficient IT unit involving development and operations [2].

Therefore, in order to portray the culture of DevOps, it is fundamental to present the ideas of the different software development frameworks.

3.1 The Waterfall model

The waterfall model is a traditional type of software development process, which progresses logically from one phase to another after each completed phase [3]. Dr. Winston W. Royce introduced it in his paper *Managing the development of large software* systems in 1970 without naming it the waterfall model [3]. The model gained its name through the downward flow between phases. The typical stages of the waterfall model are presented in Figure 1 [3].

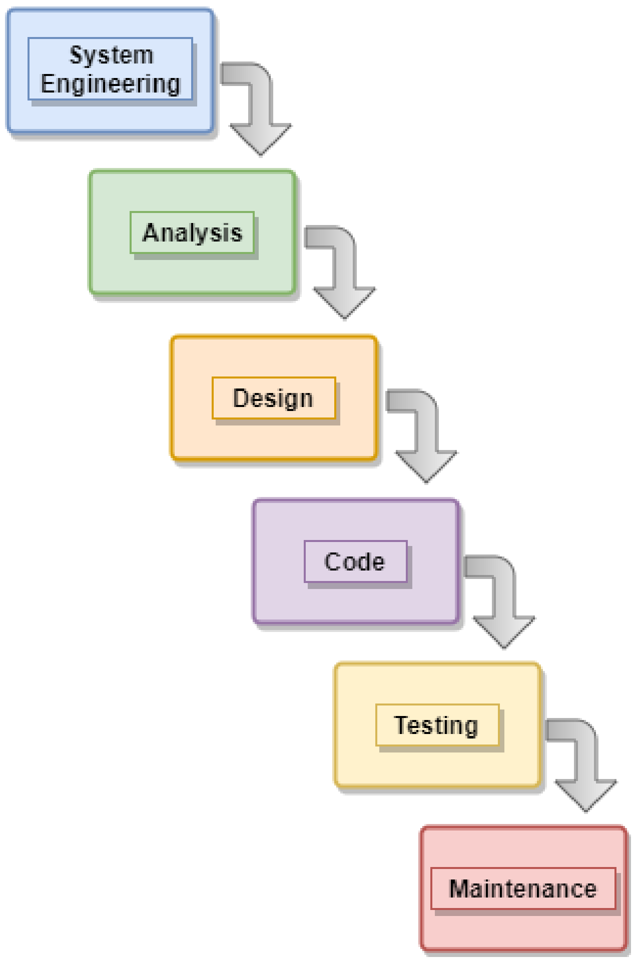

Figure 1. *The Waterfall model*

The development process beings with extensive planning on what the system and software requirements for the product should be. This phase is followed by analyzing and designing the user interface and other visuals, after which programming can begin. Finalized software is conclusively tested and placed into production. Eventually, the software is regarded as being complete and is finally transferred to maintenance. Due to the different phases being performed by different and specialized groups, the waterfall model obligates the organization’s structure to be strictly arranged [3]. Development processes are managed firmly [4].

At its core, the waterfall model attempts to produce a finished product systematically from beginning to end and eventually release it all at once. The most notable advantage of the waterfall model is being able to complete phases on schedule [4]. As every phase is transferred to the next one, each step has a set deadline [4]. Hence, the schedule is simple to follow, creating software on time. Due to the strict specifications at the initial planning stage of the project, documentation is well constructed throughout the development stage [3]. New project members can join the project with ease as the knowledge is found within the documentation [3]. This documentation creates the significant bridge for transferring knowledge of the project from one phase to another [4]. The waterfall model is most suitable for projects which can provide complete specifications at the earliest stage.

The theoretically pure waterfall model expects each phase to be perfected before moving on to the next [4]. In practice however, perfecting a phase is challenging [4]. Stages might not be completed on time, leading to the next phase having to wait or begin work with an incomplete product. This leads to slow and inefficient development. During the planning phase, not all specifications and future problems can be accounted for [4]. The waterfall model does not support adapting to feedback gained during development [3], causing potential rewriting of code [5]. Customers or users of the software might not know exactly what they want from the product. This can create an outdated and unusable product [6, p. 17]. Sudden changes or problems during any phase can significantly derail the project from its original schedule [4]. Furthermore, changes during development create costs, making the project flow over the planned budget [3]. As testing is conducted during the later stages, potential software problems are recognized late [5].

3.2 Lean and agile practices

Lean and agile are the modern approaches to software development, formed under the fast-paced software markets of today. The aim towards creating business value and delivering features based on customer wants has led to the development of more flexible software development methods. These allow for more freedom and adjustments during development. Their relevance in this research is based on the assumption that DevOps practices rely on lean and agile software development as a platform. [6, p. 36]

3.2.1 Lean development

Lean development derived originally from the manufacturing industry, where the car-manufacturing firm Toyota built its manufacturing line around just-in-time production. Toyota created their own Toyota Production System (TPS) [7]. The general thought behind lean development is the elimination of waste and unnecessary work during the development process in order to produce the most customer value. Tom and Mary Poppenieck introduced lean software development and broke it down into seven key principles. These principles are presented in Table 1 [8, p. 13-15].

Principle	Description
1. Eliminate waste	- Eliminating redundant code, features - Avoiding speed blocks created by bureaucracy, ineffective communication - Avoiding the attempt to do more than can be completed - Avoiding multitasking and fractionally completed work - Eliminating every unnecessary aspect
2. Build quality in	- Focus on assuring quality software from the start - Building quality in from the start avoids waste at finalizing stages - Pair programming: two developers simultaneously programming; high error avoidance - Test Driven Development (TDD): tests and test conditions are written before code - Incremental and frequent code integration - Test automation
3. Create knowledge	- Quality software is based on knowledge and experience of the product - Pair programming enriches more than one developer - Documentation - Code which is well commented - Code is reviewed - Sharing knowledge within team
4. Defer commitment	- Making decisions at the latest possible, but safe, stage - Especially important for permanent decisions - Creates flexibility in software product
5. Deliver fast	- Delivering quickly and collecting feedback to improve - Time to market can create competitive advantage - Avoiding monoliths - Keep things simple
6. Respect people	- Everyone, regardless of position, should be treated equally and respectfully - Transfer responsibility to workers - Empowerment without sacrificing control
7. Optimize the whole	- Optimize complete value stream, from feedback to release - Organize teams around product, not around expertise

Table 1. *The key principles of lean software development*

Eliminating unnecessary work and inefficient methods of working are, through its seven principles, at the essence of lean development. Learning what can be eliminated throughout the development cycle enhances the understanding of customer value for the specific product. Developers are empowered and trusted to conduct their work. Risks are minimized by making decisions at the latest possible time. [9]

The principles of lean as a process management method for development [10] lend themselves to the principles of agile that are presented in the following chapter.

3.2.2 Agile development

Whereas lean focuses on eliminating waste, agile software development places its emphasis on people with the most significant aspect being the end-user of the software. Agile stems from the fact that the challenge of software development is not being able to recognize the outcome of the produced code in advance. Therefore agile relies on the feedback provided by its end-users to meet the initially unknown requirements of the software. [10]

The Agile Manifesto describes the success metrics of agile as being able to provide working software through the collaboration with its end-users. People and their respective cooperation are valued more than processes and tools whilst being able to adapt to changes instead of attempting to stay fixed on a preset plan. [11]

Agile follows twelve principles that are presented in Table 2 [12].

Principle
Highest priority is customer satisfaction
Welcome changing requirements
Frequent delivery of software
Business people and developers cooperating daily
Build projects around motivated people
Face-to-face conversation is best
Progress measured by working software
Sustainable development pace
Continuous attention to technical excellence
Simplicity
Self-organizing teams
Regular reflection and adaptation

Table 2. *The principles of the Agile Manifesto*

The essence of agile's twelve principle is to be able to react to changes required by the end-user for providing the highest quality software. Releases are done iteratively and frequently. Development teams conduct work together at the same location and cooperate often with employees from the business side for the highest value creation chain. [12]

3.3 Continuous deployment pipeline

The continuous deployment pipeline is the set of automated processes that enable the fast and efficient deployment of new software features into production. The pipeline is a fundamental aspect in lean and agile software development, but also in DevOps' automation culture. The general stages within such a pipeline consist of continuous integration and continuous delivery. The goal of these stages is to authenticate the software at different phases and establish feedback to the developers, whilst doing so on a consistent basis. [13]

The following two sections on continuous integration and continuous delivery depict the components of the automation process of building software. They are an essential part of DevOps, however as the approach of this research limits itself to DevOps within a large organization and is directed at all employees, these sections do not delve into the specifics of how to construct CI and CD from a developer's point of view.

3.3.1 Continuous integration

Continuous integration, or CI, is the process of submitting small changes of code to the main piece of code, called the mainline, on a constant basis, where the time interval between new commits is short. Multiple developers working on a piece of software periodically submit minor changes to the main piece of software. The purpose of constantly applying small adjustments is to maintain improved control of the mainline. Larger changes conducted less frequently could cause compatibility and build issues. Additionally, in the case of run or test errors after a commit, the rollback to a previous version is significantly simpler and the effects on the main piece of code are mitigated. [14]

Programmers make a copy of the mainline to their system, add their changes to the program and commit these changes back to the mainline. Once a new commit to the mainline is made, an automated commit build, which includes tests, is triggered which confirms the changes' compliance within the main piece of code. Bugs and errors are discovered through these automation steps. [14]

The advantage of CI is that it automates the integration of code changes. Linking all required files and conducting appropriate tests is much faster through an automated CI pipeline [14]. Incremental code changes further enhance the creation of software [14]. These aspects reduce risks by making problems become easier and faster to recognize since developers receive most of the feedback of their code at this stage [13].

3.3.2 Continuous delivery

Once the incremental changes or new features have been completed, integrated and tested, they can be deployed to production. Continuous delivery, or CD, consists of automated processes that enable the possibility of being able to deploy the changes into production at any time [15]. Changes are delivered to an environment, which mirrors the production environment itself to ensure that the changes will also function in production [15].

Continuous delivery is the final step in continuous deployment. Deployment into production might not be conducted as often as continuous delivery. Continuous delivery warrants the possibility of rolling out at any time [15]. Continuous deployment considers releasing new features to the eventual end-user, whereas continuous delivery ensures that the release can be conducted without failures [16]. Organizations schedule software releases with marketing and other business functions. Releasing new features to end-users often requires detailed scheduling, therefore constant rollouts are not always granted. Continuous deployment enables the ease of releasing [16].

The continuous deployment pipeline, which places software changes into production, reduces manual labor as much as possible [16]. Incremental continuous deployments mitigate risks compared to large releases [16].

3.4 DevOps

3.4.1 What is DevOps

Development and Operations, or DevOps, is regarded as the most advanced software development culture based on the lean and agile methodologies. It incorporates all the tools of lean and agile software development into an approach striving for a cultural change between development and IT operations. DevOps attempts to combine multiple roles and positions during development into an efficiently cooperative and autonomous team. [6, p. 6]

As the name, Development and Operations, implies, the aim is to strive for the collaboration of different roles within the software development process, such as developers and system operators. The justification for the need of more collaboration lies in the difference between the goals of development and operations [17]. Developers, much like business, strive for creating value to the customer. Whereas operations' focus is on establishing and maintaining a reliable system [17]. Furthermore, the aim is to eliminate barriers between existing functions, such as development and operations, for more efficient cooperation and ultimately faster and more concise software releases. In the technical scope, this means improvements in the time to market, time between bug fixes, time for recovery from failure and the quality of released software for greater customer satisfaction.

DevOps ingrains all of the benefits and practices of agile software development and eliminates handovers of a project between different teams [6, p. 91]. This is achieved by having the team originally responsible for creating the software take over the tasks needed further down the development line. Development teams become DevOps teams by creating, integrating, testing, building and maintaining their software. The team building a product becomes responsible for running it [18]. Such is the DevOps mantra: "you build it, you run it" [18]. The following chapters further demonstrate the aspects that define DevOps.

3.4.2 DevOps practices

The four key values of DevOps revolve around CAMS: culture, automation, measurement and sharing [19].

3.4.2.1 Culture

Creating a culture, which emphasizes communication and corporation, is at the core of DevOps. Silos and barriers within an organization cause unnecessary delays in information flow. These delays in return reduce work efficiency and produce wasted working hours [19]. A DevOps culture aims at removing such silos and encourages an environment with open discussion between employees and teams. Management plays a key role in creating such an environment, where open conversation is rewarded and employees are empowered [20, p. 110]. Therefore, a significant part of DevOps revolves around the cultural working aspect of development.

"DevOps is a human and management problem" [21].

"You can't directly change culture. But you can change behavior, and behavior becomes culture" – Lloyd Taylor VP Infrastructure, Ngmoco [21]

An essential part of an empowering working culture for the single employee or team member is knowledge of the true purpose of the ongoing work. Understanding the overall purpose has a significant effect on the motivation and innovation of the employee [20, p. 117]. This emphasizes the natural human urge of being genuinely part of something. Management is in a powerful position for generating a sense of purpose to its teams. As management becomes more aware of the overall situation within the organization and the dependencies between projects, the same understanding can be passed on to their teams. An organizational environment without silos provides the ability for empowered teams. Additionally it renders development teams with more responsibility.

3.4.2.2 Automation

DevOps strives to automate steps in the software development process as much as possible. Creating automated tests for new builds and an automated deployment pipeline is at the forefront of DevOps automation. The purpose of such automation is the reduction in avoidable working hours for repeatable tasks that computers can process faster. Testing automation is also more reliable than human testing. Thorough tests provide reliable data that can improve decision-making checkpoints. Furthermore, testing and deployment should be easy and not a burden for the developer. Making testing and deployment effortless promotes the concept of releasing new software more often, a key approach for agile and lean development itself. Although constructing a fully automated development environment is an extensive and laborious task, it pays itself back by making the process of releasing software uncomplicated. [20, p. 76]

3.4.2.3 Monitoring

Knowing what to develop and understanding how to improve development is at the heart for creating high quality software. Hence, DevOps teams strive towards monitoring the performance of their systems and their software. Monitoring infrastructure is critical in maintaining system stability. DevOps strives for an environment where recovery from failures is fast. The complexity of interdependent systems is more controllable with better monitoring tools. [20, p. 91]

3.4.2.4 Sharing

Tied in with culture is DevOps’ sharing factor. Building an open and trusted working environment requires transparency within communication and collaboration. Knowledge and information is shared as much as possible to create an efficient working culture. A sharing culture can lead to finding solutions faster and avoids conducting the same work twice. Sharing can encompass not only single teams, but also organizations as a whole. [19]

3.4.3 Culture and people

DevOps is as much of a technical concept as it is a cultural and people minded approach. The model promotes openness, communication, collaboration and trust between employees. Having clearly set goals and free information flow promotes a DevOps culture. Ideally, communication and information flow is never a barrier during development.

Changing organizational culture is crucial to having a DevOps mindset. However, understanding how to change existing culture begins by recognizing the culture currently in progress within the organization. In his study on information flow, Ron Westrum divides organizational culture into three distinct categories [22] presented in Table 3.

Category	**Information flow**	**Silos**
Pathological	Non-existent	Full
Bureaucratic	Only within silos	Departments are silos
Generative	Full	Non-existent

Table 3. *The categories of organizational culture*

In pathological culture, information flow between employees is highly biased or misleading. The reasoning for this claimed to lie in the human nature of wanting to make things appear better, especially for themselves. Therefore, information is not passed on at all.

A bureaucratic culture is considered to promote information flow only within organizational silos themselves, but not between different "departments" as Westrum claims. Each silo acts according to the rules developed within each silo.

The generative organizational culture is the most open, as the focus lies only on the goals of the organization.

The Westrum model is a polarization of real world organizational cultures. However, it demonstrates the direction a company needs for its working culture. In order to adapt DevOps, the company must continually shift towards a generative working culture.

Efficiently functioning teams evolve over time as trust within the team develops. Such teams embody the idea of open collaboration, which in turn builds trust.

A common location for the team members improves upon cooperation. Being able to communicate in the same space with each other makes problem solving faster.

An organization can consider culture already at the hiring stage. Employing the correct people can have a significant effect on the development of openness and cooperation.

3.4.4 Benefits of DevOps

More digital projects lead to more complexity within an environment without a unified set of tools and practices. DevOps guidelines strive towards simplifying the environment. The goal is to be able to focus more on the creation of innovative features and products instead of having to concentrate purely on maintaining the stability of existing systems [23]. Development therefore gains the opportunity to become continuous. Cross-functional and autonomous teams responsible for one product are able to solve problems faster without any previous dependencies from other teams or business functions [23].

Teams with a higher degree of freedom and responsibility become more involved in the company's identity [23]. As a result, the possibility of growth in work satisfaction and productivity for employees and team members increases [23]. Furthermore, a high degree of collaboration within a DevOps culture aids in removing silos between teams and barriers [24].

Quantitative benefits of DevOps have been studied by organizations such as puppet. Their annual surveys on DevOps within the IT industry present the development direction of DevOps. The annual State of DevOps Report by puppet and DORA for 2016 presents a vast array of benefits from DevOps, of which the most notable claims are listed in Table 4 [25].

"High-performing organizations decisively outperform their lower-performing peers. They deploy 200 times more frequently, with 2,555 times faster lead times, recover 24 times faster, and have three times lower change failure rates."
"High-performing organizations spend 22 percent less time on unplanned work and rework. They are able to spend 29 percent more time on new work, such as new features or code."
"Undertaking a technology transformation initiative can produce sizeable returns for any organization."
"The long-term value of an enterprise is not captured by the value of its products and intellectual property, but rather by its ability to continuously increase the value it provides to customers—and to create new customers—through innovation."

Table 4. *Benefits of DevOps according to puppet and DORA*

The ultimate goal of DevOps is for the organization to improve its competitiveness by being able to enhance the organization as a whole and not only improving a single silo [26].

3.4.5 Problems in implementing DevOps

The most significant hurdle in implementing DevOps is the cultural shift from removing the silos between development and operations. The two roles either function in unison or are embedded within the team. [27]

A further problem is the required transfer within the organization's digital domain of old legacy systems to modern microservices. However, the scope of this research limits itself to mentioning this architectural shift. [27]

Another problem in implementing DevOps is the resistance for change. [20, p. 119-120].

3.4.6 DevOps with agile or lean

DevOps incorporates all of the best practices from lean and agile software development. Lean and agile consider the dynamics within teams: how they function as an effective unit to produce quality software as fast as possible. Successful development teams create high quality code whilst using integration and deployment automation as much as possible. Much dedication is given to communication and collaboration within the team. The lean and agile frameworks focus on the teams themselves. [28]

DevOps however, as mentioned earlier, is not a development framework but rather a culture, which attempts to bond the teams into the complete IT organization. Emphasis is placed on developers collaborating with operations and vice versa. The technical tools are present for being able to release faster with shorter feedback loops. [29]

A distinction must be made: with companies purely producing software without any overhead IT operations, the DevOps culture becomes ingrained into the teams themselves. The team is responsible for development and operations. This means handovers to operations do not occur. However, for companies where IT operations and maintenance departments exist, the DevOps culture becomes the collaboration between the development team and operations team. [28]

The essence of DevOps is not a strict set of rules. Instead, DevOps is the cultural change between different IT functions enabling an organization to move towards continuous development. DevOps represents the next step for lean and agile frameworks. DevOps requires lean and agile frameworks as its base to be successful. It is not a separate framework by itself. Therefore, DevOps culture is most suited for agile software development. [28]

3.4.7 DevOps and cloud computing

The transition towards an automated DevOps culture is aided with the development of open-source and commercial cloud computing opportunities. Commercially available tools such as infrastructure-as-a-service (IAAS), platform-as-a-service (PAAS), software-as-a-service (SAAS) and serverless architecture all reduce the need for IT to maintain and support their own infrastructure and services. [30]

Development can be provided with traditional IT functions in a more agile way without the limitations set by maintaining one's own infrastructure. Infrastructure and other services required by development can be abstracted. [31, p. 285]

Different cloud services are presented in Table 5 [30].

Service	Description	Examples
IAAS	Provides computing infrastructure with virtual machines, run by customer	Amazon S3, OpenStack
PAAS	Provides development platform for application developers	Heroku, Microsoft Azure, RedHat OpenShift
SAAS	Provides software which is fully maintained by SAAS provider	Microsoft Office 365, Google Apps, Dropbox, Flowdock, Slack
Serverless architecture [32]	Provides computing infrastructure with virtual machines, run by provider	AWS Lambda

Table 5. *Cloud computing services*

The benefits of cloud computing have been the reason for their success, as for example gaining access to virtualized hardware and software resources immediately [33, p.178]. Cloud computing, especially IAAS, has the additional advantage of scaling resources according to usage [34, p. 25]. The pricing scheme of cloud computing, which is based on the usage of the services, mitigates the problem of maintaining own underused hardware [34, p. 25]. Such an automated infrastructure enables the self-serving on services for developers [31 p. 286-287]. The costs of maintaining servers, platforms and software are transferred to the cloud service provider.

DevOps benefits from the mitigations of obstacles set by the traditional IT hardware limitations. It can function without cloud services. However, the simple and quick implementation possibilities of cloud services and their on-demand capabilities make them an effective tool for improving the efficiency of development processes.

4 Company references and example cases

This chapter represents example cases of other companies that have undergone a similar transition as Kesko is attempting to achieve. The references selected are Solita, a large Finnish company and Hewlett Packard Enterprise. Solita was selected due the company being a software producer where DevOps has become an integral part of their operations. The large Finnish company is not a software producing organization, but requires software development to occur in order to increase sales and customer satisfaction. This company was chosen as an example case as they have been able to adapt DevOps keys into their development. Finally, as another large company Hewlett Packard Enterprise, or HPE, was able to transition towards continuous development as well.

4.1 Solita

Solita, a software consulting company, has adopted the four keys of DevOps successfully within their own organization. Being a software company, Solita has had strong a strong agile framework on which its teams operate. This example is based on a webinar conducted by Solita during the spring of 2017.

The culture of DevOps is at the core of DevOps at Solita. Silos within the organization have been minimized. If such silos arise, they are recognized and broken up. An environment without silos advocates the merging of development and operations. Furthermore, organizational bureaucracy does not create barriers for their operations.

Solita has created a model for their automated testing pipeline, which is simple to operate. This is accomplished by setting up their different development environments to resemble each other. The development, quality assurance and testing environments are all similar. In doing so, development and testing are conducted within copies of the real world environments, leading to minimized risks and fewer unknown problems further down the development process. Initially Solita configured their environments manually. Transferring software between the development, quality assurance and production environments would take hours due to the manual labor of configuring them.

Monitoring tools are in extensive use for different purposes throughout the development process, with the main objective being able to find root causes in cases of failure. Monitoring systems can show whether the servers are running correctly, but extensive monitoring with centralized log files can further aid in discovering what exactly might have gone wrong. The knowledge gained from monitoring is applicable for scheduling appropriate maintenance windows. The planning of future projects becomes easier as potential problems or opportunities are discovered at an earlier stage. Any suspicious activity is observed faster with a higher degree of monitoring. Solita uses monitoring not only for noticing anomalies within its systems, but also for developing its own processes: measure job queue management and apply metrics for code bug counts, evaluate average resolution times or determine customer satisfaction. Monitoring is used in a variety of ways within the DevOps culture of Solita.

In order to share information, Solita periodically arranges 30-minute knowledge sessions for its developers to discuss whatever topics they are working on at the time. Additionally, after failures have been resolved, Solita organizes sessions that it calls "post mortems": these sessions cover the reasons behind technical failures and their respective resolutions. These sessions emphasize the importance of sharing: knowledge and experience is shared among Solita's developers. For code sharing Solita uses the version control tool GitHub.

The mantra behind DevOps at Solita "Do things better – Do better things" justifies the implementation of the four keys of DevOps. Automation and monitoring reduces risks and simplifies the process of going into production, which in turn makes delivery continuous. Problems are easier to recognize and risks mitigated even further. The feedback from the value chain becomes more efficient as the feedback loop becomes shorter. The question whether a new feature or change increases the perceived value of a product is answered more effectively. An unsuccessful software change can quickly be reversed with an automated delivery pipeline.

4.2 A large Finnish company – Company A

Another large Finnish company operating on an international scale had previously been in a similar situation as Kesko: without any in-house developers, the company, Company A, was forced to outsource its software development. The teams were hired from software companies. This meant the outsourced developers had agile development methods built into their routine. In order to have these outsourced teams function as part of Company A, the decision was made to create its own set of guidelines and rules according to which the agile development teams were motivated to develop. Company A called it a "handbook" on how they conduct agile development. This example is based on a meeting with company A.

As with other large organizations, Company A was not able to adopt fully dedicated and autonomous teams developing in DevOps mode. Company A's development teams consist at its core of cross-functional team members, with the exception of infrastructure specialists being part of multiple teams for cost-efficiency reasons. However, Company A describes them as essentially being in DevOps mode due to the larger part of teams being cross-functional and having the capabilities of maintaining their own product.

Automation rules are part of Company A’s development guidelines. The continuous delivery pipelines are setup according to the rules made by them. Each team also releases software based on the same guidelines of Company A’s definition of done (DOD). Company A has created a template for development, which is used as the baseline of tools and practices for any new project.

Motivation and collaboration is caressed with multiple measures. Company A strives to accommodate its development teams into a single location as much as possible for the quickest flow of information and knowledge and improved collaboration. They additionally consider their developers more as artists than engineers. At the end of each three-month cycle, Company A organizes innovation sprints for two weeks for its developers. During these innovation sprints, the teams are encouraged to freely develop new ideas and innovations. The focus is to strengthen the teams’ commitment to their project and empower individuals to recognize and visualize their role within the organization. According to Company A, they are still working on improving the teams’ collaboration.

Company A has begun shifting towards hosting more in-house developers and has begun hiring. Its target is to continue employing developers until the goal of three quarters of the developers being their own developers is reached. However, according to Company A, this process is time consuming. Therefore, as teams grow, they are still hiring partially from outsourced companies with the addition of Company A being in charge who is hired from the vendor. The vendor suggests candidates and Company A selects whom they acquire.

The most significant obstacle for shifting towards continuous development under an agile and DevOps culture at Company A has been motivating and empowering developers to understand their role within the organization. According to Company A, it is challenging to get people more involved and attached to their work. This in turn creates a stronger sense for the reasoning behind their work. Another hurdle in implementing modern ways of development is training managers on setting goals for their teams more clearly. With Company A’s knowledge of their culture transformation today, it would have begun recruiting its own developers earlier.

4.3 Hewlett Packard Enterprise

4.3.1 DevOps at HPE

As a large organization, Hewlett Packard Enterprise (HPE) has adopted the DevOps culture successfully within their own framework. In 2015, their digital portfolio consisted of roughly 900 projects with 100 development teams and 1400 applications supported by their IT and operations department. This example is based on a DevOps seminar presentation given by HPE during the spring of 2017.

The question HPE attempted to answer was whether DevOps can work within their old and large organization. The answer to this question was positive: it is possible to apply DevOps within firms with high complexity and system dependencies. However, the extent of projects to which DevOps is applicable is limited to the nature of varying systems. Arne Luhrs, a senior system architect at HPE, described DevOps as "One size does not fit all" for their respective organization.

In order to reach the answer of how DevOps functions at HPE, HPE began by examining and discussing what DevOps was trying to solve at HPE. The outcome was the HPE DevOps manifesto, which portrayed the guidelines for culture and automation at HPE. The manifesto was built upon everything running through code, meaning that although developers had access, they were not able to make any changes without code to execute the changes. This culture eventually lead to more code reviews and collaboration within the development process.

HPE began their DevOps transformation in 2014 by applying the culture to the development of a mobile application first. DevOps is best suited for development projects working in an agile environment, under which the mobile application was being developed. The goal was to implement DevOps on a small scale first before expanding it to other digital assets.

4.3.2 Culture change at HPE

The first task was to drive the culture towards a higher degree of collaboration. With 45,000 professionals working in the IT department of HPE, a complete reorganization was not a viable solution. Arne Luhrs called this process "how to reorganize without changing the organization". HPE coined the term "ChatOps" for improving collaboration, which is the method of using a single chatroom tool for cooperation. Due to the complexity of HPE, not all teams have the opportunity to be located in the same room; hence, all communication is conducted through the chatroom. Furthermore, the same communication tool is used for integrating everything else: monitoring tools produce critical information, service tickets are created and team members use it as a social medium. All of this information is immediately visible within the chatrooms and the chatroom itself becomes the most important collaboration pipeline. Team members can not only discuss problems but also find relevant data, graphs and metrics all through one centralized tool. “Teams became teams around an asset” and employees themselves were seen more as people instead of a resource. Team members could share and learn from each other in a more efficient matter than before, which had previously been a system of e-mail communication. This communication tool significantly improved the speed and quality of collaboration.

4.3.3 Automation pipeline at HPE

The vast range of different digital projects does not allow for a unified set of automation tools. HPE was aware of this and shifted the focus from a tool discussion to a pipeline discussion, which lead the DevOps automation aspect to become the application of rules instead of tools. The rules were set for the execution within pipelines, where the pipelines were the set of tools used for each respective project. The goal was to create continuous delivery pipelines by automating as much as possible with code, whilst reducing unnecessary paper work and human decision-making steps wherever possible.

HPE created standard steps and non-negotiable points that had to be conducted within the pipelines. Different environments and testing had to be setup in a manner where automation could verify whether everything had been tested thoroughly, hence reducing risks and eliminating slow decision-making. User acceptance testing requires real log data and is automated if the possibility to do so exists. Environment changes or code commits always require automated code reviews before execution. Commits themselves trigger automated tests. Thousands of tests are triggered with each new commit at HPE, with the duration of the complete test cycle being 11-15 minutes. The aim with automation is to eliminate the necessity for requesting permission and making sure that "this has been tested in the correct environment and is verified".

4.3.4 Trust at HPE

The final key for DevOps at HPE was increasing trust within the organization. Teams need to be integrated and empowered for the success of DevOps, yet managers often found it challenging to surrender some of their power to their teams. However, HPE recognized that a manager’s goal is to create value for teams and businesses instead of applying a process. The role of the manager is to deliver the organization’s goals and targets to the team and trust the team to accomplish them with the appropriate work as efficiently as possible. Work is done transparently and openly, where value is placed on trust and responsibility instead of rigid processes.

Today, the teams working in DevOps mode at HPE are in high demand within the organization. They are able to create value for the business faster and maintain a high quality developer experience. DevOps “Dojos” are now part of the DevOps improvement plan at HPE, where knowledge on DevOps is shared and projects can attain information on the first steps on how to transition towards DevOps. Arne Luhrs’ vision on the three most important keys towards DevOps are culture, collaboration and automation.

5 Results, problems and barriers to DevOps

The interviews were able to provide details on the first research question of this research.

RQ1: What are the problems in Kesko's software development process?

The primary finding emphasized the variety in the methods of how development is conducted within Kesko's digital projects. The lack in a concise and applicable development model without true organizational guidelines was seen as a cause for an unclear overview of digital projects and silos that understand themselves but not beyond.

This chapter presents the discoveries made through the interviews of different employees within Kesko's digital domain and provides further answers to the first research question.

5.1 Kesko software development model

Answers to the question on describing Kesko's current software development model produced a wide range of observations. The lack of a common model has produced varying methods of development for digital projects that in turn have affected the overall acknowledgement on the complexity of Kesko's digital domain.

As of today, Kesko has acquired all of its digital development from outsourced vendors. These partnering development teams are responsible for the software development of their respective project, located either on premise at Kesko's headquarters, in other cities within Finland or abroad in other time zones. Business is in charge for the value creation chain and works most actively with the development teams. IT is in a supportive role between them. However, due to the lack of in-house developers, the role of the IT department and the goals within its teams were not seen as transparent.

Both agile development with lean processes and projects with waterfall development were discovered. For example, multiple digital projects within the grocery trade sector have been conducted with agile methods for some years through the cooperation of different vendors. These projects are a grocery mobile application and a new grocery online store. The teams involved in these projects develop, test and maintain their projects autonomously with automated tools. They were seen as conducting development according to the DevOps culture, without their respective project owners necessarily realizing the fact.

In some cases agile and waterfall approaches were attempting to work in unison, especially with looming handovers to maintenance. As agile places more emphasis on the quality of code rather than documentation, a problem DevOps would attempt to solve is imminent: operations requires high quality documentation to understand the digital asset it is receiving, whilst agile development focuses on the code itself and not on passing deep knowledge on to other functions.

The vast majority of digital projects mentioned during the interviews were seen as following a traditional waterfall model; handovers to maintenance vendors were considered the norm. The organizational culture appeared to dictate digital projects form the beginning: all digital assets were regarded as “projects”, meaning they had to be implemented with a clear deadline in mind. A predetermined finite ending to a digital project places the product into a state of expecting a handover to operations at some time, not if it a handover is made at all. The situation becomes more complex with the maintenance teams being from a different vendor than the development teams themselves.

5.1.1 Project types

An important observation was the variety in the nature of digital projects within Kesko, ranging from quick and agile projects such as mobile applications and online stores to monoliths such as ERP and SAP systems. Business critical systems such as SAP are an essential part of the organization’s business model. Such systems are difficult if not impossible to adopt to agile frameworks.

Kesko is predominantly in the position of having to accept different types of software development models in its digital portfolio. Some assets are more suitable for agile and DevOps than others.

Developers and IT managers alike reported varying degrees of transparency in the common goals of their respective projects within Kesko. Teams experienced differences in the reasoning behind their projects. Some were able to report a direct purpose for their team. Others felt key targets were missing. This produced an unclear situation for motivation and perception of being part of something.

The core structure of Kesko’s software development organization is executed based on reporting to one’s supervisor on each organizational level. On one end is the development team, which reports to the product owner or project manager. The product owner or project manager, who is either from the business or from the IT department, reports upwards to higher managers within their respective organizational structure. This hierarchical flow of information can push decision-making, even for smaller cases, unnecessarily high. Additionally, information during this upwards trending flow poses the danger of either being misinterpreted or missed altogether. The key issue, which presented itself during the interviews from this topic, was the need for providing development teams with greater trust and empowerment.

5.2 Silo forming through budgeting

5.2.1 The role of the IT department

A further finding of the interviews was the presence of silos within Kesko's digital environment. On the organizational level, the silos of the IT department and business were recognizable. The current level coordination and communication between different projects influences projects to distance themselves from each other.

Many decision-making steps were reported to be conducted on a top-down approach, where even small-scale decisions had gone through a decision-making process. This was perceived to have had a negative effect on the efficiency of value delivery and product deployment. The bureaucracy resulting from this decision-making process was seen as a bottleneck for faster deployment.

The organizational matrix of having a separate IT department from the business sector was reported to have had an effect on the forming of silos. Business strives to create value under the administration of Kesko's IT, however IT was reported to not necessarily be able to provide all required resources at all times. Currently, IT and business should be able to collaborate for the best value creation pipeline. However, due to the current level of collaboration of the two departments in practice, they were not perceived as working together for a common goal as of yet.

By default, software development teams at Kesko are not traditional agile and DevOps teams such as in software development companies. These companies can provide the technical skills in-house. At Kesko, developers from an outsourced vendor create the development team, Kesko's business provides the product owner and value creator role and IT is the supportive function providing the budgeting. Further down the development pipeline is the outsourced maintenance. All of these parts were reported to lack in the levels of cross-collaboration for the efficient implementation and maintenance of a digital asset. Instead, the project managers collaborated with other managers.

Reflecting on the Westrum model on organizational culture, the interviews showed that bureaucratic culture is present within Kesko's digital domain. Functioning communication is present, but only within the silos themselves. Information flow between different stakeholders of a digital project is limited, leading to general barriers of effective development, deployment and maintenance.

5.2.2 Budgeting and prioritization

The divide between departments was seen as the most significant barrier for cultural change and conversely transitioning towards continuous development. Reasons for the divide between departments and teams were reported to lie in the limitations set by the current bureaucratic processes within the organization that had varying impact levels on communication, collaboration, decision-making, responsibility, transparency and digital asset management.

Budgeting is conducted on a yearly cycle, where the IT department assesses changes or new features requested by the business unit. According to the interviewees, this created bottlenecks hindering quick and responsive development. The scope of this research does not address the budgeting processes of Kesko. However, it is mentioned as the interviews reported budgeting processes to have a significant impact on the efficiency of being able to supply customer demands and value creation more efficiently. A revision of the budgeting system to a more stream-like approach, such as at Company A, could promote the autonomy of development teams and the weakening of silos.

Interviewees reported unclear prioritization mechanisms on digital projects to have been involved in the forming of the vague overall goals and targets. The digital portfolio appeared to lack clear priorities for projects, where multiple assets had been assigned the same priority. This in turn had created challenges in projects' budgets, their collaboration and resource sufficiency. As a result, a sharing mentality, whether for tools or resources, had not been able to develop as of yet. The prioritization mechanisms were therefore perceived as being another cause in the forming of silos.

Cost-efficiency measures are at the forefront of businesses. Kesko itself is now in the situation of having required a maintenance vendor for multiple digital projects in the future. From a DevOps cultural point of view, this step was reported to be counter-intuitive, as DevOps strives to create and maintain digital products within one team. The deal to transfer projects to one partnering maintenance team will foretell an increase in project handovers from development to maintenance. As mentioned during the interviews, the company is aware that it must run some projects with the waterfall method. However, with the contracts in place, Kesko now has the opportunity to carefully prepare for the handovers.

5.3 Automation, guidelines and toolsets

5.3.1 Automation and tools

Kesko is currently in the position of not controlling the tools of all of its development teams. This is because outsourced development teams use their own automation tools. This was not considered a problem during the interviews as development teams should have the freedom to use the tools they are most experienced with. Furthermore, Kesko itself is not able to provide the technical knowledge of setting up automated pipelines. This in turn has formed an environment where each development team conducts automation differently and according to their own best practices.

As outsourced teams use automation and monitoring tools for their digital assets, these tools limit themselves mostly to each respective project. The culture of deploying shared tools within Kesko is in progress. The interviews reported a lack in the transparency of the monitoring tools within Kesko. Projects did use monitoring tools for their own projects, however a centralized understanding of which tools were in use was not recognizable.

A team for maintaining shared development tools, such as the cloud infrastructure AWS and the communication platform Flowdock, exists. However, guidelines for using the tools were reported to not be in place. Some, but not all, interview participants were aware of the tool possibilities Kesko provides. Guidelines for using available tools were not considered transparent enough. Further problems were recognizable during the interviews: the capabilities of the tool maintenance team were limited in supporting Kesko's complete digital domain. Not all tools in use were under the supervision of this team. Therefore, the transparency and guidelines for unified tool availability were seen as lacking.

5.3.2 Testing

Consistent methods of software testing were not reported during the interviews. The current role of Kesko's IT for testing mechanisms was considered to provide a consulting role, where Kesko's own testing specialists informed projects of testing methods. However, the testing responsibility and methods were seen as residing within the projects themselves. This was believed to be the main reason for varying testing processes, leading to the absence of clear testing guidelines. Exceptions arose during the interviews of projects that tested their software throughout the development process. These were mostly the projects which developed with agile methods. However, transferring a product to a testing phase, which is common during waterfall projects, was regarded as an unsolved problem within Kesko's software development. With many projects, testing had been neglected until close to deployment time, which in turn caused testing complexities and scheduling challenges.

5.4 DevOps awareness at Kesko

Previous knowledge on DevOps varied between the participants. The majority of the interviewees approached DevOps only from the technological aspect. They regarded DevOps as being a method for automating development, which in turn produces faster development. Others recognized DevOps as being the unified collaboration between development and operations, where emphasis is placed on autonomous teams developing and maintaining their digital asset. Some of the interviewees had no previous knowledge of DevOps, but were familiar with agile software development, as some their projects had applied agile principles. The minority of interviewees, mainly the collaborating vendors, had deeper knowledge of all of the aspects related to a DevOps culture. These development vendors had already integrated the DevOps best practices into their own work, providing justification for their understanding of DevOps.

All interviewees agreed upon the need for cultural change within the organization. Improved communication and transparency were seen as the most significant step in transitioning towards continuous development. Furthermore, the interviewees also conceded the fact that cultural change is a challenging task.

From Kesko's point of view a comprehensive perception of DevOps' key factors was not present as of yet. However, the interviewees recognized the benefits of DevOps and were encouraged by the possibilities of its adoption.

6 Implementable DevOps practices at Kesko

Reflecting the received responses during the interviews with the theorems of DevOps, an overall picture can be drawn on the aspects Kesko should focus on in transitioning towards continuous development. This chapter focuses on answering the second research question.

RQ2: What points can Kesko focus on in transitioning towards continuous development?

The interviews gave insight into the fact that agile software development is still in its infancy at Kesko. However, a transition towards agile has already begun and more is being developed with agile methods. DevOps bases its practices on the methods of agile and lean. Hence, it is important to note that before a DevOps development mode can be implemented, significant emphasis must be placed on changing the overall project culture from waterfall to agile and lean first. Nevertheless, this does not exclude the DevOps culture from becoming a part of Kesko's culture already.

One of the findings of this research is that DevOps is not a model suited for every type of digital project. Business critical digital assets that are planned thoroughly and executed systematically, must inevitably be kept separate from the complete DevOps movement. Improvements on collaboration and sharing can still be applied to such development, as culture should develop on an organizational level.

The first step for Kesko should be to unify its software development. This can be accomplished with sets of clear guidelines that affect different digital projects. Especially DevOps requires a set of rules on how development is conducted, therefore a type of DevOps mantra is needed.

The following findings present points Kesko can take into account today when considering the question of what problems DevOps could address within its organization. These findings are based on the problems and ideas discovered during the interviews, the DevOps theorem and the experiences provided by other large organizations that have implemented DevOps in their own development culture. All of the aspects considered during this paragraph should be addressed in the DevOps guidelines.

6.1 Culture, collaboration and trust

6.1.1 Trust and empowerment

Much like HPE, a significant step towards DevOps would be for Kesko to attempt changing the culture through which projects are managed. A top-down approach for managing teams and digital projects can create not only bottlenecks, but also hinder efficient decision-making. Automation guidelines can move decision-making towards a leading-with-data approach during development.

Building trust and empowering teams and individuals does not happen overnight. A first step towards DevOps could be for Kesko's managerial approach to change towards pushing discussion to occur directly between developers and operators, meaning having its technical experts talking to other technical experts. This could remove some of the silo effects recognized during the interviews. The current collaboration culture causes developers to report to managers and managers discussing technical knowledge with each other. Removing such a stage and facilitating communication between experts could be a part of Kesko's DevOps culture, which promotes development that is more efficient. Removing barriers between business and IT could be managed in a similar manner. Integrating IT's service managers, testing consultants or enterprise architects earlier into the development teams with business could improve trust and collaboration. Additionally, project development teams and operations teams should be encouraged to collaborate more freely at an earlier stage.

A further change towards building trust is shifting away from the need to manage processes which have been formed by the bureaucratic limitations. Kesko's position as a software customer forces its managers into a supervisory role of the development teams, where either the manager or a higher official makes most decisions on development. Decisions that could be made by the team, especially technical decision, should be entrusted to the team instead of having to follow strict protocol slowing development down. Managerial focus should be able to shift from process management to focusing on their own expertise, which is value creation management. The recognition of such bottlenecks and their subsequent mitigation favors the transition towards a DevOps culture.

6.1.2 One tool for everything

Conducting open and efficient communication requires the correct setup. Facilitation all of its teams and their respective team members at one location is the ideal setup, however Kesko is not in the position of being able to locate all of its teams into one location. Hence, the appropriate tools are necessary.

Focusing all communication and all of the data produced during development into a single tool was a key factor for HPE to adapt DevOps. HPE was able to apply an integration capable communication tool on an organizational scale for everything conducted during development. One such tool was Flowdock, which is partially in use within Kesko's software development. The next step would be to make such a tool not only mandatory but expect every other tool, such as the project management tool Jira or the monitoring tool New Relic, to be integrated into the same chat. This method leads to a greater degree of transparency and efficiency as every discussion, problem and all relevant data can be found in one place. Problem recognition and solving becomes much quicker, as opposed to more traditional e-mail communication. Through the unified tool, team members are aware of all of the aspects that are in progress, and they can conduct research of other issues on their own. As all project relevant data and events are integrated into one tool, notifications on urgent matters such as failures or updates are visible to the whole team. With the availability of desktop and mobile versions of such collaboration tools, team members are continuously aware of the project's status.

Quick and efficient communication is an essential part of DevOps culture. Shifting away from e-mail communication opens doors towards transparency and employees to become more involved in the development process. E-mail communication has the disadvantage of being only visible to the employees involved in the discussion chain. Shifting the discussion to an open communication platform has the effect of information having to be posted only once. This eliminates the building of e-mail clutter, which can cause information to be lost during an e-mail only discussion. Questions on an open discussion platform are answered more efficiently as someone who knows the answer can be found quicker.

The key recognition for Kesko is that it is not a software company. As of now, the majority of the technical knowledge lies within its outsourced software development teams. Therefore, Kesko should be more active in creating an environment for DevOps collaboration between the teams developing for them. This could mean more sharing, more cooperation and more communication even between teams originating from different companies. Outsourced development teams are not expected to maintain a project forever. Hence, the transfer of project knowledge from development to operation becomes a critical point during the later stages of a digital asset. Promoting the collaboration between development and operations aids in the successful transfer, which in turn is helped by documentation. Using an open collaboration platform can positively influence all of these aspects.

Communication plays a significant role in team building. Using the unified communication tool for socializing and discussions of other topics besides work should be encouraged.

Hewlett Packard Enterprise emphasized the importance of treating team members as humans instead of resources.

6.1.3 Documentation

Operations teams place a high priority on proper documentation. Comprehensive information on digital products and services is necessary for sharing the knowledge related to them. In case of failure the operations team must have access to the knowledge regardless of time of the day. Maintenance of a digital asset requires detailed specifications on how the asset is built. This is the purpose of thorough documentation: promote the accessibility of the detailed specifications.

Kesko has outsourced the creation and maintenance of its digital products and services to different vendors. Each vendor currently places different priorities on their respective documentation. As different companies conduct development and maintenance, great emphasis must be placed on transferring the knowledge of the product from one vendor to another. A proposal for Kesko's DevOps culture therefore is to incorporate the expectation of detailed documentation from every development team into the DevOps guidelines. This maximizes knowledge transfer efficiency from development to operations.

Agile software development places greater priority on development than documentation. Kesko is the initial customer and end-user of the software products and services its outsourced development teams create. Furthermore, Kesko is also in a position of transferring these products and services to another vendor at the maintenance handover stage. Therefore, Kesko should place great emphasis on incorporating high quality documentation into its development culture. Access to the documentation should not be limited. Documentation access could be incorporated into the unified collaboration tool.

6.1.4 Team structure

Teams that develop software in pure DevOps mode, are cross-functional teams capable of developing, running and maintaining their product. The core DevOps mantra of "you build it, you run it" applies to teams with the appropriate skill set. As Kesko employs software development teams from outsourced vendors, the creation of cross-functional teams is challenging. The team, which creates the product, is in charge of maintaining it whilst they are developing for Kesko. However, many digital assets eventually reach a handover phase, where parts of the product's maintenance are transferred to operations. In Kesko's case, this maintenance vendor is a different outsourced company. Precise documentation aids in the transfer, yet true knowledge and experience of the product resides in the minds of the team members.

An early step towards shifting teams into DevOps mode could be to integrate members from the future operations team into the development team at an early stage. Through this change, knowledge and ownership of the product would not only grow within the development team, but also for the members responsible of maintaining it. This could produce an opportunity to bridge the knowledge gap of understanding the product thoroughly between development and operations in Kesko's situation. The benefits of having an operations' member integrated into the development team presents multiple benefits. Specifications from the operations team can be passed on to development at the earliest stage. The operations team eventually does not only have to rely on the documentation, but can refer to the experience gained from its team members who were a part of development.

Kesko is in the process of hiring its own developers. Depending on their initial numbers, these developers could be integrated into development teams working on the highest priority assets at first for bridging the technical knowledge gap currently at Kesko. Such an approach to team creation would aid in retaining technical knowledge within Kesko.

Building a cross-functional DevOps team requires not only knowledge on development and maintenance from its members, but also social skills to advance team dynamics. Kesko should have a more dominant role in the recruitment of its outsourced teams. Being able to reassure that development and operations employees are suited for the Kesko's DevOps guidelines and ultimately selecting the correct people for its teams is an essential part in developing a functional DevOps culture.

Team building exercises reinforce each individual's growth within the team. As the importance of a single team grows within an evolving DevOps culture, so does the need for ensuring a functioning team dynamic. Building trust within a team consisting of different vendors takes significantly more time and effort than a team from a single vendor. Kesko's managers should play a role in developing this team dynamic and trust.

DevOps teams do not have "project managers" in theory, but Kesko does because of its position in outsourcing development. Project management acts as the direct bridge between business and software development, hence the position's criticality is significant.

6.1.5 Defining clear project types

DevOps is the next progressive advancement from agile software development to higher efficiency. Therefore, a complete DevOps culture should be implemented in digital projects that can be conducted with agile methods. Business critical projects such as new point of sales technology requires careful planning and vigorous testing before deployment. These kinds of projects have the precise requirements from the beginning and are therefore justified as being conducted with waterfall methods.

The digital product, which a specific project is attempting to create, should have the product lifecycle specified at the very earliest stage. The clearer the lifecycle of a product is defined, the simpler it is to plan and apply the appropriate software development methodologies. According to the interviewees, product lifecycle was an aspect where Kesko could improve upon whilst idealizing a new product or service. Having product lifecycle aspects built into the development process early could allow for clearer budgeting and prioritization procedures. Recognizing the timeline for which a product is to be developed aids in prioritization and preparation for a potential handover.

Therefore, Kesko should carefully consider the creation of not only one software development model, but several different models. These models could differ according to nature of the digital asset and be applied accordingly. The agile model could be used for products with rather unknown specifications such as mobile applications and the waterfall model for projects with strict specifications. Perceived customer value is not always known, such as in mobile applications. They must be developed with agile methods where features and customer wants can change rapidly.

The models should have clear, but flexible, templates for how a digital asset is developed. Once a new project is started, the template would define the tools that are available for the project to utilize, how testing and quality assurance is to be conducted and how a potential handover is to be prepared for.

The distinction in project types does not mean to neglect agile and DevOps best practices. Cultural changes can and should ultimately affect the organization as a whole. Waterfall type projects can apply the cultural changes mentioned previously throughout this paragraph to improve efficiency and collaboration. Therefore, Kesko should incorporate multiple clearly defined project types into its guidelines and apply its DevOps guidelines for assets which are most suitable for the DevOps culture.

The multitude of digital assets and services at Kesko makes it challenging to implement changes to every asset initially. Especially DevOps is not implementable for every type of product development. Hence, it is vital to be able to distinguish the type of project at the earliest stage.

6.2 Automation and tools

Although DevOps focuses extensively on culture, it also incorporates the building of a continuous delivery pipeline with the use of automation tools for integration and deployment. During the development of a new digital asset, every checkpoint that is automatable, should be automated. The use of cloud applications should be promoted on an organizational level.

6.2.1 Automation pipelines

Without any own in-house developers or development teams, Kesko needs to continue relying on the technical knowledge provided by its development partners for building automation pipelines. As these development teams currently create the pipelines based on their own expertise, Kesko's role in automation should be more present in demanding automated deployment pipelines and setting precise requirements for them. Instead of providing automation tools, Kesko should focus on creating the rules within the pipelines, just as HPE has done. These rules should provide automation as much as possible in order to make deployment and decision-making as simple as possible. This however needs an improved visibility of the complexities and dependencies of all of its systems: not only the cloud infrastructure in use today but also the active legacy systems. The culture and sharing improvements should present an opportunity in providing such information for anyone requiring it as fast as possible.

Having an automated deployment pipeline, which incorporates automatic testing for builds, would result in the reduction of risks and difficult decision-making. The emphasis on thorough testing throughout the complete development process would provide high quality software coupled with a proficient deployment process. Hence, testing already at an early stage of development should be incorporated into the DevOps automation guidelines set by Kesko. Conducting all tests at the end of development creates scheduling and quality challenges. This approach should be avoided by requesting continuous deployment pipelines for each project that automate wide scale testing with each new code commit. Therefore, testing would be conducted constantly during development, instead of being a process during the finalizing stages of development.

Creating a set of rules and requirements could be the first step for Kesko to be more involved in the automation conducted within its digital projects. The key aspect to keep in mind for setting the rules, such as the type and amount of testing, on automation pipelines is how to prevent large releases and instead promote smaller and more frequent releases within each project.

6.2.2 Monitoring and metrics

Having a centralized knowledge of infrastructural and application systems on an organizational level is a vital part of DevOps. Recognizing failures early and often aids significantly in development that is more efficient. Wide-scale monitoring leads to quicker recovery from failures and deeper learning of the created value from software changes. Striving for functioning monitoring tools is essential, but furthermore it is critical to deploy the same monitoring tools and metrics for every project and department. Such unified metrics enable the comparability between digital assets and further the knowledge on successful development practices.

Unified monitoring tools should be accompanied with unified metrics in the DevOps guidelines. Making project progresses and statuses more transparent, accessible and comparable with new metrics could be an integral part of Kesko DevOps. Metrics such as cost per deployment could reveal overall efficiency of development teams. Other metrics such as bugs per build or team satisfaction metrics could be developed in order to reveal the true work efficiency within the team. Metrics for the quality of code could provide feedback on the outsourced vendor, however because of the lack of in-house development teams these metrics should be conducted by the vendor teams.

The recorded metrics should cover three areas: people, processes and technology. Understanding employee satisfaction, motivation and recognizing bottlenecks in development and organizational processes can provide insight into the hurdles associated with non-technological aspects during development. Metrics on technology aid in improving the output provided by the employees [6, p. 44].

6.3 DevOps awareness

Increasing the awareness of DevOps and its benefits is an effective way to spread its best practices into the organization successfully. Sharing previous success stories and making existing achievements in agile, lean and DevOps more transparent and accessible can become a catalyst for the required cultural and technological changes. As DevOps is in its essence mostly a human and management problem, the shift towards continuous development and DevOps arises from these organizational roles. As sharing is a corner stone of DevOps, sharing its angles openly can become vital during the creation of DevOps guidelines within Kesko.

6.4 Action plan

Depending on their nature and industry, specific DevOps practices can differ vastly between companies. As Aruna Ravichandran, Kieran Taylor and Peter Waterhouse recognize in their book [6], the action plan for beginning to adopt a DevOps culture consists of seven fundamental points:

1. Recognizing business goal
2. Finding support from senior management
3. Choosing the correct people
4. Choosing quick deliverable
5. Developing suitable metrics
6. Incorporating DevOps with other processes
7. Developing automation

Understanding the needs and goals of the business is the driving force behind DevOps. Building such a culture should rise from striving to improve customer value creation. IT and business should collaborate towards satisfying the common goal. [6, p. 164-165]

In order to gain support for DevOps throughout the organization, it is integral to have trust from senior management for the benefits of DevOps. Higher management is able to create a more credible force behind the adoption of a change. [6, p. 165].

As DevOps revolves heavily around people, selecting the right people has a significant impact on the first development changes towards DevOps. People involved in an evolving DevOps culture need not only be able to work well in teams, but also show resilience and flexibility as failure can accompany change. [6, p. 165-166]

DevOps development is most suited for assets that are quickly deliverable. The application or project should be manageable and deliverable whilst trialing with different DevOps practices. Suitable examples are mobile applications or webpage development. [6, p. 166-167]

The success of the DevOps development needs to be measurable. Therefore collaborate goals and the metrics to measure the success of DevOps must be developed. The metrics and their respective goals should be coordinated with the business goals, but also measure and provide feedback on the development process. [6, p. 167-168]

All stakeholders involved during the development of a digital product are to be taken into account. Therefore, forcing completely new processes and approaches should be taken lightly. Instead, the developing DevOps culture should be adjustable and capable of being integrated with other existing processes in order to satisfy the needs of all stakeholders involved. . [6, p. 168]

Building the automation pipeline should begin by assessing the most significant hurdle in the current delivery pipeline. Although the ultimate goal of automation is to commit, build and release with the least amount of manual labor, the primary step should be to automate the process currently being the most significant bottleneck in the development process. [6, p. 168-169]

These seven steps create a basis from which to begin the transition towards DevOps. Cultural change on an organizational level is complex; therefore beginning the journey towards DevOps should begin on a small scale. The success of such beginnings can become a platform for major change.

7 Discussion

The focus of this research was to discover how Kesko could transition towards the continuous software development model of DevOps whilst utilizing cloud-computing applications. Direct connections can be drawn between areas where Kesko could improve in order to work under a DevOps culture. However, the major finding was that the culture within Kesko is not fully ready to deploy teams functioning under a DevOps culture. Kesko's first task should be to implement lean and agile approaches to not only its digital development, but also to the separated departments of business and IT. As changing organizational culture within a large corporation is a challenging task, Kesko nevertheless has the opportunity to transition towards continuous development. Hence, the final research question can be answered within this chapter.

RQ3: Could DevOps be adaptable?

Developing a DevOps culture is a process, where suitable practices are adopted iteratively. DevOps in Kesko's case is clearly adaptable. However, it requires a cultural shift promoting more collaboration between development, operations and Kesko's other business functions and a more wide range agile or lean adoption.

This chapter answers the final research question in more detail and discusses some of the points that would create an ideal DevOps culture at Kesko.

7.1 Ideal situation

Having a fully functioning DevOps culture within a large organization is a major task to reach. As an example, HPE has been working on their respective DevOps culture for three years as of today.

In an ideal situation, Kesko would have a complete set of guidelines for its own established DevOps culture. These guidelines would direct software development, wherever applicable, to completely ingrained agile methods with fully automated deployment pipelines. DevOps at Kesko would be executed with a template, which portrays all of the required rules for testing, integration, deployment and collaboration.

The development teams would be completely autonomous. These teams would develop and maintain their product at all times, without any handovers to maintenance taking place. Changes or new features could be implemented at any time and failures would be recognized and repaired immediately. Ideally, the teams would consists of only in-house developers from the IT department who collaborate with the business unit for creating customer value via its digital products and services. The continued employment of software consultants for digital projects comes at a high cost. This is the reasoning behind the current model of requiring handovers to maintenance. Therefore, as the discussion is generally about cost optimization, the outsourced software consultants are often times an unsuitable option for maintaining the software created for Kesko.

As one development team cannot be tied up with a single digital asset forever, the carefully created project type models are responsible for the timeline during which the team is responsible for maintaining its product or service. These models take the product prioritization and lifecycle into account at the beginning of the project.

Unified automation and communication tools could be provided by the organization's IT, as all development teams are Kesko's own.

With the current construction of a centralized office, Kesko would have the ability to locate its potential development teams in one location.

7.2 DevOps team

The scope of this research limits itself to uncovering Kesko's current software development hurdles and attempting to apply aspects of the DevOps culture into the current environment. Therefore, in order to attain a complete overview of the problems, complexities and dependencies of all of Kesko's IT systems, a dedicated DevOps team is required. The task of this team would be to map out all cultural and technological aspects and processes within Kesko. Furthermore, the team would propose changes that incrementally lead to a transition towards continuous development. Such a team would require the support of the organization's top management. Further material on such a dedicated DevOps team can be found in literature, as for example Paul Swartout's book *Continuous Delivery and DevOps: A Quickstart Guide*. Swartout emphasizes the difficulties of changing organizational procedures and therefore suggests the deployment of such a team.

Multiple consulting companies such as Eficode and Puppet provide organizational change services, especially towards DevOps culture. The dedicated team could either be created with the knowledge and support of these companies or with Kesko's own employees. However, as Swartout suggests, the key for the DevOps team is to focus only on developing DevOps. Additionally, the team requires not only knowledge on cultural change but also on technical knowledge for being able to create automation pipelines.

8 Conclusions

Transitioning towards continuous development in its simplest terms requires a faster software release cycle. Releases must be conducted more often and in smaller chunks. Creating such an environment for quick iterative releases becomes more challenging with larger organizations, not necessarily because of technical challenges, but because of the underlying dependencies between different departments and teams.

This research has been able to discover the existence of such substantial dependencies within Kesko, but also the silos that currently limit the collaboration between these dependencies. In order to shift towards continuous development, these dependencies should be recognized and improved upon. As of today, Kesko's software development situation on an organizational scale is in its infancy in regards to continuous development and DevOps. Some teams do practice the DevOps culture within their daily work. However their implementation of DevOps stems from the fact that these teams are outsourced vendors that have applied the DevOps culture within their own respective organizations. Therefore, the DevOps culture practiced currently within Kesko is not Kesko's own. However, valuable information, lessons and experiences on continuous delivery can be directly acquired from the current DevOps development practiced by the vendors at Kesko.

Although the development of Kesko's DevOps culture is only at its first initial stage, current organizational structures and processes were regarded as being hurdles for the company's own DevOps culture. As DevOps takes agile or lean development for granted, Kesko should begin by focusing on creating its own development guidelines that incorporate functioning aspects of agile and lean into the existing organizational processes. The creation of these guidelines should assemble all stakeholders involved in the DevOps culture, for example product owners, developers, enterprise architects, operations, maintenance, release managers or information security specialists. The formation of DevOps silos should be avoided at all costs [31, p. 282].

Due to the limitations set by the current organizational structure, the question of what DevOps is expected to achieve should be taken into account. DevOps itself should not be considered as an ultimate solution, but instead a result of a functional structure. Hence, in order to adapt DevOps, transition towards continuous development and create a modern way of developing, significant cultural changes mentioned throughout this research need to be undertaken: such as measures to improve transparency and collaboration. DevOps ultimately is the functioning combination of work culture and modern technical tools.

Another question to consider is the range on which DevOps should be applied: should DevOps become the norm for every digital asset or only applicable were necessary? Acknowledging the fact that DevOps is not applicable for every type of development aids in recognizing the potential DevOps can bring to the organization. As other larger firms, such as Company A and HPE have recognized before, although tedious, change is necessary for building a new software development model.

The results gained from this research can be considered as concrete points of emphasis on where to begin mapping out an action plan for the development of DevOps. The interviews provided insight into the current state from employees directly involved in every day development. Comparing the interviews with the theorem of DevOps and other organizations produced results of cultural and technical value, which Kesko could consider expanding on. Nevertheless, the importance of Kesko's own identity regarding DevOps should remain at the forefront. Experiences on DevOps from other parties should be surveyed, yet the formation of DevOps should be reflected on Kesko's own needs. This research recognizes the vastness of the requirements for organizational change. Therefore, greater research is required to chart out and improve upon the organizational challenges associated with a large company such as Kesko for transitioning towards continuous development.

9 References

[1] Powell-Morse, Andrew . Rapid Application Development (RAD): What Is It And How Do You Use It? Online Document. Updated 2016. Cited 14.4.2017 Available: https://airbrake.io/blog/sdlc/rapid-application-development

[2] New Relic. What is DevOps? Online Document. Cited 15.4.2017. Available: https://newrelic.com/devops/what-is-devops

[3] Powell-Morse, Andrew. Waterfall Model: What Is It and When Should You Use It? Online Document. Updated 2016. Cited 14.4.2017. Available: https://airbrake.io/blog/sdlc/waterfall-model

[4] Oxagile. Waterfall Software Development Model. Online Document. Updated 2016. Cited 14.4.2017. Available: https://www.oxagile.com/company/blog/the-waterfall-model/

[5] Petersen, K. & Wohlin, C. & Baca, D. The Waterfall Model in Large-Scale Development. Online Document. Updated 2009. Cited 14.4.2017. Available: http://www.wohlin.eu/profes09.pdf

[6] Ravichandran, A. & Taylor, K. & Waterhouse, P. DevOps for Digital Leaders - Reignite Business with a Modern DevOps-Enabled Software Factory. Apress, 2016. 173 p. ISBN 978-1-4842-1842-6.

[7] Monden, Y. Toyota Production System: An Integrated Approach to Just-In-Time. 2nd ed. Springer, 2012. 424 p. ISBN: 978-1-4615-9716-2

[8] Poppendieck, M. & Poppendieck, T. Lean Software Development: An Agile Toolkit. Addison-Wesley Educational Publishers Inc, 2003. 240 p. ISBN: 978-0-321-15078-3

[9] Kelly Waters. 7 Key Principles of Lean Software Development. Online Document. Updated 2010. Cited 25.4.2017. Available: http://www.allaboutagile.com/7-key-principles-of-lean-software-development-2/

[10] Todd Brasel. Lean vs. Agile: What's the Difference? Online Document. Cited 16.5.2017. Available: https://goleansixsigma.com/lean-vs-agile-whats-the-difference/

[11] Manifesto for Agile Software Development. Updated 2001. Online Document. Cited 18.5.2017. Available: http://agilemanifesto.org/

[12] Principles behind the Agile Manifesto. Online Document. Cited 18.5.2017. Available: http://agilemanifesto.org/principles.html

[13] Andrew Phillips. The Continuous Delivery Pipeline — What it is and why it's so important in developing software. Online Document. Updated 2014. Cited 14.5.2017. Available: https://devops.com/continuous-delivery-pipeline/

[14] Martin Fowler. Continuous Integration. Online Document. Updated 2006. Cited 12.5.2017. Available: https://martinfowler.com/articles/continuousIntegration.html

[15] Carl Caum. Continuous Delivery vs. Continuous Deployment: What's the Diff? Online Document. Updated 2013. Cited 16.5.2017. Available: https://puppet.com/blog/continuous-delivery-vs-continuous-deployment-what-s-diff

[16] Martin Fowler. Continuous Delivery. Online Document. Updated 2013. Cited 16.5.2017. Available: https://martinfowler.com/bliki/ContinuousDelivery.html

[17] Matt Watson. Divvy Up DevOps Tasks, Defining the Ops in DevOps. Online Document. Updated 2017. Cited 26.7.2017. Available: https://stackify.com/defining-the-ops-in-devops/

[18] Jezz Humble. There is No Such Thing as a "Devops Team". Online Document. Updated 2012. Cited 18.5.2017. Available: https://www.thoughtworks.com/insights/blog/there-no-such-thing-devops-team

[19] DevOps Dictionary. CAMS. Online Document. Updated 2015. Cited 20.5.2017. Available: http://devopsdictionary.com/wiki/CAMS

[20] Swartout, P. Continuous Delivery and DevOps: A Quickstar Guide. Packt Publishing, 2012. 171 p. ISBN: 978-1-84969-368-4

[21] John Willis. DevOps Culture (Part 1). Online Document. Updated 2012. Cited 20.5.2017. Available: http://itrevolution.com/devops-culture-part-1/

[22] Westrum. R. The study of information flow: A personal journey. Safety Science. Issue 67. 2014. p. 58-63.

[23] New Relic. The benefits of DevOps. Online Document. Cited 15.4.2017. Available: https://newrelic.com/devops/benefits-of-devops

[24] DevOpsGuys. Why DevOps? Online Document. Cited 15.4.2017. Available: https://www.devopsguys.com/why-devops/

[25] Puppet. 2016 State of DevOps Report. Online Document. Updated 2016. Cited 18.3.2017. Available: https://puppet.com/resources/whitepaper/2016-state-of-devops-report

[26] Dawn Foster. What is DevOps? Patrick Debois Explains. Online Document. Updated 2016. Cited 15.4.2017. Available: https://www.linux.com/blog/what-devops-patrick-debois-explains

[27] Alex Manly. 5 Challenges to DevOps adoption and how to overcome them. Online Document. Updated 2017. Cited 15.4.2017. Available: https://www.contino.io/insights/5-challenges-to-devops-adoption-and-how-to-overcome-them

[28] Curtis Franklin Jr. Agile Vs. DevOps: 10 ways they're different. Online Document. Updated 2016. Cited 15.4.2017. Available: http://www.informationweek.com/devops/agile-vs-devops-10-ways-theyre-different/d/d-id/1326121

[29] Jayne Groll. What is a DevOps 'Best Practice'. Online Document. Updated 2016. Cited 18.3.2017. Available: https://devops.com/devops-best-practice/

[30] Apprenda. IaaS, PaaS, SaaS (Explained and Compared). Online Document. Cited 17.5.2017. Available: https://apprenda.com/library/paas/iaas-paas-saas-explained-compared/

[31] Kavis, M. Architecting the cloud – Design decisions for cloud computing service models (SaaS, PaaS and IaaS). John Wiley & Sons Inc, 2014. 351 p. ISBN: 978-1-118-82627-0

[32] Mike Roberts. Serverless Architectures. Online Document. Updated 2016. Cited 8.5.2017. Available: https://martinfowler.com/articles/serverless.html

[33] Marston, S., Li, Z., Bandyopadhyay, S., Zhang, J. & Ghalsasi, A. Cloud computing — The business perspective. Vol 51. Decision Support Systems. 2011. p. 176-189. Available: http://www.keencomputer.com/images/KEENCOMP/CLOUD/cloud-computing-business-perspective.pdf

[34] Grossman, R. The case for cloud computing. Vol 11, Issue 2. IT Professional. IEE. 2009. p. 23-27. Available: https://pdfs.semanticscholar.org/fd95/05897a97b2f82a73148dc87ce3067a33c6ab.pdf

Appendix

Common questions for everyone:

1. What is DevOps in your opinion?
2. How do you see Kesko's current continuous development model?

   a. What is good? What could be improved?
3. What do you consider a successful working culture?

   a. How do you see a change taking place?
4. What pain points come to mind currently at Kesko in your projects?
5. What are the barriers in coordinating your projects efficiently?
6. Do you believe changing the development model is a challenge?

   a. Why or why not?
7. How do you currently see the willingness of changing to a more modern model within Kesko?

   a. How could that be changed?
8. Which do you see as more important and why: freedom or overall project management?
9. If given the chance would you agree with removing handovers wherever possible?

   a. How could this be done?
10. How often do you think software should be deployed into production?
11. What roles do you expect DevOps teams to contain?

Questions for developers:

1. You as a potential partner: how do you see working with other vendors in one team?
2. If Kesko had a specified DevOps model: as a potential partner are you excited or burdened?

   a. How much of your own tools or your pipeline would you be willing to change?

b. Which should be the same?

2. How is testing implemented currently in the projects you are a part of?

3. Can development be conducted fast or efficiently enough at the moment?

   a. Why or why not?

   b. How would you develop it?

Questions for business:

1. What are your expectations of CAMS in Kesko?

2. What is the biggest blocker for faster development?

3. Should digital projects have a model with a finite ending?

   a. Why or why not?

Questions for IT:

1. How is testing implemented currently in the projects you are a part of?

   a. What problems do you recognize?

2. What role do you see yourself in currently within the digital projects?

3. What role do you see yourself in the future within the digital projects?

4. Can development be conducted fast or efficiently enough at the moment?

   a. Why or why not?

   b. How would you develop it?