Anssi Lahtinen

TEST AUTOMATION STRATEGY IN DEVOPS ENVIRONMENT: AN IT MANAGEMENT VIEWPOINT

JYVÄSKYLÄN YLIOPISTO
INFORMAATIOTEKNOLOGIAN TIEDEKUNTA
2020

TIIVISTELMÄ

Lahtinen, Anssi
Testausautomaatiostrategian luominen DevOps-ympäristössä: Tietohallinnon näkökulma
Jyväskylä: Jyväskylän yliopisto, 2020, 88 s.
Tietojärjestelmätiede, pro gradu -tutkielma
Ohjaaja: Abrahamsson, Pekka

Jatkuvasti muuttuvat teknologiat, sekä jatkuvat muutokset niitä ympäröivillä markkinoilla ovat luoneet ohjelmistokehitysorganisaatioille tarpeen sopeutua muutokseen. Digitalisaatio ohjaa organisaatioita asiakaslähtöisiin lähestymistapoihin, sekä vaatii organisaatioilta uusia tapoja ja resursseja, joita ei mahdollisesti ole aikaisemmin koettu tarpeellisiksi. Jatkuvasti muuttuvat ympäristöt kuten pilvi- ja verkkopohjaiset teknologiat luovat tarpeen palveluiden kehittämiseksi nopeammin, paremmalla laadulla sekä pienemmällä julkaisusyklillä. Jatkuvan julkaisemisen sekä integroinnin periaatteet ovat luoneet edellytykset DevOps-viitekehykselle, joka ylläpitää ketterien ohjelmistokehitysmenetelmien tuomia hyötyjä, mutta muokkaa myös osaltaan organisaation rakennetta ja kulttuuria.

Tämä tutkimus pyrkii muodostamaan kuvan tehokkaan ohjelmiston testausautomaatiostrategian luomisesta DevOps-ympäristössä. Koska DevOps on järjestelmäkehityksen saralla melko uusi viitekehys, pyritään tässä tutkimuksessa myös määrittämään sen ydinkyvykkyydet olemassa olevan kirjallisuuden, sekä tutkimuksen perusteella. Mallia tutkittiin sen tuomien hyötyjen, taustavaatimusten sekä mahdollisten implementointia hidastavan esteiden löytämisen kannalta. DevOpsia on tutkittu myös strategisesta näkökulmasta, jolloin sen yhteyteen on liitetty liiketoimintastrategia sekä jatkuva innovaatiokehitys. Vaikka malli itsessään käsittelee ohjelmistokehitystä, on nämä näkökulmat otettu huomioon tutkimusta tehdessä.

Projektiluontoisten toimintamallien tapauksessa organisaatioilla saattaa olla ongelmia ketterien ohjelmistokehitysmenetelmien käyttöönotossa. Tutkimuksen ensimmäisessä osiossa keskitytään DevOpsin ja testausautomaation ilmiöihin yksittäisinä kokonaisuuksina, sekä niiden tehokkaaseen yhdistämiseen. Toisessa osiossa analysoidaan laadullinen haastattelututkimus, jossa selvitetään kuinka DevOps-kyvykkyydet näyttäytyvät organisaatiossa sekä selvitetään, kuinka testausautomaatiostrategia tulisi rakentaa.

Asiasanat: DevOps, testausautomaatio, ketterät ohjelmistokehitysmenetelmät, tietohallinto, muutosjohtaminen, liiketoimintaprosessit

ABSTRACT

Lahtinen, Anssi
Test automation strategy in DevOps environment: an IT management viewpoint
Jyväskylä: University of Jyväskylä, 2020, 88 pp.
Information Systems, Master's Thesis
Supervisor: Abrahamsson, Pekka

Software developing organizations need to adapt to the ever-changing technologies as well as constant alterations in markets surrounding them. Digitalization has steered organizations to customer-driven approaches while requiring new assets and skills which might have not existed before. The constantly changing environments such as cloud and web-based technologies require organization to develop services faster, with enhanced quality and in demand of smaller release-cycle. The requirements of continuous integration, continuous delivery and continuous deployment have created the framework of DevOps. While maintaining the benefits of agile software development methods DevOps also concentrates on changing the organizational structure.

This study concentrates on creating an efficient software test automation strategy in a DevOps environment in a case organization. Since the framework and its capabilities have been vaguely defined, model of DevOps was constructed from existing literature by defining the core capabilities of the framework. The model was studied to find out the benefits, background requirements and possible barriers of adapting the framework in practice. DevOps has also been researched from a strategical viewpoint as how the framework affects business and change management processes. While the framework strives to streamline developing practices, these dimensions were also carefully examined while constructing the research.

In case of project natured operating models, organizations might have problems of adopting agile software methods. The first section of the research concentrates on the frameworks of DevOps and software test automation and an efficient combination of these two methods. The second section is about conducting and analysing a qualitative interview research. This qualitative research is about examining how DevOps capabilities are presented in the current operating model and how a test automation strategy should be built.

Keywords: DevOps, software test automation, agile software development methods, IT management, change management, business processes

FIGURES

TABLES

TABLE OF CONTENTS

1 INTRODUCTION

As technologies rapidly change and evolve, organizations look for frameworks and tools that can enhance software development and business processes. One of the most intriguing frameworks of the last decade has been DevOps – a combination of agile software development methods and specific organizational shifts and cultural changes. Information systems are critical for organizations to thrive so efficient software development methods are constantly researched on. Organizations need to be proactive regarding software development, constantly changing requirements and changing technologies around them. These organizations need to adapt their operating models to match these dimensions as they look to streamline their processes. This chapter is divided to three sub-sections: motivation, research question, and structure of thesis.

1.1 Motivation

Technologies such as cloud and web-based services have created demand for different software development methods. This research concentrates on three software engineering methods: continuous delivery, continuous deployment, and continuous integration. These methods are built to enhance construction, testing, and releasing the initial software. While all these models resemble each other there are a few differences between them. Where continuous delivery automatizes the whole product pipeline except the actual release continuous deployment strives for automation of the whole pipeline. Continuous integration gives software development teams more possibilities to integrate their work which can improve release cycle and product quality. While these frameworks slightly differ from each other, the similarity between them is that DevOps utilizes all of them in a specific way. By enabling continuous integration and deployment the product release cycle can be done in smaller phases while continuously releasing new software for testing. This requires lot from software test-

ers as they need to keep up with software development. (Fitzgerald & Stol, 2017.)

DevOps – a combination of the word development and operations – is a framework that combines agile software development methods with organizational changes and continuous software development. (Lwakatare, Kuvaja & Oivo, 2015.) The purpose of DevOps is to bring these traditionally siloed teams together while transparently communicating and changing organizational structures. The definition of DevOps be somewhat vague with two different branches of research: the other concentrating on technical and tangible entireties while the other concentrates on organizational and cultural changes. By combining these two branches the defining characteristics of DevOps are defined as utilization of agile software development principles, test automation, continuous monitoring, combination of software development and operations, frequent software releases and cultural movement within organization. By adopting the framework organizations can enhance the quality of the final product and production efficiency, enable quicker reaction to requirement changes and streamlining the development process with different automated processes. (Virmani, 2015.)

Even though DevOps is a rather new concept, test automation has been a part of software development for decades. Software testing can be divided to two categories: manual and automated testing. Test automation is automation of different activities such as development and execution of test scripts, confirmation of different testing requirements and usage of automated testing tools and methods in software development (Karhu et al., 2009). By enabling test automation software development teams can reduce costs, minimize the amount of manual labour, and increase efficiency and quality of software development. Test scripts can be executed to test a specific part of the software or ultimately the entire program. Automated test scripts can help organizations allocate the resources needed in manual testing elsewhere, minimize the possibility for human errors and run tests on a more frequent basis. This can be utilized especially in regression testing, where typically tests of the previous release are executed in the newer version to inspect functionalities and possible defects. However, software development teams should not automate every test possible, but those which are cost-effective to be executed manually. This requires careful design from test developers and IT management since some manual testing is required in almost every software developing project.

DevOps itself is one of the most trending software development methods and frameworks during the last decade. The nature of utilizing agile software development methods while combining siloed business units by enabling transparent communication throughout the project is attracting software development teams to implement the framework. However, the operating model is a complex entirety which needs to be implemented in carefully planned phases. The difference between previous agile development models and DevOps is the aspect that it strives for change in organizational structures and culture. According to Senapathi, Buchan and Osman (2018) supporting software engineer-

ing capabilities are just as important as the technical aspects of implementing DevOps. These capabilities include dimensions such as open communication, responsibility alignment and trust among management and employees. Learning new methods and technologies as well as feeling valued and higher level of autonomy have positive influence on engagement to the project. According to the researchers these capabilities can be encouraged and enhanced by IT management which could ultimately lead to a more successful product release. With successful implementation of supporting capabilities organizations can enhance their probability of a successful DevOps implementation while boosting the overall employee morale.

1.2 Research question

Even though software test automation is a part of DevOps framework there is minimal amount of empirical studies around the combination of these methodologies. The synthesis of this research is based on the BizDevOps model by Fitzgerald & Stol (2014) which argues for implementation of business strategy in DevOps environment. The business processes involved are dynamic open-ended artifacts that develop with the changes to business environment. Integration between business processes and software development is needed as these two dimensions require tighter integration in-between. The goal of this research is to find recommendations and guidelines for organizations to follow for cost-effective test automation strategy implementation. For these reasons, the research question can be shaped the following way:

- *How to implement test automation strategy in DevOps environment?*

To answer the research question, the research itself is divided to two specific sections with the first section being a theoretical framework built on existing literature and research. The second part of this study is the empirical section which is discoursed as a qualitative interview within the case organization.

1.3 Structure of thesis

Chapters two to four address DevOps and software test automation as separate dimensions, as well as creating the theoretical framework of successful combination of them. The first chapter of the theoretical section concentrates on defining the core capabilities of a DevOps framework and how they are visible in the organizational context. This chapter also addresses the benefits formed when implementing the framework successfully. Chapter three concentrates on software test automation separately and which benefits and disadvantages the model can produce when implemented. The last chapter of the theoretical sec-

tion ties DevOps, software test automation and business strategy as a theoretical synthesis derived from Fitzgerald and Stol (2014). The theoretical section of the research was conducted as a literature review to create a theoretical synthesis for software test automation strategy. The actual literature of this section included scientific publications of software engineering, Information Systems, and strategical management. Most of the literature came from the following publications: IEEE Software, International Journal of Computer Theory and Engineering, Journal of Systems and Software and Information Systems Management. Because of the minimal empirical research of DevOps Google Scholar was also used as a database of gathering theoretical implications of the framework. Regarding DevOps, the assessment of references was reflected more on authors and publication than the number of citations made. This is because the framework itself is rather new and the empirical studies regarding the model has merely started.

The rest of the research is covered by the empirical section. The theoretical synthesis of the framework applied was built on previous chapters. Chapter five describes research design of the study. The chapter introduces qualitative semi-structured interview as a research method and describes how it was used on the study. Chapter six concludes the empirical results of the interviews made. This chapter also constitutes the primary empirical results made in this study while reflecting them to the theoretical synthesis. The 7th chapter discusses about the theoretical and practical implications of the study and the 8th and final chapter concludes the research. The final chapter discusses about answering the research question, limitations of the study and future research opportunities.

2 THEORETICAL BACKGROUND

This chapter describes and visualizes the theoretical background of DevOps and software test automation. While DevOps strives for automation of all possible – including test automation – they are presented in separate sections to better describe the respective dimensions. Both concepts are examined in the light of existing literature and possible empirical research. This chapter was created to give a detailed description of the theoretical background and the base for the research model. The first section and its sub-sections addresses DevOps and continuous software engineering as a framework. The second section concentrates on software test automation as a separate dimension and the final section summarizes the chapter.

2.1 DevOps

This sub-section describes continuous software development and DevOps as a framework, with the first section concentrating on the first mentioned continuous software engineering and its relevant branches. The second section describes the history and rationale of the DevOps framework. Third section concentrates on core characteristics of DevOps from existing literature. The fourth section describes different approaches of similar frameworks. Fifth section describes requirements for DevOps adoption and the final and sixth section concentrates on empirical knowledge of existing DevOps literature.

A shift towards continuous software deployment creates possibilities and challenges. DevOps – a shortened version of the words Developers and Operations - is a method created to simplify these challenges. DevOps was originally created to combine software development and IT operations (Lwakatare, Kuvaja & Oivo, 2015). Data, requirements, tools, and practices must be available to all parties involved in the project. Not only do these aspects always need to be available but also easily found, delivered constantly and accurate enough that anyone within the project can work on their respective area. DevOps was also

introduced to use automated systems to reduce information gap between project team entities so processes can ensure real-time communication. (Cois, Yankel & Connell, 2014.) By integrating these aspects, the model facilitates a lean connection between different actors which have been traditionally separated. (Ebert, Gallardo, Hernantes & Serrano, 2016.) While doing so the model enables real-time communication to detect possible defects in the system developed and minimize the futile endeavour working with incorrect information.

2.1.1 Continuous software engineering

Continuous software engineering is a method which strives to quickly develop, deploy, and get quick feedback from the software produced. This sub-section concentrates on three different software engineering methods: continuous integration, continuous delivery, and continuous deployment. These methods are built to enhance construction, testing, and releasing the initial software. However, there are differences between these methodologies. While continuous delivery is more concentrated on releasing the software manually, continuous deployment aims to deliver software frequently through automated deployments. The main difference between these two methodologies is that in continuous delivery software teams can decide when to release patches of the software where in continuous deployment this step is automated to a certain point. Continuous integration on the other hand is a method where developers integrate work in the software more often so it can improve different aspects of product release such as release cycle and product quality. (Shahin, Babar & Zhu, 2017.)

Continuous software delivery is a software engineering method where the software teams produce software in short cycles. By doing so the teams can ensure that the quality of the software is good, and any version could be released whenever needed. Continuous delivery consists of a deployment pipeline which has three main components: visibility, feedback, and continual deployment. Visibility as an aspect ensures that every task and stage is visible to everyone in the project team. This promotes collaboration and transparent communication. Feedback is ensuring that the team can detect possible defects in the system as quickly as possible. This helps with reacting to requirement changes and possible bug fixes. Continual deployment enables releasing any part of the service whenever needed. By doing so the program can be inspected at any given time as well as continuously tested and reviewed. (Shahin, Babar & Zhu, 2017.)

Continuous deployment differs from delivery in a way that the deployments in this method are frequent and automated to some environments but not necessarily to the customers. The other main difference between these methodologies is that continuous deployment is a prerequisite for continuous delivery, but not necessarily the other way around (Fitzgerald & Stol, 2014). In practice this means that as soon as the developers have committed a change, updated product will automatically go to production through the deployment pipeline instead of doing the final step manually.

Continuous integration is a software engineering method where product of work is integrated to the actual software multiple times a day. As well as making the release cycle faster and increasing product quality, continuous integration also can enhance the software teams' productivity since new code is released early and often. This method has appeared as a method of eliminating waste between software development and deployment. By this definition, continuous integration as a method has a lot of same features as DevOps does as they both try to eliminate the gap between development and operations. Continuous integration is implemented in many agile software development methods which will be explained in the later chapters. Popularity of continuous integration has also been increased by many free tools which help automating the processes mentioned. Continuous integration has also been around for a longer time than continuous deployment and delivery, which might be one of the reasons of its general success. It can also be said that continuous integration is a bigger entirety which include the aspects of continuous deployment and delivery. (Fitzgerald & Stol, 2014.)

Even though there are many similarities, these methodologies cannot be directly compared to DevOps method. This is because DevOps is a much wider concept which includes aspects of organizational culture and shift. On the other hand, DevOps might be a product of continuous delivery as it is adopted within the said framework. As does continuous integration, DevOps also needs a functioning link between the departments of development and operations. DevOps and history of the framework is described in the following sub-sections with references to other similar frameworks.

2.1.2 History of DevOps

DevOps is a framework created to enhance collaboration between software development and IT operations teams. Instead of traditionally using isolated categories separating different functions and operations DevOps was created to enhance communication and delivering value swiftly among project parties. The framework was also created to enhance continuous development, decreasing problems of communication within the project team, and speeding up the problem resolution time. (Ebert et al., 2016.)

According to Zhu, Bass and Champlin-Scharff (2016) developers used to spend as much as 2 hours per day handling different builds. DevOps was partly created for minimizing this gap with continuous development. Improved internet and cloud services have made the change possible during the last decades even though DevOps as an idea has been introduced some time ago. Some agile software development methods such as Scrum were introduced at the turn of the millennium. Agile software development methods are defined as iterative phases which are followed by continuous delivery of the software. (Highsmith & Cockburn, 2001). This way different phases of software development can be derived to specific categories which makes modifying them later in the process easier. These methods also empower the customer since they are continuously

getting concrete products of the process. Nevertheless, Virmani (2015) argued that continuous integration principles in software delivery lifecycle alone are not enough to achieve efficiency in organizations. By adapting DevOps organizations can implement continuous feedback in their software development and improve the product more frequently.

Within project teams, transparent communication throughout the project might be a key factor between success and failure. Even though this part has been studied very little regarding DevOps framework Diel, Marczak and Cruzes (2016) argued that the framework includes the same communicational challenges as any other software development method. These challenges include aspects such as geographical, socio-cultural and distance related challenges. These dimensions can be seen even bigger factors in organizations that are global and spread out geographically. By automatizing specific tasks regarding communication DevOps as a framework can enhance communication between different teams. This can be seen critical especially between development and IT operations teams since these two teams have traditionally been the teams with the biggest gap between. This also helps with the problem resolution time since both teams have the same real-time information and communication throughout possible defects. (Diel, Marczak & Cruzes, 2016.)

Even though the history why DevOps was created is clear the actual definition of the framework remains somewhat unstable. Most of the definitions have same dimensions in them but a unifying definition remains vague. In the next sub-section known definitions are explained and investigated thoroughly. By doing so the benefits and challenges of DevOps framework are easier to detect. This will also help with the later stages of this paper with unifying concepts and describing the most essential ingredients of these frameworks.

2.1.3 Defining DevOps

DevOps is defined by Riungu-Kalliosaari, Mäkinen, Lwakatare, Tiihonen and Männistö (2016) as a phenomenon in software engineering which combines the traditional software engineering roles with an enhanced communication to advance production release frequency to maintain software quality. These roles include the actual software engineering, IT operations and communication in-between. According to the authors the other core characteristic for DevOps is usage of agile principles and automation to configure different deployment environments. Combination of development team's tasks such as continuous integration, deployment, delivery, and security are combined with operations' tasks such as continuous use and run-time monitoring. DevOps comprises both technical and non-technical practices that help software-intensive organizations to build responsiveness to client needs through frequent and automated software releases (Lwakatare, Karvonen, Sauvola, Kuvaja, Olsson, Bosch & Oivo, 2016). Bridging the gap between different departments and improving collaboration between them is required from a development team to function efficiently.

While DevOps is seen as change to current methods, it has also been researched from the cultural movement point of view. The characteristics for these cultural aspects are open communication, incentive and responsibility alignment, respect, and trust (Senapathi, Buchan & Osman, 2018). While the cultural changes might not be enough to define what DevOps is, they are factors that influence on rapid development processes. DevOps is a framework designed to enhance all these areas in software development and its surrounding core capabilities.

Utilizing agile principles, automation and monitoring in software development is a critical part of DevOps (Riungu-Kalliosaari et al., 2016). Properly used, these dimensions can speed up the release time and development teams can work autonomously concentrating only on necessary tasks. In their research Stillwell and Coutinho (2015) noticed that frequent releases, continuous test automation and monitoring improved a software development teams' work in several ways. By releasing frequently developers get feedback on their work and communication between a larger team is minimized. By running monitoring and automated testing developers get real-time feedback on their work and possible bugs can be noticed via automated testing. Callahan and Spillane (2016) also argued that test automation and monitoring speeds up the development process as test automation can be executed after every release which gives proper feedback on work done. These dimensions also help with rolling back a release if serious flaws are noted in the released product.

Frequent releases can help with decreasing risks that are related with deployment of the software as well as helping with quicker feedback to changes made in the software, its different setups, and different environments. By enabling fast as possible feedback organizations can help their future endeavours by managing new data which will be processed in the later stages of software development. By following the framework of DevOps, organizations can maintain continuous improvement on their software development and improve their overall software quality while eliminating waste. By doing so software development teams can discover flaws faster and easier. Finding these defects early can also help the development team's work since they can be found earlier thus creating minimized rework. (Fitzgerald & Stol, 2014.)

One core characteristic of DevOps comes from its name – combining software development and IT operations. Since implementing DevOps includes changes in the structural level it should also be viewed as an organizational change. By bridging the gaps in the organizational level and within different project teams the organization should shift toward unified processes in software development and operations. Mohamed (2015) introduced four different keys to bridge the gap between software development team and IT operations: quality, automation, collaboration, and governance. Quality aspect helps with the faster development cycle, where automation aspect ensures repeatability. Collaboration aspect concentrates on communication between different project teams and governance aspect is about blending these above-mentioned aspects to a functioning entirety. By addressing these factors organizations can enable the implementation of DevOps. These factors also help with unifying these tra-

ditionally separated organizational structures and enables agility across the whole life cycle of the service.

In the existing literature the definition of DevOps clearly has two different streams of research: dimensions that include technical and concrete descriptions like automation and monitoring and more abstract cultural movement which includes dimensions of open communication, incentive and responsibility alignment, respect and trust (Senapathi, Buchan & Osman, 2018). These social aspects are effective actors in agile software development and are required for organizations to succeed in DevOps environment. While the social aspects might not be the main defining characteristics of DevOps framework, they are so-called supporters for specific software engineer capabilities. However, these capabilities are needed for organizational success regarding software development. Learning new methods and technologies as well as feeling valued and higher level of autonomy have positive influence on engagement to the project. All these capabilities can be encouraged and enhanced by IT management which can ultimately lead to a more successful product release. These social aspects can be optimized with proper team combinations as well as empowering the employees. High autonomy and motivating collaboration have also been factors increasing the development teams' morale and overall product quality (Senapathi, Buchanan & Osman, 2018).

The defining capabilities of DevOps framework are described in table 1 below. These core capabilities are shown in three different columns with the focus on impact and possible benefits gained. The table will be explained in the following paragraph.

TABLE 1 Core capabilities of DevOps

Defining characteristic	Impact	Benefit	Reference(s)
Agile software development principles	Continuous delivery of the product	Modifying the product after version release	Lwakatare et al. (2016); Highsmith & Cockburn (2001)
Test automation	Automatic testing of new releases	Automatic reports of the changes needed, feedback	Callahan & Spillane (2016)
Continuous monitoring	Frequent feedback from changes made	Faster response time, quicker fixes	Stillwell & Coutinho (2015)
Combination of software development and IT operations	Organizational shift	Faster and more transparent communication	Mohamed (2015)

Frequent software releases	Continuous progress of the project	Decreasing risks with deployment	Fitzgerald & Stol (2014)
Cultural movement within organization	Open communication, responsibility alignment, trust	Enabling/supporting specific software engineering capabilities	Senapathi, Buchan & Osman (2018)

Each characteristic constitutes benefits which can be seen from table 1 and have their own influence on the initial software development and its final life cycle. It can be argued that DevOps is a framework that encourages software development teams to release early and often and update the product by the feedback from test automation, monitoring and customers. The framework also consists of organizational shift of combining software development and IT operations and enhancing communication within. The last part of defining DevOps is the cultural movement which acts as a supporter for specific software engineering capabilities.

2.1.4 Framework approaches

Implementing DevOps as a software development framework requires strong efforts from an organization. Organizations might have to make changes in their organizational structure to combine departments, change to different technologies and modify their habits of working (see sub-section 2.1.3). Even though there is no one standard to define DevOps we can say that the main purpose of the framework is to employ continuous software development processes to support the life cycle of agile software development (Senapathi, Buchan & Osman, 2018). Even though DevOps has claimed a lot of interest during the recent decade there is very little empirical research done about the practical parts of implementing DevOps as the dominant practice. Traditional framework of DevOps has also been used to create new trends. This sub-section concentrates on the approaches of DevOps, what are the main drivers for organizations to pursue DevOps as their functioning framework and how new trends of software development processes have been utilized in the existing literature.

It was also argued in the previous sub-sections that DevOps is a change to processes, but also a change to the organizational culture. In his research Virmani (2015) argued that organizations can decide which approaches or principles of DevOps they want to adapt. These approaches vary with desired resolutions to organizational needs, organizational capabilities, and project needs. Therefore, it could be problematic for organizations to move from not utilizing DevOps framework to full end-to-end DevOps utilization. Different capabilities for DevOps bring different benefits and organizations should plan these capabilities to fit their operational strategy. The figure below clarifies different tasks related to DevOps and different capabilities required for the framework.

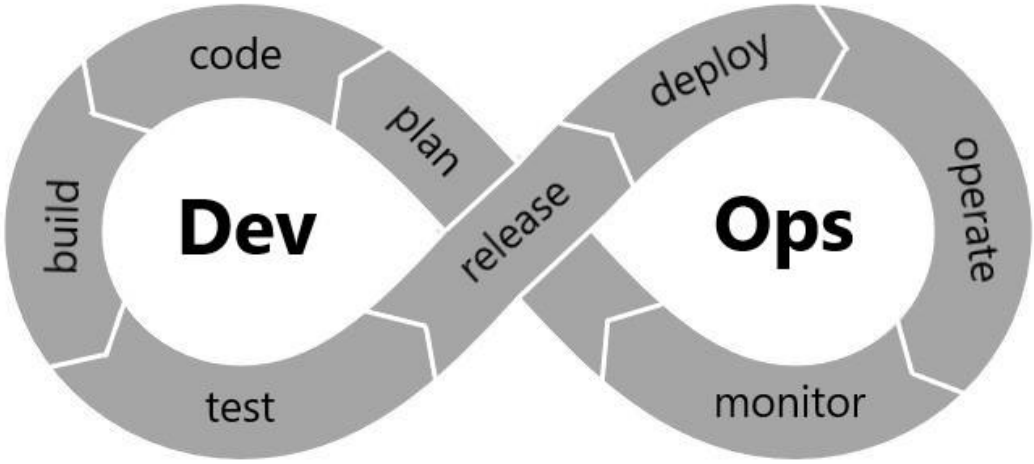


FIGURE 1 DevOps processes (Hüttermann, 2012)

Regarding its many core capabilities DevOps can be approached from different perspectives by organizations. There is more than one possible approach regarding the framework of DevOps. Even though the definition of this said framework vary there are same elements in different descriptions of the model. The above figure illustrates different dimensions of development processes and operations tasks which are critical for the framework. Code, build, and test are described as development processes while deployment, operation and monitoring have previously been only operations processes. Release and planning are done in co-operation with both parties, which makes the communication transparent throughout the process. Continuous integration is described as release in the figure, which leads to deployment and operating. Continuous feedback on the other hand helps organizations with planning and building. The figure is described as a loop because the process of real-time communication and lifecycle-wide traceability never stops. Different methods can help organizations achieve these challenges and implement the framework as their own. (Hüttermann, 2012.)

While DevOps has been an intriguing concept by organizations to adapt during recent years there are other software development frameworks that utilize the same concepts. BizDev is a framework planned to combine software development teams and business strategies while DevOps concentrates on software development and IT operations (Fitzgerald & Stol, 2014). BizDev focuses on aligning software development dimensions such as continuous integration and development aligned with continuous planning which help business strategy teams understand the requirements of development teams and vice versa. While doing so the main purpose is to enhance communication within these teams: make software development teams understand customer needs and the market while business teams learn about implementation of desired preferences. The problem of BizDev is that it is mostly seen as only a business administration development framework with very little literature found on computer or information system papers. From a computer science perspective BizDev can be defined as a framework of continuous integration of business

strategy and software development (Forbrig, 2018). Non-stop interconnection between different business departments and software development is important for successful business strategy implementation.

BizDevOps framework was created from these perspectives. The framework was introduced to not only keep the software development processes aligned with continuous use, trust and run-time monitoring but also align them with business processes such as continuous planning and budgeting. The framework enables continuous business process modelling as well as continuous requirements engineering. According to Fitzgerald and Stol (2014) these business processes are dynamic open-ended artifacts that develop with the changes to business environment. This is the main reason why integration between business processes and software development is needed as they require tighter integration in-between. With the alignment of business strategies, software development and operational tasks organizations can create a sustaining loop which stimulates continuous innovation and improvement. The figure below strives to simplify the differences between DevOps and BizDev and how they benefit each other in the BizDevOps framework.

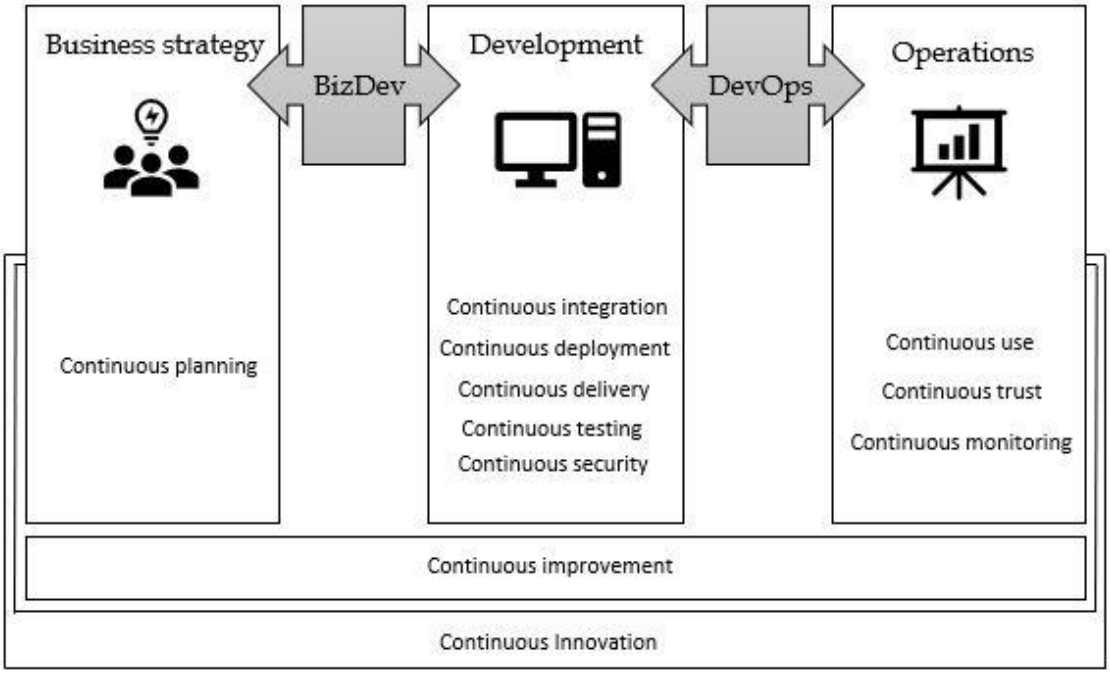


FIGURE 2 BizDevOps framework (Fitzgerald & Stol, 2014)

By combining these two different methods organizations can ensure that their business strategy is in alignment with development and operations. This way continuous improvement is created, and the framework ultimately leads to continuous innovation. However, as stated previously there are very little empirical studies of adopting DevOps. This could be due to the factors that DevOps itself is a complex process which an organization can execute in small steps. It can also be the fact that aligning business strategies with software development and operations processes can be a large, time consuming project especially for

bigger organizations. Building an operating model based on the DevOps framework can be done in several steps which makes it more difficult to study. Another possibility for the minimal research might also be the factor that organizations think they are utilizing DevOps framework while there still are major inconveniences regarding the respective operating model. That is why in this thesis empirical knowledge is described in its own sub-section. (Fitzgerald & Stol, 2014.)

2.1.5 DevOps adoption

This sub-section is focused on adoption of DevOps. The framework has been defined in previous sections as a change to organizational processes as well as organizational culture. Organizations can use the set of principles provided by DevOps in implementation, but they must decide how are they going to adapt DevOps and with what technologies amongst themselves. (Virmani, 2015.) For these reasons understanding the needed phases and overall benefits are crucial for organizations seeking to implement this framework in their business model and organizational structure. This study is focused on IT management's viewpoint which makes it even more crucial for an organization to thrive.

DevOps introduces a set of principles for software delivery that the organizations need to follow for a successful adaptation. These principles are continuous planning, continuous integration, continuous deployment, continuous testing, and continuous monitoring. (Virmani, 2015.) Continuous planning can be defined as agile and lean business plans which can react to changes needed. By getting feedback, reacting to it and adjusting business plans correctly organizations can plan their processes to support continuous product life cycle. Continuous integration from IT management's viewpoint concentrates on continuous processes to achieve automation in changes of the current build. By doing so organizations can be sure that the version of the software is always up to date. One consistent definition point of DevOps is automatization of specific tasks and assignments. By automatizing deployment companies can reduce their deployment time drastically which facilitates the work of both parties: software development team and IT operations. (Callanan & Spillane, 2016.) Automatizing testing is a requirement for continuous testing. This means the automatization of every test that needs to be ran on a continuous basis. Tests should be running on different builds automatically which reduces the ultimate deployment time. This research will process test automation in the later chapters more closely since it is the goal of the target organization. Continuous monitoring follows these dimensions of approaches and can be followed with variety of different parameters. By observing these aspects IT management can react to changes if needed, follow the evolution of the project, and get continuous feedback regarding the system.

One main aspect and core capability of adapting DevOps is merging the two traditionally siloed departments of software development team and IT operations and bridging the gap between them. Not only is this a change for or-

ganizational processes but the organizational culture too. This is one of the main definitions of the framework and it is critical for a successful adaptation. Automatization of specific tasks is one component of bridging the gap but there are several other techniques and practices to help this matter. Depending of the technologies and tools used IT management will have to establish clear guidelines of which technologies, languages and methods will be used. Because of the short release cycles technical and non-technical challenges should be dealt with extreme precision. The above-mentioned challenges could be solved by unifying working habits and technologies used, platforms such as cloud services, and deployment models. Other ways of bridging the gap are merging teams of developers and operations people to manage possible miscommunication or even establishing new specific roles for people working on bridging the gap between development and operations (Wettinger, Breitenbücher & Leymann, 2014).

By addressing these above-mentioned dimensions with fitting business needs organizations can benefit greatly from adapting DevOps. Automatizing tasks saves resources such as time and ultimately money which creates more possibilities for revenues. The resources needed for two traditionally different teams can be shared which also reduces the costs of resources. Since cloud services and other infrastructure services have become more frequent there is no need for organizations to invest on hardware. This can reduce costs but also enhance the communication within since technologies and methods are the same for all parties. While automatizing every possible task, deployment becomes a repeatable process while errors and faults are minimized. Not only does this make deployment substantially more reliable but also more efficient for all parties as they get constant feedback on tasks done. Repeatable deployment also helps with possible fixes since test are ran continuously. From IT management's view adapting DevOps is about adaptation to change and sustaining highest possible quality with the lowest possible cost. These are factors that every business area concentrates on whether it is about a software development organization or not. (Virmani, 2015; Bass, 2017.)

2.1.6 Empirical knowledge of DevOps

It was argued in the previous sub-section that the empirical study of the actual implementation of DevOps has been unsubstantial until the last few years. This can be explained with the fact that there are many phases regarding DevOps which organizations need to acknowledge and prepare for. Not only there are many challenges regarding organizational shift, adjustments in the processes and ways of working and changes in the organizational structure and strategy but also possibilities to implement only parts of the framework. This subsection concentrates on how DevOps implementation appears in the existing literature and if there are possible gaps in the research already done.

Lwakatare, Kuvaja and Oivo (2016) argued that agile software development practices are required for successful implementation of DevOps and lean

software development practices can guide DevOps implementation. DevOps is also acquired quicker by the organizations already utilizing agile and lean software development methods. This is because DevOps as a framework is based on continuous delivery and lead time which partly matches these mentioned methods. Minimizing changing technologies and methods as implementing DevOps can reduce resistance to change and uncertainty. From the existing literature benefits have been found in increased frequency of quality deployments and prolific collaboration between the two traditionally siloed teams. Actual benefits can also be found from the social actors. High autonomy, continuous collaboration and feeling valued are boosting factors for team morale. However, two things appear critical in many studies for successful implementation of DevOps: creating an automation pipeline and crossing functional organizational structures. These factors enable the benefits DevOps framework was designed for and are critical for a successful implementation (Senapathi, Buchan & Osman, 2018).

Even though the benefits of implementing DevOps have been proven to exist there are gaps in the existing literature. Lwakatare, Kuvaja and Oivo (2016) argue that the use of mostly used metrics such as deployment rate and lead time are inadequate to elect whether the effects are caused by implementation of DevOps or other similar agile software engineering approaches. The effects are quick software releasing ability, frequent software releasing ability and improved software quality. These same effects are benefits of other agile software engineering methods which makes the actual benefits hard to distinguish. This is explained with the deficiency of empirical research. While DevOps is widely seen as intriguing software engineering method most of the research concentrates more on informal discussions and less on the actual empirical study. Riungu-Kalliosaari et al. (2016) argued that DevOps might not be fitting for all industries. The cultural aspects of changing communication and working methods can be challenging especially for bigger organizations while smaller companies might be able to handle the change easier. This way communication might be lacking critical information which can ultimately lead to defects in the process. From IT managements' perspective these aspects are critical since they have effect on the whole development process. The vague, non-universal definition of DevOps is also an aspect which can lead to a false belief of utilizing the framework. Adoption of DevOps becomes difficult when companies might not know which methods to implement. The future research of DevOps should concentrate on clarifying the definition to a universal level and concentrating focus on empirical research instead of informal contributions.

2.2 Software test Automation

Test automation is one part of the DevOps framework, but it has been a part of software development long before the said framework was invented and introduced. This section describes how definition of test automation is formed, in-

troduces different types of software test automation and how it can be used to enhance software development and organizational processes. The first sub-section concentrates on the actual definition of test automation in software development. The second sub-section is about different types of software testing. Third sub-section concentrates on how the implementation of test automation can be organized and which tools organizations need to utilize these frameworks. The fourth and fifth sub-sections discusses the benefits and limitations of test automation frameworks.

2.2.1 Definition of test automation

According to Karhu, Repo, Taipale and Smolander (2009) software testing can be divided to automated and manual testing. Manual software testing can be applied when automating specific tests is not cost effective. Nevertheless, automated testing is often desired for efficient allocating of resources and enabling potent software development. In this case automated testing is automation of different activities such as development and execution of test scripts, confirmation of different testing requirements and usage of automated testing tools and methods. Automated software testing was originally created to lower costs, minimizing the amount of manual labour, and increasing efficiency and quality of software development. Software test automation can be roughly divided to two different approaches: user interface testing and API driven testing. Where user interface testing concentrates more on the interface of the software and its functions such as mouse clicks or typing, API driven software testing focuses on the programming interface. This sub-section concentrates on the definition of automated software development as a method and possible benefits and disadvantages of the framework. (Karhu et al., 2009.)

The definition of release test automation and automated testing is to present a framework which includes a standard design for test scripts and the framework should also include a support for the test driver (Ammann & Offutt, 2016). Different frameworks of test automation exist in lot of different contexts, but the main definition of test driver and test scripts apply to most of them. The test driver runs a set of tests by executing the software repeatedly and in different possible ways. This is done by automated scripts which can test a specific part or the entirety of the program. The test driver should also ultimately compare the results to the expected results of the test case and finally report these results back to the tester. Tester can then compare this feedback to the expected results and monitor how this specific part of the program conducts under different tests. By automating software testing organizations can reduce software costs. Kit (1995) estimated in his research that more than 50 percent of software development costs come from software testing. Therefore, by enabling effective methods of test automation organizations can greatly reduce costs and make software development process more effective since developers are able to concentrate on multiple processes.

Test automation was originally created to solve problems such as manual software testing being time consuming and automating software testing increasing development efficiency especially in regression testing (Karhu et al., 2009). Regression testing in software development is defined as specific tests which try to prove regression in the software. Regression occurs as the operational part of the program is intentionally terminated. Usually regression appears unintentionally while developing the software. Regression tests are executed after modifications are made to the program. A typical way of regression testing is to run tests which are made for the previous releases and see if the issues arise again. This is because in regression testing test cases are actualized iteratively after making desired changes to the software. (Yoo & Harman, 2012.) By enabling constant automated tests organizations can detect possible flaws in the program faster which helps with future releases and fixes. By automating constant tests, the functionality of the program can be ensured since specific parts are under constant testing. In addition of using automated testing scripts test automation also consists of verification of testing requirements and the use of automated test tools (Collins, Dias-Neto & de Lucena Jr, 2012). By carefully considering the aspects, quality attributes and validation requirements testing requirements can be planned and used throughout the life cycle of the service. Software test automation is planned to be iterative processes which can be repeated constantly. Creating test cases and verification of testing requirements are responsibilities of the testers. Software test automation as a framework was designed to minimize waste which is why the testers need to plan testing requirements carefully. Testers should not be automating tests that will take longer automatized thus not being cost-effective.

Choosing and using the right automated tools can also be crucial for efficient software test automation. These problems can arise if the tools are too complicated or have limited usability. Complicated tools will create problems skill-wise as they are often complex mechanisms that need elaborate technical skills to fully utilize them. That is why the test requirement should be done before selecting automated tools. The other significant dimension to consider is different commercial applications and their possible limited usability. These tools need to have preferences needed which is also possible to plan with proper test requirement verification. By properly evaluating the needed tools, implementing the tools, and giving efficient training organizations can ensure that they have the right tools for the specific project. (Wiklund, Eldh, Sundmark & Lundqvist, 2012.)

2.2.2 Different types of software testing

Software testing and automation can be done on different levels and methods. There are multiple different automated test tools, but a few definitions remain the same between all of them. Software testing is mostly concentrated on integration testing, system testing and/or unit testing. These tests are executed on different levels of the software interface and are a critical part of software de-

velopment. By testing the software developers can be sure that different parts of the program work as intended and required. Software testing also ensures that the parts of already made software works normally. Software testing can also be useful for finding possible flaws when used iteratively. This aspect helps with bug fixes and can ultimately speed up the deployment rate of the software drastically. (Karhu et al., 2009.)

In unit testing the word unit is usually defined as a basic component of the program. Unit testing of the software means testing each basic component (unit) of the software. It is used for verification that the detailed design for the said unit is accurately implemented (Zhao, 2003). Because the testing itself is done after implementing different components to the software unit testing is an efficient way of inspecting possible flaws and errors in the software. By doing so the developers can influence on the software at the earliest possible stages of the software's life cycle. The most common ways of software unit testing are specification-based unit testing which is often referred as black-box-testing and program-based unit testing, also known as white-box testing. The differences of these two methods can be found from their names. Where specification-based testing concentrates on verification of functions of the software components per an external view, program-based testing is more about the internal logic structures of the software. In both cases test results can be used to verify the efficient functionality of the software or finding possible flaws and changes needed. The limitations of unit testing are that the developer cannot possibly test every possible execution path in all the software applications. This way all bugs might not be realized and detected. That is why the program needs to be tested on a higher level. (Zhao, 2003.)

According to Leung and White (1990) integration testing is executed after the individual components and modules are combined to a working software. Unlike unit testing, integration testing is done on the module level and not in the statement level. The actual emphasis of the testing is more on interactions and their interfaces rather than on specific components and their ingredients. There are many different methods regarding software integration testing, and they can be divided to incremental and non-incremental methods. The difference between these two strategies is that incremental strategies tests one module at a time where non-incremental strategies gather the modules together and test them as a group. As well as unit testing integration testing can ultimately effect on different phases of software development. By proper testing developers can notice flaws as well as issues regarding module specifications. By enabling efficient integration testing interface can be properly tested and changes are still relatively straightforward to apply. Integration testing is limited to testing different modules and might not show the actualized results in the interface level. There can also be different flaws regarding specification issues which can slow down the development process. (Leung & White, 1990.)

System testing is applied on a complete and integrated system. System testing is used to evaluate the whole system compliance and how it meets the specified requirements. After integration testing is applied successfully the

modules can be transferred for system testing. According to Freeman (2002) system testing provides a sort of verification process of the objectives and system requirements. System testing should be approached from the user-friendly perspective since the end-user is not concerned with how the system works or responses but rather in the proper function of the system. System testing is executed to test design, behaviour, functionality and believed expectations of the user. Nevertheless, much like the previous testing strategies and methods system testing might not illustrate all possible flaws in the system since there are too many possible functionalities and execution paths. System testing can also be very time consuming and heavy to execute since it is applied to a whole integrated system.

There are a lot of different approaches regarding software testing and this sub-chapter was about introducing the most used. Automating specific tests can appear as a challenging task but it also brings different benefits. Organizations need to realize that systems and services are constantly evolving products which makes planning and execution of test automation even more critical. Organizations also need to realize which test should and should not be automated. Testing and test automation were designed for efficient use of time and resources. This means that not every test or task should be automated, only those which automation has real impact on. By carefully planning test cases and test automation companies can minimize waste as well as improve the overall life cycle of the product.

2.2.3 Implementing test automation

For a successful implementation of test automation there are few critical aspects that organizations need to consider. Test automation can fail as a project which is why companies need to plan it carefully. The planning includes software architecture, needed tools, test automation plan and clear objective planning. Adopting test automation can be a very demanding effort which is why companies need to build it around their IT and budget strategy. Planning must also include clear phases of the upkeep since test automation tools and expertise can be expensive. For these reasons test automation plan needs to be thorough and transparent for all parties and needs to be complied with as much as possible. This section summarizes which aspects organizations need to consider about implementing a test automation framework and which resources and expertise are needed.

To start implementing test automation in their strategy organizations must acknowledge the fact that it will have costs and can occasionally be very expensive. Costs must be accounted for the whole life cycle of the service since software systems are constantly evolving. By implementing test automation in their IT strategy organizations also need to think about transparency. According to Kasurinen, Taipale and Smolander (2010) test automation frameworks may vary between different business units which makes the planning phase even more crucial. Test automation may also greatly vary between different products

and, projects and services within one organization unit. By implementing test automation in their strategy organizations also need to plan which kind of tests are needed for the software project. Unit tests and regression testing are among others which are often present in software projects can be all automated to some extent. By deciding which test to automate and which not is important for the overall strategy because of the resource usage. (Kasurinen, Taipale & Smolander, 2010.)

By including test automation in software architecture organizations can ensure increasing control and reducing risks. The knowledge of test automation documentation and tasks enable minimum cost and maximum coverage network which can be utilized early and often (Amaricai & Constantinescu, 2014). This is important especially for larger scale services in which different units must be tested from the very beginning to the end of the product's life cycle. By adding test automation to the software architecture developers can keep up with the requirements and specifications easier while the system is under constant testing. Getting feedback of the tests then help developers make the changes needed which simplifies the developing process towards the end. By identifying the exact test case, test script, test data and test locators, developers can track the exact test case from the code to the desired action which helps debugging the actual product. Including test automation in software architecture ultimately needs to be documented, planned, and executed the same way the traditional software architecture: thorough and efficient. This way organizations can constantly check whether actions are executed the way they are planned to and have a proactive way of resolving different problems. (Kasurinen, Taipale & Smolander, 2010.)

There are variety of test automation tools which test automation frameworks can be built on. Like already discussed in the previous sub-section, organizations need to plan the tools to be efficient on their usability and not too complex for the developers to handle. These tools can be created within the organization or obtained from different vendors. However, since the tools have already been processed in previous sections this section will be more about tools having impact on different levels of test automation frameworks. By defining testing and test automation in previous chapters it can now be said that tests can be made in different levels of software development. Tests can be made under different interfaces such as coding interface or customer interface. The main difference between these two dimensions would be testing methods among the software protocol level and user interface level. Software tests can also be made to specific units or the whole system. By this definition, the tools can vary greatly where testers and IT management need to carefully design which tools to use. The tools may ultimately form the actual framework since they can be used to boost one another. Tools used in test automation need to be specific for every software project. This means that IT management needs to plan the tools used to match the requirements and specifications of the IT project. (Wang & Du, 2012.)

Test automation plan and objective planning require not only expertise from the testers but also commitment from IT management. Deciding the best methods and tools, planning test cases, creating the framework or choosing from existing ones and documenting and disseminating the expertise are usually time consuming, substantial expense projects. The management process of testing is described in the figure below.

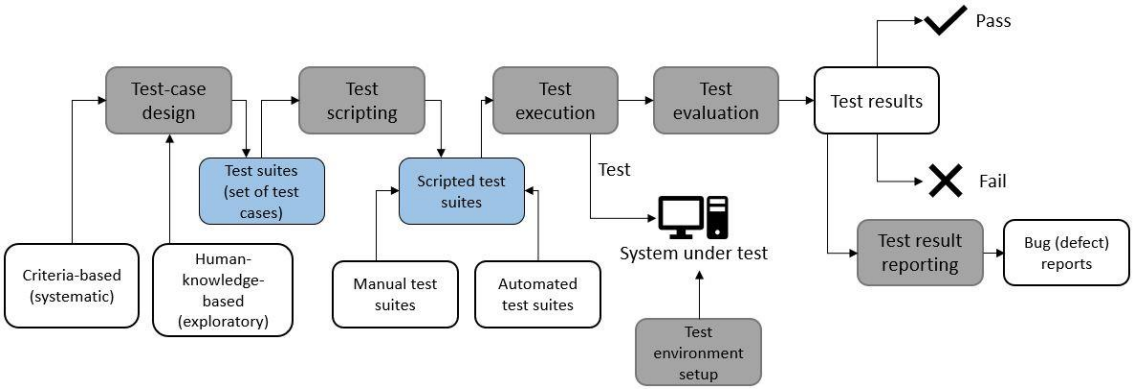


FIGURE 3 Test management processes and activities (Garousi & Elberzhager, 2017)

The figure above describes the phases of test automation plan. While most of the tasks described in the figure have manual labour in them most of them could also be partially or fully automatized. Testers and IT management will have to plan everything from test-case design to test evaluation and results to different test environment setups. Not every minor task should be automatized since test automation takes resources in the form of skills, manpower and assets. The plan described in the figure could be further continued to the end of the service's life cycle but the main idea for this section is to describe the process of implementing test automation and objective planning. Test automation plan strives to eliminate unnecessary tasks while automating these actions (Garousi & Elberzhager, 2017). IT management and testers need to design which processes and activities need automatizing while minor tasks which would take more resources automatized should be left as they are. By doing so they can ensure the most efficient way of utilizing automation in their software development processes.

2.2.4 Benefits of test automation frameworks

Test automation frameworks come with many benefits, but it also has some limitations to it. While offering a high-powered and beneficial framework for different types of automated tasks it is still a method that some organizations choose to opt out from. While offering many benefits such as repeatability, lower software development costs, reduced effort, and efficient usage of resources some organizations still turn to manual testing. In their research Kasurinen, Taipale and Smolander (2010) argued that companies drift away from test automation because it is time-consuming and, in most cases, expensive to create.

Because of these reasons' organizations should set suitable test cases to confirm functional properties in test cases which have the minimal number of changes per development cycle. This way test automation can obtain different benefits for change management and avoid different defects such as compatibility issues between different interfaces. This section concentrates on benefits and limitations of test automation frameworks and how organizations should utilize these dimensions.

Garousi and Elberzhager (2017) argued that test automation would benefit companies in lower costs and repeatability in software development which ultimately leads to reduced effort. Even though the investment to test automation can be rather significant resources such as time, effort and manpower can be saved after successful and efficient implementation of a test automation framework. Trends such as agile development and continuous integration require quick responses. Requests and specifications should be met in minimal possible time while providing best possible quality at the lowest possible cost. From the business point of view test automation should be implemented to tasks which can be proven to enhance continuous deployment and integration times. By eliminating unnecessary waste organizations can improve the products' time to market with test repeatability and enhance the products' return on investment by automating tasks.

According to Wang and Du (2012) tests ran manually are almost always prone to human errors. Applying test automation for these tasks improve product quality since automated tests can be repeated with the exact same specifications. By removing human errors continuous quality can be ensured with minimal risk factors regarding human actions. Improving product quality will also ultimately lead to faster and more cost-efficient outcome since malfunctioning services and errors cause more expenses. By applying test automation framework organizations and testers can also ensure faster response to changes needed. By getting constant feedback from monitoring the tests customer requirements and specifications can be confirmed which ultimately leads to better customer satisfaction (Stolberg, 2009). By repeating tests constantly customer expectations are more likely to be fulfilled, as well as ensuring the functionality of the service.

By applying and implementing an efficient test automation framework organization can achieve benefits on economic and organizational performance level. An efficient test automation method improves service quality, enhances the product time to market and eliminates waste. Replacing specific manual testing activities should be planned to substitute actions with high repeatability and unnecessary usage of manual labour. This way organizations can maximize the benefits gained from test automation frameworks. The figure below simplifies the benefits of test automation frameworks and which outcomes the organizations will eventually receive.

TABLE 2 Benefits of implementing test automation framework

Benefits	Outcome	Reference
Repeatability	Constant execution of automated tests, quicker response	Garousi & Elberzhager, 2017
Reducing unnecessary manual tasks	Lowering costs	Garousi & Elberzhager, 2017
Reduced effort on manual testing	Allocating resources elsewhere	Garousi & Elberzhager, 2017
Enhanced software quality	Faster time to market, improved product life cycle	Wang & Du, 2012
Quicker response to changes needed	Improved customer satisfaction	Stolberg, 2009

By adapting a test automation framework, the organization will have to commit to it in a strategical level. To achieve the benefits mentioned in previous section and in the above table are only possible to achieve if the test automation framework is transparent, scaled for specified usability and efficiently used on everyday software development.

2.2.5 Limitations of test automation in software development

Even though test automation and its different frameworks have many benefits in them organizations also must think about limitations regarding these aspects. By automating different testing phases organizations must consider resource consumption in the form of time, manpower and money. Test automation frameworks also should not be built just for automation's sake but to benefit different development and business processes regarding product development. If done indifferently consumption of financial and other tangible and intangible resources might increase to the point in which implementing test automation framework is not cost-effective.

Rafi, Moses, Petersen and Mäntylä (2012) argued in their research that there are few limitations which prevent organizations of adopting test automation frameworks. These dimensions include high initial costs of designing the test cases, acquiring test automation tools, and acquiring the expertise to efficiently use and manage these systems. In addition, IT management must consider maintenance of these test automation tools and which resources their up-

keep requires. Many test automation tools offer high risk-high reward approaches with substantial acquisition costs with very little proof of their actual benefits for specific organizations and actions. These tools are often difficult to configure and are not fitted to different software development environments. By electing to implement a test automation framework organization exposes itself to these risks which might ultimately have negative effect on business processes and organizational culture.

As was the case with designing the test suites, finding a tool with a precise and specific fit for current needs may require a great effort from the organization. Roughly half of current practitioners think that the current test automation tools fit poorly in their requirements and the systems are unfitting to be tailored for specific needs of testing (Garousi & Mäntylä, 2016). As does technology itself, also test automation tools evolve constantly. Instead of trying to cover every aspect of different levels in test automation organizations should look for tools that fit their specific requirements. The decision of when and what to automate is critical when considering success of the whole process. A lot of current test automation tools are not developed to cover a specific area of this domain but to offer as much general coverage of the matter as possible. This may not suit organizational specifications which makes implementation of these tools more difficult and sometimes vain.

Test automation frameworks are complex entireties which eventually lead to even more complex tools. Still, testers are needed to deliver functionalities and specification verifications as fast as possible. This can be problematic if the testers need to build up their own test suites and create and document the testing process. This creates a paradox of not finding the defects in the system on time as the service is developed (Gandhi & Pillai, 2014). Test automation frameworks should improve the quality of the developed service which might not eventually be the case if the testers cannot keep up with development. Testing is critical for assuring the software quality and it should not be compromised with delays in the processes. This puts even more pressure on testers which makes testing prone to errors.

Applying and implementing a test automation framework has its limitations. Organizations need to invest resources such as money, knowledge, manpower and time to a practice which can eventually prove to be even more burdensome than manual testing. Pressure on continuous integration and deployment requires constant outputs from the testers which should not be delayed by methods or tools. By efficiently researching and designing their development processes IT management can decide the best practices, tools and change management to properly implement these methods. Testing should not be automated just because of automation but to enhance internal processes and improve the overall quality of the product.

2.3 Summary

This chapter has been about defining the core capabilities of DevOps framework, different test automation types, frameworks, requirements, benefits, and limitations. Even though clear benefits of these frameworks can be found in software development area there are some barriers which organizations and their IT management need to consider. Benefits such as quality improvement, saving time and economic factors can develop in to disadvantages if test automation plan is poorly executed. This sub-section is a rehearsal of defining the core capabilities of DevOps framework, defining main points of the test automation dimensions mentioned above, and how the target organization should utilize them in their test automation strategy.

It was argued in section 2.1.3 that the core capabilities for DevOps framework are agile software development principles, test automation, monitoring, combination of software development and IT operations, frequent software releases and the cultural movement within organization. These capabilities were extracted from the existing literature as focal points of the DevOps framework. DevOps was also recognized as an operating model which can be separated to specific parts. Organizations implementing DevOps can either adapt these capabilities separately or implement only specific parts of the framework. The problem of researching frameworks like DevOps is that organizations might already be utilizing some parts of the model unknowingly or implementation of only specific parts and not the entirety of the model.

By defining different types of testing and testing models from existing literature it is possible to examine definition of test automation more closely. In this research software test automation is defined as automation of different activities such as development and execution of test scripts, confirmation of different testing requirements and usage of automated testing tools and methods (Karhu et al., 2009). Test automation in software development can be at least partially applied to different types of testing such as integration testing, unit testing and system testing and within them to regression testing. By testing functionalities testers and developers can ensure that the service works as intended and minimize the number of defects in the system. This benefits the whole life cycle of the service as continuous deployment is enabled. By doing so customer requirements and specifications can be easier to achieve and possible defects in the system can be seen quicker.

More benefits from the organizational point of view can be seen from different resources such as cost savings, time saving and faster time to market times. Resources previously used for software testing can also be allocated elsewhere which can enable multiple simultaneous projects. All these dimensions are tangible resources which can be measured and implied in the organizational strategy. Nevertheless, if test automation plan is poorly executed some disadvantages can arise from test automation frameworks. These disadvantages include aspects like high obtaining and maintenance costs of test automation

tools, insufficient knowledge of the system and poor usability for specific organizational tasks.

As a summary, test automation can greatly benefit software development teams when done properly. Test automation should include every aspect from test-case design to test results and choosing the right tools of working. At its best test automation will upgrade many different organizational activities, tasks, and resources. Manual labour can never be entirely withdrawn, and test automation should not drive for it. The purpose of these set of actions are to find test cases which are most repetitive, not to automate every task within software testing. By doing so software development teams can save critical resources and at the same time improve their service quality and minimize the amount of manual labour.

3 RESEARCH MODEL FOR TEST AUTOMATION STRATEGY IN DEVOPS ENVIRONMENT

Even though test automation and continuous testing is a part of the DevOps framework there are many regularities organizations need to consider from both entireties. This chapter introduces a reconstructed BizDevOps-model from Fitzgerald and Stol (2014) with emphasis being on business strategy, development, operations, cultural and organizational changes, and continuous software engineering methods. The first sub-chapter introduces the framework, while the following sub-chapters are descriptions of the specific areas of the theory. Second chapter describes business strategy as a guideline in building the test automation strategy in DevOps environment, where the third chapter concentrates on DevOps and recaps the main aspects of the framework. Fourth subchapter describes the test automation strategy implementation with its essential sections. Fifth and final sub-chapter summarizes the framework and describes which benefits it brings to software developing organizations.

3.1 Research model

The model itself was reconstructed from different, already established continuous software development and test automation models. Even though traditional DevOps research usually does not include business strategies and different business development methods in the framework. Nevertheless, it can be argued that the similar connection as there is between development and operations is also needed between development and business strategy (Fitzgerald & Stol, 2017). Implementing test automation strategy in DevOps environment needs to be implemented by agile methods but also recognize aspects of monitoring efficiency, value delivery and change management in business processes. For these reasons, organizations need to consider business strategies and their aspects of change management, business process management and continuous planning in the theoretical and practical implementation of the framework.

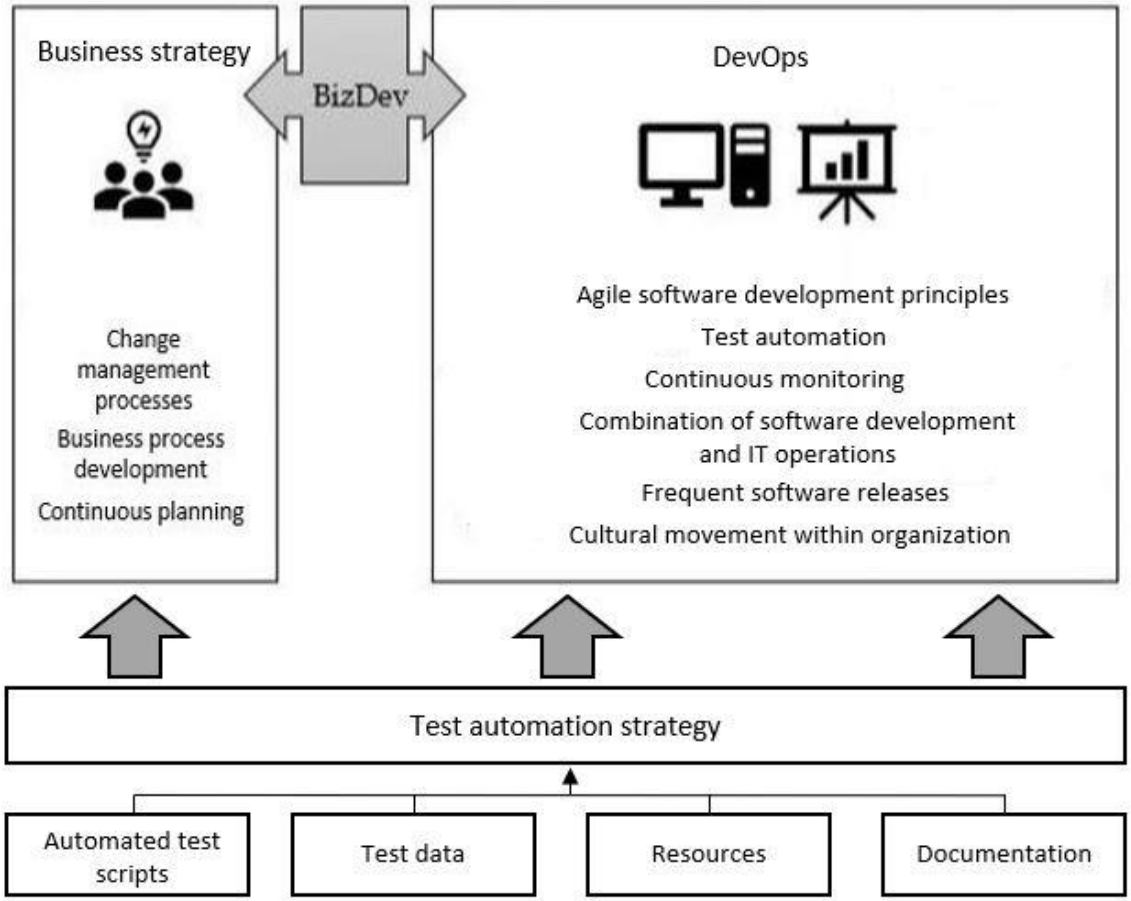


FIGURE 4 Implementation of test automation strategy in DevOps environment (derived from Fitzgerald & Stol, 2014)

This figure visually simplifies the requirements for functioning and effective test automation strategy in DevOps environment. The rationale of why these aspects are needed for efficient implementation are argued in the following sub-sections. By implementing business strategies in their test automation strategy organizations can predict budgeting more accurately and plan their test automation in alignment with business processes. DevOps relies on full automation from building the product to testing it which makes lead times shorter. Thus, automated processes need to be planned on both levels: business and development. This ensures automation of correct processes as well as enhancing business process development.

3.2 Business strategy in DevOps planning

With the implementation of business strategy to DevOps framework software developing organizations can ensure the correct processes are automated. Organizations need to understand the current delivery capabilities. Improvement of this aspect must be done by measuring business metrics such as revenue. By

measuring test coverage and deployment times organizations can see the benefits or possible disadvantages of these tasks (Fitzgerald & Stol, 2017). When test automation strategy is properly planned, documented, and executed business processes and their management will eventually change too. Since business process management is a methodology which strives to increase profitability by modeling, automating and optimizing the business processes it can be argued to have direct impact on test automation planning (Scott, 2007).

By aligning traditionally siloed departments of development and IT operations to one entirety organizations need to update their change management processes and business process management which ultimately enhances continuous planning. Regarding change management in the context of test automation strategy, the most important aspects for managers are understanding that each software process improvement is unique and the understanding of the elements in change involved (Mathiassen, Ngwenyama & Aaen, 2005). Since DevOps as a framework requires organizations to alter their organizational structure and culture change management processes can be critical for successful implementation of these methods. New technologies themselves will not change practices. Organizations and managers need to change process definitions, technologies, and practices effectively which might require a great deal of effort. For effective change management in the software interface organizations need to consider creating vision, committing to change, planning specific initiatives, operating agile and improving monitoring. While test automation can help with these objectives, managers need to figure out the most appropriate aspects of methods in leading change management. (Mathiassen, Ngwenyama & Aaen, 2005.)

Continuous planning is an entirety which includes organizational, strategic, and business planning (Suomalainen, Kuusela & Tihinen, 2015). To effectively adopt continuous planning, organizations need to plan their approaches around each section. By properly implementing continuous planning in their test automation strategy organizations can gain benefits in leadership enhancement, transparency throughout organizational context and competency development. Organizations seeking to develop their continuous planning should acknowledge each of the aspects mentioned before as all of them influence organizational structure and culture. By properly leading change management, optimizing business processes regarding automation and effectively planning organizational strategies automation processes can be designed and executed effectively. By doing so organizations can remove unnecessary manual tasks and enhance quality by removing humanly errors.

3.3 DevOps framework in the model

While test automation is included in the main capabilities a DevOps framework (see chapter 2) there are dimensions which organizations need to consider while planning their test automation approaches. The main DevOps capabilities men-

tioned in chapter 2 can be implemented separately but any of these changes will have an effect at least on organizational culture. Implementing new technologies, approaches and methods will affect organizational culture with ways of working and changing habits. Support factors such as transparent communication and trust are needed for organizations to succeed in DevOps environment. Test automation is one section of the DevOps entirety and it needs to be properly executed in the complex ensemble of DevOps framework. (Senapathi, Buchan & Osman, 2018.)

While being only one section of the framework efficient test automation still at least partly affects every capability of DevOps. By automating software testing the quality of software can be improved, number of errors in testing can be minimized, tests can be easily repeated, time used for testing can be allocated elsewhere and the actual functionality of the service can be ensured faster. Other than test automation, core capabilities of DevOps were defined as agile software development methods, continuous monitoring, combination of software development and IT operations, frequent software releases and cultural movement within organization. All these dimensions are facilitated by efficient test automation strategy and can enhance provisioning business processes. Ultimately DevOps strives for end-to-end automation, where test automation strategy is a critical piece of the entirety. (Smeds, Nybom & Porres, 2015.)

DevOps framework was probably originated from continuous software delivery method. According to Ebert et al. (2016) business strategy is traditionally seen as a separated dimension but it can be argued that an efficient implementation of DevOps takes business strategy into account. For example, merging two traditionally very isolated departments is a strategy-level decision which needs to be carefully planned. Same thing can be said about continuous and automated monitoring which can help organizations identify possible defects before they affect critical business processes. For these reasons business strategy needs to be aligned with DevOps implementation.

3.4 Test automation strategy

Organizations can have very different approaches regarding test automation strategy. Like mentioned before in chapter 3 testing can be divided to API testing and user interface testing which are two very different methods. Nevertheless, an efficient test automation strategy always should include at least documentation, resources, automated test scripts and test data. By also considering business process development, change management processes, continuous planning and other DevOps capabilities organizations can create a fully functioning test automation framework which enables automated testing for the most critical, repetitive, and cost-efficient tasks. By combining business strategy and process planning it can also be ensured that the most critical processes are automated. Poor planning and conduction of test automation strategy can ulti-

mately be less cost-effective than manual testing. (Amannejad, Garousi, Irving & Sahaf, 2014.)

The essential aspects of a test automation strategy were defined in chapter 3. To establish an efficient test automation framework, organizations need to define resources, documentation, automated test scripts and test data. Resources can be defined as economic factors as well as abstract resources such as know-how and skills. By combining test automation strategy with DevOps framework, the resources can also be distributed on organizational level within traditionally siloed departments. Organizations might need to invest on test automation tools which makes strategical planning of the test automation tasks critical for business processes (Ramler & Wolfmaier, 2006). Documentation of test automation is essential for continuous delivery and development. By thoroughly documenting software test automation phases organizations can ensure transparency throughout the product life cycle. Proper documentation can also help organizations solve possible complications on later stages of the product development. This documentation can be stored in test automation tools which makes inspecting and utilizing it easier and more efficient (Collins & de Lucena, 2012). Automated test scripts are made with test drivers for autonomous software test automation. Test scripts are made from test data, which is gathered from test documentation. Test cases are built based on the data, and they are later formed to test scripts which are ultimately automated. This method can be used to share steps with other forms of test automation (API and UI testing) which makes it a clear guideline to follow in planning test automation strategy. (Wang & Du, 2012.)

Even though test automation strategy can enhance different aspects of software development organizations need to carefully plan their test data, drivers, and test cases especially in user interface testing. User interfaces are usually non-static interfaces which are made to be operated by humans and not machines. This means changes in functionalities, changes in usability and changing contexts (Vos, Kruse, Condori-Fernández, Bauersfeld & Wegener, 2015). This creates challenges for organizations as to whether test user interfaces manually or automate tests regarding graphical user interface. This is especially a difficult task if the software is developed somewhere else. Organizations need to presume that the structure of the interface remains the same which sometimes might be a false assumption. By combining methods with business strategy organizations can ensure continuous development by automating specific processes.

3.5 Summary

Even though software development teams, operations teams and business strategy has been traditionally seen as very different departments and entities merging them as one makes sense. By doing so organizations can harness resources and enable effective communication for typically cross-organizational

business units while striving for agile development methods. By constructing and enabling a test automation strategy companies can create guidelines which to follow in continuous testing and monitoring. Test automation as a method is only a part of DevOps framework but other DevOps capabilities as well as organizational business strategy should be considered while planning test automation strategy. (Kasurinen, Taipale & Smolander, 2010.)

Implementing an efficient software test automation strategy can bring benefits such as improved service quality, reducing unnecessary manual tasks, quicker responses to changes and functionalities needed and constant repeatability of the test suites. By combining DevOps, test automation strategy and business strategy organizations can enhance their business processes and enable continuous planning regarding organizational, strategic, and business planning. These dimensions can simplify existing processes or create new process builds. However, organizations need to consider the life cycle of test automation suites especially in graphical user interface testing. Maintaining and updating existing test suites can demand lots of resources if functionalities or usability of the interface changes. By this definition organizations need to ensure critical processes which can be constantly repeated without jeopardy of user interface changes. (Wang & Du, 2012.)

Testing is a critical part of software development and one of its biggest expenditures. When done properly, test automation strategy can be cost-effective while enabling organizations to allocate critical resources elsewhere. Even though DevOps strives for full end-to-end automation not every task of testing should be automated. Organizations need to carefully plan their test automation strategy for the most critical processes which can be constantly executed and can be ensured to remain static. If done poorly, test automation strategy can occur as an expensive and unnecessary expenditure where test suites are still executed manually. By carefully planning, defining, and executing test automation strategy organizations can gain the highest return on investment while automating unnecessary manual tasks. This will also simplify continuous software development and delivery with the certainty of a functional software. (Amannejad et al., 2014.)

4 RESEARCH DESIGN

This section concentrates on the collection and analysis of empirical material of the thesis. On a more specific level this section is about how interview research and its different methods were used to find answers to the research question of the thesis. The following sub-sections describes the aim and goals of a research interview, semi-structured interview as a research method, the foundations which led to selection of this method and analysing the data gathered by interviews. This research was done as a commission for Vaisala in which the researcher worked at the time the research was done. This research was done to produce recommendations and guidelines of adopting DevOps framework and creating an efficient software test automation strategy.

4.1 Goals of an empirical research

Software test automation research has been increasing during the last years and organizations are trending towards automating unnecessary manual tasks. By automatizing software test automation organizations can gain benefits in the aspects of quality, effectiveness and performance which affect change management processes and business processes altogether. The purpose of this research is to create a guideline which organizations can follow while planning a test automation strategy in different DevOps environments. While DevOps itself remains as an unstable concept this research indicates six main characteristics for DevOps. By doing so organizational performance and characteristics of current ways of working can be measured and compared to the defined concept. With the combination of business strategy to DevOps organizations can determine the correct business processes which need automatizing and proactively react to needed alterations in change management processes. DevOps itself is a rather new concept which was created to respond to challenges of continuous integration and deployment by automating every possible task. While doing so,

DevOps is a way of reacting to constant change by streamlining software development, version control and operations. (Virmani, 2015.)

Whether DevOps is inspected from the scientific, practical, or industrial point of view it still is a rather new and unexplored framework to build software development on. The changes in organizational culture by DevOps and the benefits they bring have been the most researched topics of DevOps researches during the last few years. The purpose of this research is to combine business strategy with DevOps in the form of recognizing the change management processes needed to change and the business processes which automatization benefits. This is done by creating a test automation strategy which recognizes these aspects and takes them in to consideration while engineering the automatized test instances. Consequently, the research question of this research is defined as:

- How to implement test automation strategy in DevOps environment?

The outcome is a guideline which organizations can follow while planning their test automation strategy. Goals of this research is to highlight the most important factors which organizations need to prepare for when creating the strategy as well as pointing out the most critical business and change management processes affected by test automation framework.

4.2 Choice of the research method

The framework of DevOps which was introduced in previous chapters has been described as a model which quickens and streamlines the software development process and enhances the quality of the service produced. These goals are achieved via specific practices such as combining development and operations teams, automating all available and unnecessary manual tasks and monitoring service quality and efficiency. The defining characteristics were described in chapter two and they are the core characteristics in which this research is based on. Test automation was introduced in chapter three and the main characteristics of an efficient test automation strategy were combined from many different models and theories. The actual framework was derived from Fitzgerald and Stol (2014) who combined the framework of DevOps with business strategy.

Fitzgerald and Stol (2014) argued that software development teams and operations' team's strategy should also be combined with business strategy. This way this combination of traditionally separated business units can communicate easier and more transparently while working in the most efficient way regarding business processes. While requiring tighter connection between development and planning it also integrates continuous planning as part of the framework. By this definition plans between business and software functions are dynamic and open-ended artefacts that evolve in response to changes. By

utilizing this framework organizations can efficiently define the business processes in need of automation and how it can affect certain responsibilities such as development and operations processes. Even though there have been researches regarding DevOps during the last decade the scientific research seems to concentrate on the entirety of the framework, especially to the development aspect. Empirical research has also been minimal since adopting this kind of framework takes time and resources which makes the research intractable.

Software test automation is generally seen as a critical part of DevOps framework (Hüttermann, 2012). This kind of test automation was done long before DevOps framework was introduced. Nevertheless, it still is a logical part of a framework which concentrates on automating tasks and streamlining software development efficiency. Test automation and its benefits and disadvantages have been researched a lot but the research regarding DevOps and business processes are minimal. For this reason, it is more meaningful to research test automation strategy in DevOps and business strategy environments to reveal the true benefits and possible disadvantages of the method.

DevOps and its relations with different fields such as business strategy could be researched as a longitudinal study which means collecting quantitative information from DevOps adoption and its effects on business processes and practices. This method would require possibly years of study since DevOps can be adopted in phases and all the core characteristics might not be what an organization seeks or even tries to adopt. This is not an option because of the limitations of a Master's Thesis. Qualitative research is also more reliable than quantitative when researching a new phenomenon where the definition remains blurred and all variables of the framework cannot be defined.

With the help of research and literature introduced in the previous chapters it can be said that DevOps as a concept remains unstable and vague. There is no universal definition for the framework except the main characteristics which are defined in mostly used theories. Because of the minimal empirical research of DevOps, the relationship between DevOps and test automation and combining these two with business strategy qualitative research is the correct choice. According to Hirsjärvi, Remes, and Sajavaara (2009) qualitative research is more suitable than quantitative research especially when there is very little information about the phenomenon researched. With this definition it is more meaningful to use qualitative methods as a research method while inspecting the combination of DevOps and business strategy and building a test automation strategy around these frameworks. Qualitative research is also a more effective way to find crucial information for future research.

4.3 Semi-structured interview as method of gathering data

As it was mentioned in the previous chapters, DevOps and its definition remains unclear, especially in the scientific community. The possibility of utilizing the framework in various ways and already having specific parts of DevOps

framework adopted impede the scientific research (Riungu-Kalliosaari et al., 2016). Because the framework can be adopted in various ways and phases this research was conducted by semi-structured interview where the conversation is not steered to specific literature. The main point of this type of research is that the existing literature represents only a part of the desired research where some aspects might remain undefined or vague. Qualitative research in this research strives for finding new strategical aspects, ways of impacting current and future business processes and possible benefits and disadvantages of creating a test automation strategy.

This research was done by gathering data via interviews. The interviewees were assigned by the organization and are all professionals in different fields of IT. The synthesis was to research individual perceptions of DevOps as a model and how test automation strategy could enhance business processes and strategical organizational decision making. Qualitative research helps addressing specific limitations in scientific software engineering research by drawing single studies together and making them larger, more applicable theories (Cruzes & Dyba, 2011). This way the results are more likely to be applicable for various organizations and other specific stakeholders. Combining multiple theories is important for this research since DevOps is defined vaguely and executed in various ways. According to Qu and Dumay (2011) qualitative research, and more specifically interviews, are the most efficient ways of research when there is very little research from a specific scientific area. Since empirical DevOps research is unsubstantial, the framework can be implemented and adopted in numerous ways and the actual definition remains unclear, qualitative research is the best option.

Since the interview was done as semi-structured the conversation can be steered to what seems to be the most essential elements for the research. The interview itself is interactive and the interviewer can address the answers which seem to be crucial for the research and process the answers more thoroughly. Interviews can also be more effective as the interviewees can be reached out after the interview to clarify certain aspects. By keeping the interview in themes, the interviewees can share their thoughts, ideas, and opinions freely without the need of leaning to the existing literature. This way the current state of these dimensions is based on experience of the interviewees. The conversation remains free but is guided with certain themes that are described in the later sub-sections. For these reasons semi-structured interview was chosen as a method of gathering data. (Hirsjärvi et al., 2009.)

4.4 Themes of the interview and choosing the interviewees

As for this research the themes were divided in three different cores. The themes were created from the current state of capabilities in the organization, framework of DevOps and its core capabilities and test automation. In addition, the interviewees were asked of their background and current job description.

The interviews were done in Finnish and the whole interview and its themes can be found at the end of this research in attachment 1. The interview is explained visually below in figure 5 where questions and themes can be visualized. The conversation was free, and specific questions were only asked if the interviewee did not have any opinions or experiences of the said matter.

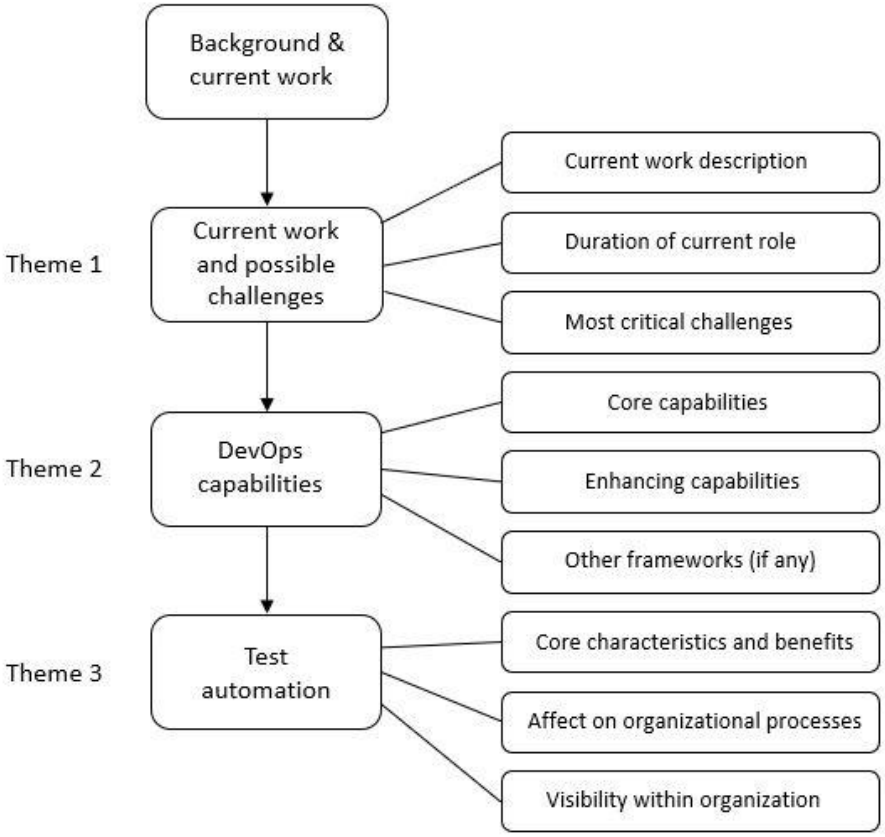


FIGURE 5 Interview themes

The first theme of the interview was about the current state of capabilities in the organization. This section processes the current work the interviewee does, how long has he or she done the mentioned work, and which are the most critical challenges that the interviewee would like to solve regarding current ways of working. These themes were addressed to gain a better understanding of the interviewees background and what kind of challenges they might have regarding organizational capabilities. The themes were later continued to understand if the interviewee has facts to back up their views or if they are just opinions. By gathering background information and current challenges the future themes can be altered by the interviewer and the actual problematic areas can be addressed.

The second theme of the interview concentrated on DevOps and its core capabilities. This section was about the core capabilities of DevOps and how they are present in the interviewees current work. The second section concentrated on if and how these capabilities defined in the existing literature arises

and how the organization could enhance possible problem areas. The theme was also introduced to find out about the interviewees attitude regarding DevOps and its core capabilities. The capabilities defined from the literature are used to set a common ground for the interview, but the interviewees can also give new perspectives which can be useful for the research and the whole scientific community. This also helps with understanding the framework in practice where the interviewees give examples of the organizational operating model.

The last theme of the interview was software test automation. Like the theme of DevOps, the main characteristics were also defined in the existing literature and used as a combination of many definitions. The purpose of this section was to figure out if test automation is already visible inside the organization, how it can be seen and if it affects organizational processes. Not all interviewees knew about how it is utilized in the organization so the conversation were then steered to how they think it should be utilized. These comments and perspectives were later used to set ground rules for test automation strategy and how it should be accounted in business processes and change management practices.

As this research was done as a commission the target organization got to decide which employees they wanted to include in the interviews. All interviewees work within the same organization. According to Qu and Dumay (2011) it is crucial that the interviewees are carefully picked for a successful qualitative research. The study assumes that the interviewees are competent and have as much possible knowledge of the matter as possible. While test automation and DevOps are only partly implemented in the target organization the results of this study should be applicable for organizations planning their test automation strategy. The research question and answers provided can be applied to smaller and bigger organizations while scaling the results with the size of the organization. The researcher was working with the organization at the time and interviewees were provided by the organization. First interview was tested with time left before the second interview. Ultimately, no changes were needed, and all interviews were executed with the same core.

4.5 Analysing the data

To simplify later stages of analysing the data gathered the interviews were recorded by the researcher. Duration of the interviews varied between 40 and 60 minutes depending on how much the interviewee had knowledge and experience on specific frameworks. Specific questions were asked from the interviewees if the conversation stopped. These questions can be seen from the interview core from attachment 1. Interviews were recorded so they could be transcribed later. Transcribing was done in two phases: first phase was to write down a transcript from the interview. This was done by simply writing down everything that was said during the interviews. The second phase of transcribing the

interviews was to divide answers gathered to specific themes. This was done to simplify answers gathered and to ease later analysation of the gathered data.

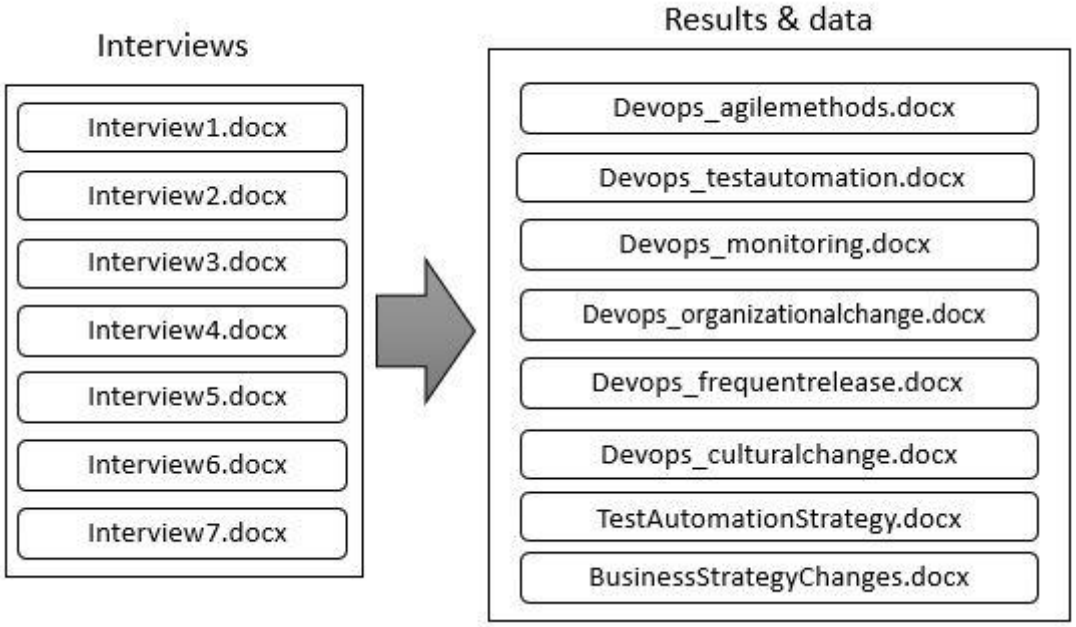


FIGURE 6 Interviews and gathered data

By dividing the answers to specific themes, the results could be handled easier. Every theme had answers from different interviewees so these dimensions could be examined from multiple angles. This was also done to combine results in more coherent way. Gathering of the data was done with semi-structured interviews which means that the answers might not be as consistent as desired. The data gathered was also used later in the research to create tables to visualize certain parts of DevOps and test automation capabilities, organizational operating models, and possible impact on strategical business processes. The two-phase transcribing was also useful since the interviewees had different amount of knowledge on specific processes, frameworks, and operating models. Thus, certain features emerged during different parts of the interview. Some interviewees also had to be asked specific questions when the interview was more structured and guided. These questions were added to gain better knowledge of existing status of organizational capabilities and hence answering the research question.

4.5.1 Case Company Description

As mentioned in previous sub-chapters this research was done as a commission for an organization named Vaisala. At the time of the research Vaisala was the global leader in environmental and industrial measurement (Vaisala.com, 2020). This means that the business is conducted by two different business areas: industrial measurements and weather and environment. The organization pro-

vides solutions that combine services, products, and applications. Researcher and all interviewees worked for the organization at the time the research was done. The researcher had interviewees and research question given by the IT management of the organization. In addition, the researcher had to create a model of implementing test automation strategy in DevOps environment since the organizational shift towards DevOps implementation was still ongoing. Even though the organization had already implemented some parts of DevOps capabilities such as agile software development methods the process was still unfinished. Guidelines and capabilities within the organization were researched to understand the bigger picture of current organizational capabilities and operating models.

DevOps framework as an operating model and specific tools enabling key capabilities such as software development version control was still an ongoing process within the organization. By conducting the research, the organization listed specific requirements of which test automation could enhance and improve. These requirements included such aspects as change management processes, business processes and cultural changes in the organization. In addition, the research was conducted by researching core capabilities of DevOps and comparing those to the current state of the organization. The interviewees were also given themes of future development. The answers varied between different interviewees which is natural when all interviewees had different job descriptions. This could be marked as a positive outcome in the results since impacts of software test automation and DevOps framework could be inspected from different angles.

The purpose of this research from the organizational point of view was to better understand the current state of organizational culture, how it could be enhanced with DevOps and software automation and what effects will these have on business processes. For this the researcher had to study current processes and how they are executed. By analysing the processes and automating specific tasks repeatedly the analysis of changes needed and possible outcomes of implementing software test automation were realized. Based on this analysis the conclusions were made and they answer the research question of the empirical study.

4.6 Interviewees and individual theme interviews

While the case organization was described in the previous chapter (see subsection 4.5.1) the interviewees are processed in this sub-chapter. The interviewees were selected by the case organization. The interviews were done personally and were recorded in one session each. Interviews were done personally to get the best possible description of the current state in the organization. The interviews and conversations were all done in December 2019 and were done on the case organization's premises. Summary of the interviewees and their background are described in table 3.

TABLE 3 Summary of the interviewees

Person	Role	Job description	Experience in current organization
P1	Solution owner, Master Data Management	Development	10 years
P2	IT Project Manager	Project management. Planning, reporting and monitoring	2 years
P3	Business Solution Owner, CRM	Software development and architecture	1 year
P4	Business Solution Manager, Sales & Collaboration	Leading development projects and small development	4 years
P5	CIO, Chief Information Officer	Broad responsibility of systems, supporting business processes and IT strategy	4 years
P6	Head of Business Solution Development	Prioritizing and scheduling, roadmap agenda with different business units	4 years
P7	Development Specialist	Acquisition of information systems, requirements analysis, development	7 years

The themes mentioned in figure 5 (see sub-chapter 4.4) were discussed in detail in the interviews. The expertise of different areas of DevOps varied which was ultimately one purpose of the research: to get knowledge and best practices to implement the method in practice. The researcher got information of current state and future ambitions from multiple viewpoints. DevOps was more familiar with the employees working in development as it was with employees working on different tasks within the organization.

First theme of the interview was a free conversation of current tasks and possible challenges regarding those. This way the researcher could pin out specific challenges and inspect these with more attention. The second theme of the interview was DevOps and its core capabilities. The description was formed from existing literature and was compared to the current state of the organization. Third and final theme was software test automation and the possible benefits or disadvantages it brings to the organization. The purpose of these themes was to create a model which functions as a guideline for implementing software test automation strategy, how it affects business processes and what benefits it can bring for the organization.

5 EMPIRICAL RESULTS

The semi-structured interview and the results gathered are reviewed in this chapter. Adapting new operating models and methods always require changes in the organization. This can be implied via new technologies and methods, but they almost always require changes in the organizational level. From the literature used in this research it seems that when adopting DevOps methodologies organizations need to align business strategy activities accordingly (Fitzgerald & Stol, 2017). Even though DevOps can be implemented in sections the change needs to be carefully planned and executed. Changes can bring significant changes in the organizational level which are discussed in the later sub-sections.

This research concentrates on test automation strategy in DevOps environment and the alteration it brings to change management and business processes. The subject is approached by utilizing the existing literature and defining core DevOps capabilities. Recent studies have argued that organizations need to align their business models within the operating model for best possible results. The first sub-chapter describes the core capabilities of DevOps and how they are currently implemented in the organization. The second sub-chapter concentrates on software test automation by analysing existing literature and results from the interviews and the third sub-chapter describes the possible modifications in change management processes. Fourth and final sub-chapter discusses the overall model created by existing literature and the results of primary empirical conclusions. This chapter will present the empirical results (EC) as well as the primary empirical results (PEC) gathered from the interviews. Empirical results are presented according to the model created in previous sections (see figure 4).

5.1 DevOps core capabilities

The theme regarding DevOps and its core capabilities was started by asking the interviewees their vision and knowledge about the method. These questions

were made to see if their vision matches the core capabilities defined by the existing literature. The theme was also used to get proper description of the current state of the organization and whether specific sections are already implemented in the organization. Even though some interviewees had more knowledge of specific aspects than others the overall knowledge of the theory and its core capabilities were recognized.

The core capabilities of DevOps which were combined from the existing literature are agile software development methods, test automation, continuous monitoring, combination of software development and operations, frequent releases, and cultural movement within organization. All the capabilities were discussed with the interviewees. Because of various roles and types of tasks handled the interviews were guided to the area of expertise which led to more specific answers about certain dimensions. In general, all interviewees defined specific areas of DevOps in the same way it is defined in the existing literature with some interviewees having more competence on specific perspectives than others. The interviewees themselves brought up aspects of agile methods, cultural change, monitoring and test automation. Nevertheless, the actual definition of DevOps remained unclear to some employees even though specific areas of it are already implemented within the organization.

The general opinion of DevOps within interviewees was that they saw the potential in the framework, even if the definition was vague for specific dimensions. Employees who worked closely on software development saw the potential in version control and streamlining the actual development process. Version control enables multiple developers on the same project and releases can be easily rolled back if needed. On the other hand, the same interviewees questioned the growth of the role that DevOps brings along. Using resources such as time and skills must be carefully planned so that developers can utilize their time in needed development processes. From business and management side the interviewees saw the potential in eliminating unnecessary manual tasks and saving resources and allocating them elsewhere. Next sections describe the core capabilities of DevOps, how they appear in the organization and how the interviewees see these aspects in practice.

5.1.1 Agile software development principles

As it was defined in the existing literature earlier in previous chapters (see chapter 2) agile software development methods are iterative phases in software development which are followed by continuous delivery (Highsmith & Cockburn, 2001). Unlike traditional waterfall methods, the phases are iterative and give more control to the customer. In DevOps this also means implementing continuous feedback in software development. Agile methods were discussed with all interviewees. The benefits mentioned in the interviews were similar of what have been found from existing literature. The interviewees summarised agile methods in the organization the following way.

> "We have adopted agile as way of working and we have some tools to surround it. The few projects that I have been a part of have had tools such as Kanban and Scrum to ease the development process. It also has a part in small development processes. As we are still in the process of adapting specific agile methods management could advance these methods even more." - Respondent P3

> "These aspects should be approached case-by-case. For example, agile is not always the best way if the requirements are very specific and the project is very small. We should implement agile within waterfall" – Respondent P7

> "By implementing agile software development methods, we need to have commitment from all different parties involved" – Respondent P2

The interviewees said that the implementation process of agile methods is still ongoing, but the effects can already be seen to some level. By doing development in small phases the releases can be easily rolled back and modified if needed. Nevertheless, some disadvantages were also seen.

> EC1: Agile software development methods might not always be the best and only way to approach software engineering and its supporting functionalities.

By doing so the development team can work on multiple projects at the same time while getting continuous feedback on the service provided. By adopting new operating models such as agile methods the organizations need to fully commit to it. Implementing the method must be specific and conciliated for specific needs. By doing so organizations can ensure continuous delivery while maintaining or even enhancing the product quality. Thus, agile methods in software development can bring benefits for not only the developer but also for the customers.

> EC2: By implementing agile development methods the quality of the service is ensured by releasing more often.

From the managerial point of view interviewees P5 and P6 both mentioned better quality and increased performance as outcome of agile methods. By enabling these methods, the organization can build a cross-functional agile framework which affects different business units. This could also help employees in sorting of their daily tasks. By adopting a cross-organizational framework organizations could enhance their effectiveness by enabling similar organizing for all the parties involved. This would also mean change in the organizational culture which is inspected more thoroughly in the upcoming sections.

> EC3: Committing all parties across the organizational borders is critical for successful agile implementation.

By utilizing the answers given in the interviews the primary empirical conclusion was formed. There was a total of three empirical conclusions from agile software development methods and their implementation. Based on these em-

pirical conclusions, primary empirical conclusion PEC1 was formed as following.

> PEC1: Improving and implementing agile and other development methods ease developers' work when properly planned and utilized according to project

By examining projects carefully organizations can ensure the best possible way of developing methods. Adapting agile software development methods within different models can be used while maintaining the best possible performance regarding to specific projects.

5.1.2 Software test automation

DevOps as a framework strives for automating all possible manual tasks regarding software development. One of these tasks is software test automation. Even though software test automation could concentrate on either API or GUI the effects and benefits of it remains virtually same. According to Karhu et al. (2009) software testing is a critical part of a successful software development and automation is about only reducing the manual tasks needed. It can be argued that without effective testing software development can face many different issues such as unfunctional service or unwanted functionalities. Successful software test automation automatizes testing and improves the quality of testing by reducing human errors. Testing in the organization was summarized in the upcoming way.

> "Especially a few bigger projects that we work on right now we are testing the same features constantly. Since we have just begun systematic testing it is still a process we need to develop. In this case basic regression testing and iterative testing which can be utilized to test the same features is needed. And I feel like this is the part where automation helps the most and makes the process easier." – Respondent P1

> "Test automation requires enough resources and know-how. The problem is that we can't test everything since sometimes the service might come from a different supplier" – Respondent P2

> "Automated testing reduces lead times" – Respondent P3

Test automation was also talked in every interview. The benefits of software test automation found in existing literature were defined in the previous chapters as repeatability, reducing unnecessary manual tasks, reduced effort on manual testing, enhanced software quality and quicker response time to changes.

EC4: Automated software testing ensures quality of the product by constantly testing the service

The interviewees who worked on the development interface saw the biggest benefits in managing the payload and reducing manual work while allocating the time elsewhere. Continuous feedback and quicker response time to needed changes were mentioned in the interviews and were essential benefits for test automation framework.

EC5: When utilizing suppliers in software development testing should be included in the requirement phase to understand testing possibilities

According to one interviewee organizations need to ensure the quality of testing. This includes aspects such as competence of the tester, quality of the tools used and integrity of the actual tests. If these conditions remain unfilled the software test automation framework can end up as a failed product which only increases workload. Organizations need to remember that software might be constantly changing which means adapting test automation to the software tested.

EC6: Regression testing and iterative automated tests are needed for successful agile method implementation

Managers saw the potential in software test automation frameworks as eliminating waste, enhancing quality and effectiveness, and improving the product life cycle. Many interviewees saw the process from development to testing often too slow and perceived test automation as a working resolution for the problem. From the empirical conclusions of the interviews the primary empirical conclusion was formed.

PEC2: Automated software testing reduces lead times and it requires careful planning, resources, and know-how.

Software test automation can bring organizations benefits such as shortened lead times and enhanced product quality. The data gathered from the interviews were utilized in the primary empirical conclusion from software test automation.

5.1.3 Continuous monitoring

Continuous monitoring is one of the critical aspects of continuous feedback which DevOps strives to enhance. By running monitoring and automated testing developers get real-time feedback on their work possible defects can be no-

ticed easier. Continuous monitoring can be used to observe defects in the system but also inspecting the behaviour of the users. This way organizations can gather data for further development as well as enhance the quality of the software. In DevOps context continuous monitoring encompasses the whole software development process from continuous integration to continuous delivery. The purpose of continuous monitoring is to increase knowledge among the development team and improve the quality of software via automated reporting. (Virmani, 2015.)

Continuous monitoring as a theme came up in two interviews (P1 and P6). While both interviewees saw the importance of monitoring it was clear that continuous monitoring is not something that was done on a regular basis.

> "Continuous monitoring is critical for continuous development. This can be enabled by continuous software testing" – Respondent P6

The actual capability of continuous monitoring was perceived the same way as in the existing literature: monitoring is and should be used to create automated reports of highlighted items. This report can be used to detect flaws during different phases of development, release, and production.

> EC7: Continuous monitoring is enabled by continuous testing

These reports can also work proactively as some flaws and defects can be detected from the reports during the development phase. Continuous monitoring can be done on different levels such as physical devices or database layers (Soni, 2015). These levels can track physical usage of the system or for example logs written by execution of different applications.

> "We have made many reports where we highlight certain errors in product data. We consider this as monitoring because we can make specific setups and see if the product data is intact or if we forgot something or made a mistake" – Respondent P1

By creating automated reports, the alerts and feedback can be made available for the whole development team. Monitoring also enables tracking of errors created in the system and how it has evolved. It was also said by an interviewee that monitoring has been done for a long time.

> EC8: Analysing the reports carried out by continuous monitoring enhances the quality of the software

This implies that monitoring is not a practice created with the framework of DevOps, but more of a procedure adapted from older software development models. As a trending software development method DevOps might have brought more interest to continuous monitoring which can be good for future software development. Both interviewees that brought up monitoring agreed that it is essential for development process. By the empirical conclusions made

from the interviews the primary empirical conclusion was formed in the following way.

> PEC3: Continuous monitoring is enabled by continuous software testing, but it ensures the quality of the product as well as verifies the requirements desired.

With an efficient monitoring, organizations can ensure the desired requirements of the service. By constantly testing the features the development team can get feedback of the functionalities and possible defects or suggestions for refining.

5.1.4 Combination of software development and IT operations

One of the six core capabilities of DevOps can be visualized from its name – combining software development and operations teams. By combining these traditionally siloed departments organizations can make the product life cycle more transparent while enhancing communication between the teams. According to Mohamed (2015) there are four critical keys to bridge the gap between these teams which are quality, automation, collaboration, and governance. These aspects require transparent communication within the teams as well as managerial procedures of blending these teams to a functioning entirety. The aspect of unifying policies and procedures were discussed with all interviewees with no exceptions.

> "We should concentrate on continuous development even more. At least we could be even more agile which requires commitment from all parties. All communication should be transparent between these parties which makes development flexible. I'm not saying that implementing agile methods is the only possible way, but commitment and transparency should be implemented by all parties involved" – Respondent P2

All interviewees agreed on one thing: integrating a way of working which is standardized and can be applied for everyone. This helps with organizing tasks and brings all parties of the development process to the same level.

> EC9: Development process is made more flexible when enabling transparent communication within the organization

While organizational tools can help bridging this gap the commitment needs to come from managerial level. By adopting an operating model which enables transparent planning and communication within different business units the organization can enhance their effectiveness and performance. Three interviewees (P2, P3 and P7) saw that communication between different business units and service providers has been problematic and it is something that has been

under improvement lately. By improving communication and technologies behind it, organizations can ensure continuous development within the project and enable the best possible working conditions. Interviewees put current state of operating model the following way.

> "Communication transparently and implementing the method requires commitment. Otherwise it might be waste of time. Everyone also needs to realize what it requires." – Respondent P3

By enabling transparent communication and unifying working methods development can be made more flexible and effective. Two interviewees saw problems in combining the two teams. As many employees work in projects, combining these two departments and making organizational changes to ways of working their workload might become unmanageable.

> EC10: Implementation of DevOps framework needs commitment from the managers as well as the employees

By enabling transparent communication and operating models across the organizational boundaries managers and employees need to commit to the change made. By utilizing the same operating model across different business units, the operative activity becomes more transparent as the operating model is unified.

> "The main point is to utilize the same operating models in operative units as well different business units." – Respondent P7

> "The cycle and handover from dev-side to operations needs to work seamlessly. By continuous deployment and risk assessment we can reduce errors as well as enhance quality" – Respondent P6

> "DevOps and agile development methods are utilized in cycles. We must make sure developers have the time for these projects" – Respondent P1

By adapting DevOps model organization needs to commit to its working ways. Even though DevOps itself can be modified to organizational needs there are specific requirements for successful implementation of the framework. Commitment from all organizational levels and transparent communication seem to be the most critical requirements. DevOps as a framework also requires more responsibility from developers. Unmanageable workload can be solved by the managerial staff by careful planning of current and future development projects.

> EC11: Adapting DevOps model the tasks carried out by developers might change and increase the workload

By adapting DevOps and agile software development methods the managers also need to ensure that the developers have the correct amount of time for each project. By increased workload developers might not be able to do all tasks allo-

cated for them in time which could decrease the quality and lead times of the project. By extracting the empirical conclusions from the interviews, the primary empirical conclusion was formed.

> PEC4: DevOps framework enhances communication between business units, requires commitment and increases the quality of the overall development process.

Adaptation of DevOps framework requires unified ways of working and transparent communication throughout the organization. Having different operating models within the organization creates barriers such as scheduling problems as well as decreasing performance quality. By adapting to the model, the continuous loop of deployment, integration and delivery is enabled which can benefit organizations in many different ways.

5.1.5 Frequent software releasing

Frequent releasing and version control were talked with the interviewees that work in development interface (P1, P3 and P7). By frequently releasing and utilizing version control organizations can minimize risks with deployment and easily roll back features if needed. Version control also enables developers to work on multiple instances at the same time which makes developing process more productive and efficient. Fitzgerald and Stol (2014) argued that frequent releasing can help organizations with continuous improvement with enhanced quality. This can also help with minimizing waste, which was mentioned in many interviews (P2, P4, P5 and P6).

> "We are looking at the different tools to enable efficient version control. This would help us with rolling back features and finding specific modifications if needed, especially on projects with multiple developers. Now we have developers that work on the same code and there is not a way that we could do this which ensures quality and efficiency. That is something we are working on right now." – Respondent P3

> "DevOps-based tools help with automated testing features" – Respondent P1

By enabling agile development tools, the developers are enabled to carry out specific tasks. DevOps as a framework concentrates on automating specific tasks which allow developers to inspect feedback and changes easier.

> EC12: Agile development tools such as version control help developers see the changes made and release more often

Enabling specific software development related tools such as version control can enhance organizational performance. By enabling this method multiple de-

velopers can work on same code while ensuring quality and efficiency. All three interviewees agreed with existing theories that by releasing frequently more data can be gathered of modifications made which results in more feedback. While doing so, tools like version control help minimizing tasks as it can be fully automated.

> "By frequent releasing we could minimize waste in development processes" Respondent P4

While tools were also discussed with these interviewees, all of them agreed on using specific development methods for specific needs. One interviewee mentioned that there are times that development might be done for three months without testing. According to the interviewee this is does not fall into category of agile software development methods and needs better planning as many developers work on multiple projects at the same time.

> EC13: Frequent releasing eliminates unnecessary waste

By frequently releasing the development team can get information on defects and requirements. Frequent releasing helps with future development processes with more frequent feedback. While implementing continuous planning organizations can ensure that employees have time for different projects and can plan their future work accordingly. Of the empirical conclusions formed from the interviews the primary empirical conclusion was formed the following way.

> PEC5: Frequent releasing with correct tools help developers see changes made, react to possible alterations, and eliminate unnecessary waste.

Enabling specific DevOps based tools such as version control helps developers with seeing the made changes, rolling back the releases if needed and enabling multiple developers on a same project. Frequent releasing also ensured that the requirements of the service are met as well as gives developers feedback. This eliminates unnecessary waste since developers can work on development processes which are required and defined earlier in the process.

5.1.6 Cultural movement within organization

While organizational culture is hard to define since it contains everything that happens within the boundaries of the organization. Senapathi, Buchanan and Osman (2018) argued that there are four key dimensions which enable cultural shift in organizations. These dimensions are open communication, incentive and responsibility alignment, respect, and trust. If frameworks like DevOps are desired to be implemented these aspects need to be embraced by management

and staff. Fully committing to the organizational change and its surrounding aspects organizations can achieve benefits such as increased performance and well-being of employees. Organizational culture and its changes were discussed with three different interviewees (P2, P3 and P5) and one of the interviewees described current organizational culture the following way.

> "We have a relatively small organization. Nevertheless, we are missing tools of enhancing productiveness. The problem is that we still see the organizational culture as someone accountable for developing and someone accountable for operative side. This is something we must work on, and we must include different business units in our operating model and daily work. That requires changes in the organizational structure too." – Respondent P5

> "DevOps requires organizational change in habits and policies" – Respondent P3

Even though open and transparent communication can ease the cultural shift the changes need to be executed by management. Increased respect and trust can arise in the form of allowing different working methods and empowering the employees. Still, the initiative needs to come from management and the methods need to be substantiated for employees, starting from refining organizational structures if needed.

> EC14: Unifying the operating model increases productiveness.

This way managers can create trust within the teams and organizational culture can be changed. Organizational innovation and performance can also be enhanced by empowering employees. (Senapathi, Buchanan & Osman, 2018.) All interviewees agreed on the aspect of adopting frameworks like DevOps a shift in organizational culture is required.

> "Our clients are usually inside our organization and this requires commitment from all parties. This doesn't always happen, and it usually shows as a stretched schedule" – Respondent P2

One interviewee pointed out that at least in one project the resources are scattered within different teams. This deteriorates communication and might have negative effects on the performance of the project. All interviewees agreed with existing literature on the fact that the business units and different roles need to be combined as a working entirety in which transparent communication and common operating models are shared and used. By doing so organizations can improve the continuous planning and employee well-being while increasing productiveness.

> EC15: Unifying the operating model enhances transparent communication

By adapting to unified operating model across the organization development teams enable specific benefits. Cultural and organizational change needs to be justified from management and employees need to see the benefits regarding

the change. From the empirical conclusions the primary empirical conclusion was formed.

> PEC6: Cultural movement towards DevOps within organization enhances productivity and unifies the way of working.

By setting policies and altering the organizational structure the movement in organizational culture can be structured. By enabling supporting software engineering capabilities such as trust managers can ensure that the shift in habits and procedures are made as easy as possible. Organizational culture always varies, and managers need to find the best practices of implementing the structural change.

5.2 Test automation strategy

Even though test automation has been a part of software development longer than DevOps as a framework has existed it still is a crucial part of the model. Existing literature introduces DevOps as a model where every possible manual task is automated regarding software development, which includes software test automation (Fitzgerald & Stol, 2014; Ebert et al., 2016). By utilizing test automation organizations can gain benefits such as increased productivity and improved cost-efficiency. With a proper test automation framework organization can ensure continuous software testing and minimize errors in testing. The framework also helps with monitoring with automated feedback of the development process.

The case organization has had some test automation executed within different business units it is not used within IT development. All interviewees working in development interface mentioned that software testing is something that has been just recently systematically started. These interviewees also mentioned that all testing is currently done manually which might take a lot of time and can include errors at times. In addition to automated test scripts test automation strategy was argued to include test data, resources, and test documentation. Test documentation defines use cases while test data can be argued to include both documentation and the logs after test execution. Resources are defined as economical resources and skills. By acknowledging these dimensions organizations can set up a functioning guideline for test automation strategy. (Wang & Du, 2012; Collins & de Lucena, 2012.)

The interviewees mostly agreed with existing literature about test automation strategy and creation.

> "If we take for example Salesforce. We already have the materials which can be utilized to create standardized flows which can be automated. These are something that

> need to be done continuously at least 3 or 4 times a year (automated Salesforce release cycle) or maybe even more." – Respondent P1

> "We have talked about this within the organization and we should create good enough tests during the project phase that they could be utilized later as automation for regression testing. Of course, our service providers have created these types of test for their unit, system and integration testing but that is done out of sight from our point of view." - Respondent P2

By these definitions it could be argued that software testing does not currently have defined guidelines to follow. These definitions can also vary regarding the type of testing is done.

> EC16: Test automation strategy and test suites should be included in the requirement phase

By doing so the developing team could inspect the tests needed. This is especially important if the software is also developed outside the organization. The main difference between API and GUI testing is that user interfaces are often non-static interfaces where changes are made in functionalities, usability, and contexts (Vos et al., 2015). Salesforce is a cloud-based software service which provides customer-relationship management (CRM) services. Challenges in testing a software like Salesforce is that it is developed elsewhere, and the organization must presume that the interface remains the same. If the service provider decides to alter functionalities of the system, the automated tests can ultimately fail.

Resources regarding test automation strategy can be extracted to economical resources and skills. Tools which enable software test automation might be expensive which requires careful planning of business processes and tests to automate.

> "The resources we have right now - which are skills, people and time – we possibly can't implement test automation everywhere. We need to acknowledge who it concerns and who is responsible for it. Defining test cases is a big project and we need to decide who is responsible for it" – Respondent P2

By enabling needed resources and defining responsibilities organizations can create transparency throughout the development process. All interviewees agreed on the fact that test documentation is critical for successful test automation strategy.

> EC17: For an efficient test automation strategy, organizations need to identify the most critical processes to automate

By creating effective test cases during the requirement process organizations can already start planning their test automation strategy and test suites. This also enables effective use of test data since data for test cases can be extracted from test documentation. From managerial point of view one interviewee men-

tioned that metrics of tests created, time used in testing and other relevant indicators need to be measured while planning test automation strategy. By allocating resources, creating effective test documentation which ultimately leads to competent use of test data, organizations can create automated test scripts which are defined in the requirement phase and remain the same throughout the life cycle of the system.

> EC18: Allocating economical resources, manpower and skills are critical in creating an efficient test automation strategy

Organizations also need to realize the constant evolution of the software which requires careful software test automation planning. By proactive planning organizations can plan test cases which are more likely to stay static and require minimal changes in actual tests. The focal point is to create tests which are crucial for business processes and stay intact so they can be repeated time after time. The primary empirical conclusion was formed the following way.

> PEC7: Software test automation strategy should include the requirement phase, include most critical processes, and be a strategical guideline of allocated resources, manpower, and skills.

By following the steps mentioned before organizations can create a strategy which includes the most critical processes while maintaining maximum cost-efficiency and performance. Even though the process might differ depending on the size of the project and organization included these steps help organizations in creating a strategical framework of software test automation.

5.3 Business strategy as part of the framework

Implementing an effective test automation strategy will induce changes in change management processes. Todnem By (2005) defined change management in organizations to be critical in order to survive in constantly changing environments. Even though change management processes can vary between different organizations the processes should be bound to organizational strategy and vice versa. Organizations can and should plan their change management processes accordingly but the need for change can often be unpredictable. By effectively implementing test automation strategy in organizational change management processes companies can proactively design automation of currently manual business processes.

Organizational change management processes were discussed with all interviewees. The interviewees saw testing and more precisely test automation as

a process which could impact change management processes in various ways. One interviewee mentioned that in projects requirement phase is done in advance, so it resembles more traditional developing methods than agile. The changes are later managed with change management processes if needed. If test automation is implemented in the requirement phase as creating functionalities and test suites the actual process can remain the same.

> "For change management processes it's essential that test automation helps with shortening lead time. It can also help with improving quality since tests are automatic, repetitive, and deducted from human errors. Of course, it can also backfire if test cases are poorly planned. This is something we need to integrate in our tools. If requirements are created carefully, they can be easily transferred to test cases." – Respondent P2

By shortening lead time organizations can improve their cost-efficiency and improve commitment to the project. The most critical tests can be automated while reducing manual work and allocating time in other projects.

> EC19: Business processes and change management can be affected by software test automation

From managerial point of view one interviewee mentioned that if test cases and suites are created in the requirement phase testing could occur throughout the project, and not only at the end of the project. This would ensure proper testing throughout the product life cycle and enhance quality of the service provided. The same interviewee also mentioned that with current resources manual testing throughout all these phases is not possible since the organization lacks resources to do so. By these definitions it could be argued that implementing a proper test automation framework would ensure product quality in planning and designing phase and enable continuous delivery by constantly testing desired functionalities.

> "It is critical part of change management and quality control. Currently we are testing software manually. Usually test resources are the main challenge regarding testing. In our case it means that the more we can automatize testing, the more we are testing. So, you could say that it is a very critical part of our change management process, so we can make changes and see the possible defects as soon as possible." – Respondent P3

The effect of allocating resources can also affect change management processes. Allocating resources to create efficient test cases in requirement phase can reduce manual work, allocate time and effort elsewhere and enable continuous software testing. This could also be defined as continuous planning.

> EC20: Continuous planning is a logical part of a DevOps framework

One interviewee mentioned that DevOps based tools such as version control would bring transparency and control to developing work. This way organiza-

tion can ensure overlapping development process and create test cases more effectively. This would need a clear definition of processes that need testing and the affect it will have on other processes.

Generalization of test automation and its effects on change management is difficult because the practices might vary. Both test automation and change management can be done differently depending on organizational context, habits, and the actual product. Organizational change management is bound to organizational strategy and it also includes management of different business processes. Utilizing the empirical conclusions from the interview the primary empirical conclusion was formed.

> PEC8: Business strategy, change management processes and continuous planning should not be separated from the DevOps framework.

The purpose of this section was to highlight change management processes in the target organization and see which the employees saw most important. Since adapting methods like agile development is still an ongoing process in the organization the true effects of test automation strategy regarding change management processes are still speculation. Organizations also need to remember that poor planning in test automation strategy might ultimately inflict more costs than manual testing, which is why it is especially important in organizations where software is also developed elsewhere. (Vos et al., 2015).

5.4 Summary of primary empirical conclusions

Combining the development process within business units and enhancing transparency and communication between these units is essential for effective software development within the boundaries of changing technologies. Software test automation strategy can help organizations enhance their development processes and business processes with increasing quality and efficiency. Nevertheless, companies need to plan the most effective ways of utilizing software test automation especially if the software is also developed elsewhere. The same can be applied to GUI testing, where interfaces are planned to be used by humans. The interview data was first used to form empirical conclusions (EC) and afterwards primary empirical conclusions (PEC) of the current knowledge. Ultimately there were eight specific primary empirical conclusions extracted from the interview which are visualized in this section.

The eight primary empirical conclusions from the interviews are visualized in the below table. These conclusions can be applied to the framework of creating a test automation strategy in DevOps environment from the perspective of the target organization. Even though changes in current operating mod-

els might always be required within organizational level, organizations might already have resources of DevOps and automation aligned. As stated in the existing literature DevOps as a framework can also be implemented in small phases by adopting different pieces at different times.

TABLE 4 Primary empirical conclusions

PEC No.	Primary empirical conclusion
1	Improving and implementing agile and other development methods ease developers' work when properly planned and utilized according to project.
2	Automated software testing reduces lead times, but it requires careful planning, resources, and know-how.
3	Continuous monitoring is enabled by continuous software testing and it ensures the quality of the product as well as requirements desired.
4	DevOps framework enhances communication between business units, requires commitment, and increases the quality of the overall development process.
5	Frequent releasing with correct tools helps developers see changes made, react to possible alterations, and eliminate unnecessary waste.
6	Cultural movement towards DevOps within organization enhances productivity and unifies the way of working.
7	Software test automation strategy is an entirety which should be included in the requirement phase, include most critical processes, and be a strategical guideline of allocated resources, manpower, and skills.
8	Business strategy, change management processes, and continuous planning should not be separated from the DevOps framework.

Table above illustrates the primary empirical conclusions of DevOps core capabilities, software test automation strategy, and combining business strategy with the model. These conclusions were made from the interviews in the target organization by analysing the data given by the employees. DevOps itself is a large entirety and the target organization is implementing the framework in

smaller steps. This is visualized in the target organization in the form of agile development methods which are currently in the implementation phase. While software test automation enhances quality and monitoring it also allows organizations to allocate resources elsewhere. On the other hand, automation in target organization is still a future process which this research offers foundations on. With the adoption of other DevOps capabilities organizations can enable continuous integration, delivery, and planning. Business strategy can also be argued to have a critical role in adopting DevOps environment since business and change management processes might be affected by automation.

6 DISCUSSION

DevOps as a framework is still in its infancy and the actual definitions of the framework vary. Nevertheless, there are still benefits from the framework that can be measured. This section describes the results gathered from existing literature and the semi-structured theme interview and compares the results to the theoretical synthesis. This chapter concentrates on theoretical and practical implications of implementing a test automation strategy in DevOps environment. DevOps as a framework requires changes in organizational culture, skills, roles, and tasks handled. Core DevOps capabilities were defined from the existing literature. From the automation point of view this research concentrated more precisely on software test automation and provides strategical dimensions of implementing a framework for the target organization and companies alike. Primary empirical conclusions (PEC) defined in the previous chapter are used in this chapter to form theoretical and practical implications of the research. These conclusions can be visualized from table 4 (see sub-section 5.4).

6.1 Theoretical implications

Even though actual empirical study of DevOps and implementation of the framework is minimal this research provides benefits and challenges regarding implementation of test automation strategy in DevOps environment. The synthesis this research was built on was derived from Fitzgerald and Stol (2014) and the foundation was to define core capabilities of DevOps and argue that business strategy needs to be aligned with the framework to create an efficient software test automation strategy. Connections between different business units might be required to create an efficient test automation framework. This is visualized by changes in business and change management processes. By inducting a DevOps framework, organizations can enable continuous integration and delivery where the cycle of releases can be adjusted. Existing literature has argued that DevOps framework is dysfunctional with traditional software devel-

opment methods because of its continuous nature. Thus, organizations need to implement agile or lean software development methods to build their software development on for efficient test automation strategy. Nevertheless, from the empirical conclusions (PEC1) of this study we could argue that organizations can make so called hybrid models where agile development methods are included within traditional methods. When planned and implemented correctly traditional software development methods can be used for requirement phase, especially if the project is very small and specific. This way the project itself stays intact throughout the process. By utilizing agile methods in the actual development work the continuous nature of the project preserves while having static requirements throughout.

In their current operating model, the target organization has been implementing agile software development methods, but software testing is still manual. Implementing agile software methods is crucial for a functioning test automation framework for continuous monitoring and delivery. By adapting these methods software development teams can ensure fast enough release cycle to match the definition of requirements (Lwakatare et al., 2016). In target organization customer satisfaction can often be measured within the company which makes inspecting metrics more convenient. DevOps practices are designed to enable production of quality components regarding software development. By building an effective software test automation strategy the target organization can gain benefits in many different business areas while streamlining their own business processes and change management. The empirical conclusion (PEC2) extracted from the interviews was that building a test automation strategy can cost the target organization a lot of resources if planned poorly. The actual model should be derived to include the most important business processes and build the framework from simpler processes up. By planning test management processes carefully organizations can ensure streamlining and transparency of the process while documenting the process thoroughly (Garousi & Elberzhager, 2017). This helps organizations in cases of changes in employments since all processes are documented transparently.

To modify their operating model to one fitting the description of DevOps procedures the target organization needs to concentrate on implementing the model on a more practical level. While a test automation framework can provide many benefits, not all procedures and process should be fully automatized. Organizations need to consider which processes they should fully automate, and which processes need to have automated scripts but are used as semi-automated for regression testing. This means running the automated test scripts manually. This can be especially helpful in GUI testing where changes in the interface are not as frequent as they might be in the API interfaces. With this definition, the target organization needs to have a strategic plan for test automation framework and processes in need of automation. This can be prepared by following the test management processes and decision making in the strategical level.

As existing literature describes, DevOps is a rather new framework which affects organizational structures. The operating model effects on skills, roles, tasks, and the organizational structure itself (Virmani, 2015). This change requires commitment from not only employees but the whole managerial and organizational level. While automation plays a big part in a framework such as DevOps this research found no reasons from the existing literature why development and operations teams should be separated from each other. By committing employees to work towards mutual goal organizations enhance communication within. Nevertheless, the traditional role of developers and operations change along the framework. These departments and their employees are now expected to have technical and practical skills as well as good communication and social skills. An empirical conclusion (PEC4) can be used to argue that the implementation requires changes in the actual structure of the organization as well as habits within. This can increase the workload of developers especially in the project natured operating model which the target organization utilizes. The allocation of resources of manpower needs to be examined and allocated so that developers have the time allocated for different projects. While development and operations teams traditionally have had their sights on different goals the strategical leadership is needed to change organizational culture and committing employees to the operating model.

Even though DevOps framework can be accumulated piece by piece organizations need to strategically plan the starting points. Traditional software development models are unfitting to a framework like DevOps, but exceptions can be made. Smaller projects might have clear requirements and steps which makes adding agile methods unreasonable. However, in these cases organizations should plan their test automation processes from the requirement phase. This way the strategy can be planned during the life cycle and the actual software can be tested throughout. This requires pervasive understanding of the processes and products to recognize the possible defects and indicate them to the correct direction. The framework requires untraditional thinking when it comes to software development, skills, and organizational ability to change and learn. (Ebert et al., 2016.)

The theoretical synthesis provided in this research is founded on one target organization and its endeavour to implement a functioning DevOps and test automation strategy. The framework build should be possible to implement in organizations in need of DevOps and software test automation framework to build their operating model on. As it was stated before, DevOps operating model can be applied in different ways which makes it unique in every organization. The implementation of the model requires supporting software engineering capabilities. The primary conclusion (PEC6) can be extracted for theoretical implication in a way that for enhancing productivity and effectiveness within the organization supporting capabilities such as trust and responsibility alignment need to be enabled. This can also require modification in habits and policies. Enabling unified ways of working can be critical for companies like the target organization where software is also developed outside the organizational

boundaries. In these cases, the emphasis should focus on considering the effect of agile software development methods, inspection of processes in need of automation and streamlining the manual tasks needed in services provided. However, roles, skills and organizational structure can vary greatly between different organizations which might make the results of this research un-applicable for every organization.

6.2 Practical implications

A software test case was built to be a practical part of this research. The test case was done as an automated test script which the researcher executed manually to see if there are any defects or exceptions during or after the test. The test case was built using software UIpath and was conducted in Salesforce test environment. Accesses to both resources were provided by the target organization. This test was built as a GUI test which tested a process that can be seen in attachment 2. The process was simply to create a new account in Salesforce CRM and afterwards ensure that the data was migrated to the enterprise resource planning system used by the target organization. The process was executed 50 times to see how the service handles the script and to get a bigger sample of data to verify the observations. This sub-section also includes the practical implications made from the primary empirical conclusions.

The actual use case of the test created is visualized in attachment 2. The use case consists of logging in to the service, navigating to new accounts, creating a new account with specified and mandatory information, adding an address, saving new account, and closing the application. The use case scenario describes the actual action which is required to create a new account in the service. After executing the use case scenario, the script also verified that the account was made, and the new account was migrated to the ERP test environment by confirming the reference number. This way the actual creation of the new account and the data transfer can be confirmed which can be used as a criterion for a successful account creation.

This test was executed 50 times as a semi-automated process in which the researcher executed the automated test scripts manually. The results can be examined in attachment 3. The test failed three times and when examining the reasons, the script failed because of some add-in dysfunctionalities which are needed to run the web-based service. Tests were executed in a web browser (Google Chrome) since the service is web-based. These errors could be evaded by adding exception handling or specific try catch blocks. However, this was not done to get a clear picture of the success percentage of executed tests and the possible reasons for failed tests. in 47 successful tests the new account creation worked normally and the migration to ERP system in test environment was successful. If fully automated, exception handling could be added to the script to handle errors such as add-in problems by restarting the process. In this case it would probably have enhanced the overall success rate of the process. Never-

theless, the success rate of the test script was 94 percent which is a good result even without any exception handling.

The practical section of this research was done as a graphical user interface testing. This means that the test followed the path of an actual human user and repeated the steps that a user would go through. A practical part of the research was also to do the same test manually 10 times and compare these execution times to the ones the scripts take. While done manually, the steps required for a successful process took 82 to 86 seconds which included all phases mentioned above. While executed with automated scripts the tests took 8 to 10 seconds depending on how fast the migration went through. With this result it could be argued that the test took only 10 percent of the time which manual process requires. With this argument and practical implications extracted from the empirical conclusions (PEC5) it could be argued that with an efficient test automation planning and execution the developing process converts to more efficient. This might also make the developers' work easier as feedback from the tests can be used to evaluate the process. Even though this was only a very small process to automate the benefits are already visible in the results. This process could also be combined with different automated processes to construct entire process builders. The information could be further processed with modifying the customer or examining a customer offering or incident pipeline. The organization needs to focus on the most important processes which are static and more likely to stay intact. This way the success rate of the test scripts is maximized, tests can be repeated, and the test scripts are more likely to succeed.

Even though there are clear benefits in GUI testing, organizations need to remember that the functionalities of applications and services can and probably will change. The test suites need to be planned the way that they are easy to modify and can react to changes made in the user interface. By this definition, continuous monitoring needs to be executed for the maximum benefit of test automation. The data extracted from primary empirical conclusion (PEC3) indicates that for effective continuous monitoring software testing needs to be done. Logically if automated, testing can be repeated more and with higher precision. The changes made in functionalities are usually noticed in API testing which might be a different case in GUI testing. This especially important in cases where the software is also developed elsewhere (eg. Salesforce). Test suites also should be planned with change management processes in mind. If a specific process includes various automated test suites change management regarding process management can become challenging. By extracting data from the empirical section of this research (PEC8) it could be argued that business strategy, change management processes, and continuous planning should be included in the target organizations DevOps framework. By changing the preferences or features in the service, the test suites also need to be updated. This requires resources such as skills and time. When done properly, the changes can be easily modified while streamlining the actual service and iterating continuous planning as a part of the framework.

If these conditions occur software test automation can benefit organizations greatly. This research had a practical section of GUI testing and the results might not be directly comparable to API testing. Nevertheless, there are similar attributes in both testing activities. The future research of GUI testing could focus on building more complex process builders and compare the times between manual and automated testing. Implied results of the theoretical synthesis seem to follow the guidelines constituted earlier in the research while it was put in practice. The metrics of saving time, resources and improving quality can all be measured from the practical example. While this practical example was done outside of the service developing timeframe the same test scenarios can be used as regression testing to see that the previously defined requirements are fulfilled. A practical implication can be extracted from the empirical data (PEC7) as software test automation strategy needs to be carefully planned and executed to be effective. When poorly planned, automated tests can take more time and be less cost-efficient than manual software testing. Theoretical framework of planning a test automation strategy and measuring its benefits can thus be realized and argued to have positive effects on business processes.

7 CONCLUSIONS

Constantly changing services and technologies require quick reactions and preferably proactive actions from organizations. Business environments and markets change constantly, and organizations need to modify their operating models to get competitive advantage. The core capabilities are needed for quick reaction to possible defects as well as continuous development and delivery. Modifying organizational structure to support these capabilities is required for successful framework implementation. If organizational structure and culture are enabled to streamline supporting software development features the organization is more likely to succeed. These capabilities also help with administration of the services, maintain stability, quality, reliability and enhance transparency throughout the organization. Even though software test automation is only a part of the DevOps framework it can help organizations enhance their service quality as well as saving and re-allocating resources. This section utilizes the primary empirical conclusions (PEC) from this research as answers to the research question.

7.1 Answers to research question

This research concentrated on building an effective software test automation strategy in DevOps environment. Target organization operates on weather technology and industrial measurement businesses. However, these business areas were not specified in any part of this research to give more focused research model for organizations alike. This research defines the core capabilities of DevOps, how these capabilities can be implemented, and which requirements should organizations consider when implementing a software test automation strategy. The purpose of this research is to build a research model of implementing a successful software test automation strategy in a specific environment. To research this topic the research question was composed the following way:

- How to implement test automation strategy in DevOps environment?

Research question and its composition is described more accurately in chapters 2, 3, 4 and 5. This is done by reviewing existing literature and defining the core capabilities in DevOps framework and software test automation. The phenomenon of DevOps and software test automation strategy were researched via building a research synthesis using existing literature and conducting an empirical study. DevOps was described as a framework while defining the six core capabilities of an effective framework (see sub-section 2.1.3). The actual theoretical synthesis that this research is based on was derived from Fitzgerald and Stol (2014) and their BizDevOps-based model. The model was modified with the six core capabilities defined before, test automation strategy and its core features and combining business strategy with these dimensions. Interviews and primary empirical conclusions were based on this model (see figure 4). The framework of DevOps and the core capabilities were introduced through many different frameworks to get the most accurate and current description of the operating model. Even though DevOps as a framework was built to automatize all unnecessary manual tasks this research concentrates more on software test automation point of view.

Continuous deployment, delivery and integration are the outcomes where DevOps aims to. With continuous software monitoring organizations enable collecting feedback of the software performance. It could be argued that monitoring speeds up the development process as test automation can be executed after every release which gives proper feedback on work done. By enabling this organizations can provide software developers with constant input which also helps with enhancing communication between different organizational units. By enabling continuous monitoring faster response time is enabled which ultimately can lead to quicker fixes regarding the software. As software development and IT operations are traditionally siloed business units continuous monitoring enhances the communication within and helps with merging the units. Continuous monitoring was extracted from the interview data as PEC3 as enabled by continuous testing. While software monitoring has been done longer than DevOps itself has existed, creating automated reports and alerts that are available for the whole development team enhances efficiency within the project. This information has existed on theoretical and practical level before which can be said to verify the existing research. However, an organization like the target organization in this research needs to apply testing because all changes made in the developing organization might not always be visible. Monitoring can also be used as tracking the software evolution. While traditionally seen as a task done by IT operations continuous monitoring is essential for successful software development. By merging development teams and IT operations this data is shared within the project team which helps organizations also with deployment and release dimensions. The research also revealed that the data

gathered from continuous monitoring can be used later to enhance different aspects of the developed software.

Agile software development methods are defined by Highsmith and Cockburn (2001) as iterative phases which are followed by continuous delivery of the service. They are also a critical part of DevOps framework because of their continuous nature. Existing DevOps research is based on continuous deployment which makes traditional software development models (e.g. waterfall development methods) incompatible. Nevertheless, it can be argued that in small, very strictly defined projects agile software development methods might complicate the development process. A primary empirical conclusion (PEC1) extracts information that organizations can create hybrid development models which might include aspects of both trends. This can be classified as new information since current DevOps research argue that only agile software development methods can be utilized. If the project and its requirements are simple and static traditional software development methods can be used for the fastest and most efficient possible completion. By utilizing these models, other dimensions of DevOps such as continuous monitoring can still be combined with the development processes. Organizations need to find the best software development methods regarding complexity, size, and nature of the project.

By frequently releasing the software the risks regarding the actual software deployment can be minimized. Combining frequent releasing with continuous software monitoring the project team can gain knowledge in changes made in the software and its different setups and environments. This was also confirmed by this research as PEC5 confirmed the added value of seeing the changes made, reacting to alterations, and eliminating unnecessary waste. This information already existed and was backed up by this research. Frequent releases also empower the customer where the actual results of the development process can be visualized in short cycles. By frequently releasing the project team can track requirements with the customer while adding or removing desired preferences. Frequent releases are especially important when there are many developers working the same development phase. By enabling methods such as version control which support agile software development methods organizations can ensure the quality and efficiency of the development process. As version control can be fully automated it ensures that desired requirements of the service are met and constantly monitored. This also enables rolling releases back if necessary, which leads to minimized risks in final deployment.

As DevOps aims to combine two different departments the change also requires cultural movement within organizations. As combining these departments require constant learning from the employees as new tasks arise the initiative for change must come from the strategical management level. Managers also need to argue why change is needed and vindicate actions to do so. Even though transparency between different teams enhances efficiency the workload of specific employees might ascend. This requires careful resource planning from the management as well as recognizing available resources and their possible utilization. While the change has effect on organizational structure it also

concerns organizational culture. To adopt DevOps as a framework, organizations need to embrace specific supporting software engineering capabilities. These capabilities include dimensions such as open communication, incentive and responsibility alignment, respect, and trust. This information was confirmed in this research by two different primary empirical conclusions (PEC4 and PEC6). DevOps as a framework requires changes in organizational structure but also supporting capabilities for an effective implementation. Traditionally the two research branches of technical, more concrete aspects and abstract dimensions such as trust and responsibility alignment have been researched as different dimensions. In the light of these findings it can be said that all technical, structural, and abstract aspects of the framework need to be realized for a successful implementation. These dimensions are required for a successful change in organizational culture and structure. Learning new methods and technologies as well as feeling valued and higher level of autonomy is proven to have positive influence on engagement to the project. These social aspects can be optimized with proper team combinations as well as empowering the employees. This also boosts team morale within certain projects which ultimately can lead to increased team performance. (Callanan & Spillane, 2016.)

Even though DevOps aims to automatize every unnecessary manual task this research concentrated on test automation and its implementation strategy. As can implementation strategies of DevOps vary, software test automation strategy should remain intact throughout the process. By creating sufficient documentation, careful planning of the resources needed and utilized, creating automated test scripts, and utilizing test data to enhance processes organizations can create an efficient test automation strategy which can be modified with the processes. This was confirmed by a primary empirical conclusion (PEC2) as automated testing reducing lead times but requiring a carefully planned test automation strategy. By carefully planning processes to concern automated testing the benefits can be measured in increasing software quality, repeatability of testing, reducing unnecessary manual tasks and effort on manual testing and ultimately quicker responses to the changes needed. All these aspects help organizations achieve benefits regarding software testing which might not be achievable by manual testing. DevOps as a framework is based on automating all possible manual tasks from version control to software testing. For that reason alone, software test automation strategy in a DevOps environment is a critical factor of a successful software development and should always be included in DevOps planning processes. This data already existed and was backed up by this research.

While software test automation can bring many benefits to software development teams there are also disadvantages that should be considered when planning the strategy. DevOps framework can increase the workload of specific employees especially the ones working with software testing. Implementing a strategy requires resources since the processes desired for automation should be thoroughly inspected. The aspects limiting organizations from test automation strategy also include dimensions such as acquiring test automation tools

and expertise and the upkeep of these assets. Selecting tools which can be scaled to DevOps environment is critical for constant releasing defined by continuous deployment. This aspect requires a lot from the testers since they need to keep up with developers. By planning the test scenarios in the requirement phase organizations can prevent and minimalize these problematic scenarios. By investing resources such as skills to software testing and its automation organizations ensure that the workload of testers remains efficient throughout the DevOps development team. While planning test automation strategy, organizations need to ensure the release cycles so that software test automation upholds the position regarding software development. The problem of testing keeping up with development can especially be seen with technologies that require constant releasing. The creation of a software test automation strategy was compressed by PEC7 as a guideline which includes the requirement phase, include most critical processes, and the allocation of different resources. Even though most of this conclusion is backed up by existing research the implication of including test automation strategy in the requirement phase might be critical for companies such as the target organization. This way if the software is also developed elsewhere defects and changes needed can be found on time and quality of the software and the processes automated is increased.

Software test automation comes with its benefits and challenges which slightly differ regarding API and GUI testing. Still, a lot of these aspects can be reflected on each of these software testing branches. As it was demonstrated earlier in this research (see sub-section 6.2) by automating GUI testing time used on a simple account creation process could be decreased to 10 percent of the original time. Naturally, this decrease can be even more exponential when combined to bigger process builders. However, especially with GUI test automation impact on change management and business processes should not be overlooked. This argument was backed up by PEC8 as a claim that business strategy, change management processes and continuous planning should not be separated from DevOps framework. Even though continuous planning was already inducted by Fitzgerald and Stol (2014) as a part of the framework, including change management processes increase transparency throughout the business units involved. Creating test automation suites for processes can impact change management processes on multiple different ways. By improving quality and lead times organizations can enhance their development process and include unnecessary manual tasks in their test automation strategy. Nevertheless, processes can become dependent on test suites which means pauses on process development. By creating thorough documentation and requirement analysis organizations can minimize the risk of stoppage in change management and process development cycles. This can be extracted as new information regarding combination of DevOps and business strategy.

DevOps has very little empirical studies around it partly because of the vague definition, the magnitude of the entirety and short time the actual framework has existed. Still, for an effective implementation of test automation and DevOps strategy it can be argued that business strategy needs to be im-

plemented in the process planning phase. It gives organizations clear picture of the current state where processes go as well as provides viewpoints that software development models usually lack. By combining DevOps model with business strategy, a clear model of process planning can be executed with minimal risks. DevOps itself is an operating model for IT organizations which increases the performance of software development with effective internal communication. By including continuous monitoring from requirement phase through the whole product life cycle DevOps also includes tasks traditionally seen falling to IT operations. Even though DevOps itself is defined as a concept many organizations unwittingly utilize different models and phases of DevOps. To create an efficient DevOps and software test automation strategy, organizations need to recognize different approaches of agile software development methods, strategy, and dependencies (Bucena & Kirikova, 2017). It also widely used by operatives which need to release often (e.g. cloud and web services).

This research was done to better understand core capabilities of DevOps and implementing an efficient software test automation strategy in these kinds of environments. One point of the study was also to argue whether business strategy needs to be involved in DevOps process development and what affects does it have on the actual framework. By inspecting references and requirements it can be said that the research question can be approached and answered adequately. The results outline the core capabilities of DevOps with clear definition of business process involvement and software test automation definition. The research was done the way as originally planned. Future possibilities for DevOps and test automation studies could be researching the process of implementing the actual framework to an organization having a different operating model currently.

7.2 Limitations of the research

The limitations of this study be divided to three different areas. As this research was done as a Master's thesis the study is limited to a certain point. While contemplating on aspects like organizational culture and performance the actual outcome in this research is based on existing literature and studies. While researching these kinds of dimensions would need a stronger representation of the current state of the organization and how changes is these environments would ultimately affect the organizational state and performance. This study was conducted as a two-part research where the first part examines and creates the theoretical framework for the research and the second part illustrates the empirical study of the current state in the target organization. This way the most accurate research can be conducted and examined in the given limits of a Master's thesis.

In this research there is only one organization addressed and the outcome is formed by interviewing the employees of this organization. As this study was done as a commission for the target organization it was natural to examine the

prevalent state of the organization. This means that the results of this research might not be applicable to every organization straightforward. However, it should be remembered that DevOps has been presented as a unique framework where every organization adapting it as an operating model would need to carefully plan their implementation processes. By the high scalability of the framework the implementation process can be modified for specific needs or even obtained partially.

Limitations could also be measured in the theoretical framework created and if it would be applicable in practice. Qualitative research helps addressing specific limitations in scientific software engineering research by drawing studies together and making them larger and more applicable theories. This method was used in this research since the framework of DevOps has very little empirical studies around it. By conducting this research organizations can have more precise description of the core DevOps capabilities as well as guidelines and recommendations of building an effective software test automation strategy. This research is not a detailed instruction of benefits gained before and after implementation of a software test automation framework but more of a guideline for organizations in search for assistance in adapting the procedure.

7.3 Future research opportunities

DevOps as a framework is still a rather new concept and needs to have more empirical studies around it to ensure the proper definition and application of the model. By its definition DevOps is a software development framework which includes the aspects of agile software development methods while reforming organizational structure and culture. Some organizations might have already implemented parts of DevOps framework without realizing it which makes the research of the implementation process more difficult. While utilizing agile software development methods by releasing many times a day the future research should concentrate on how software testing can keep up with constant releasing of the service. For DevOps framework it is also crucial that employees in charge of specific stages of the service do their tasks in time for continuous deployment and delivery. This can be especially difficult for employees in organizations who have multiple projects ongoing.

From software test automation viewpoint future studies could concentrate on building bigger process builders and measuring the process of developing functioning test automation suites and comparing these to manual comparisons. More detailed information could also be researched after a test automation strategy has been defined and implemented in this type of environment to see what kind of affects it has on specific business processes. DevOps can be seen differently because of the multi-dimensional nature of the framework. While adapting models such as agile software development methods organizations need to realize that projects should be seen as individual and unique entireties which should be formed the best way applicable. DevOps is a logical choice in

concepts such as web and cloud-based services where the need of releasing is constant. However, the framework should also be examined in smaller environments where requirements can be very strict and specific. In the light of this research these are the reasons it is important that the framework of DevOps will be researched more and in different environments in the future.

REFERENCES

Amannejad, Y., Garousi, V., Irving, R., & Sahaf, Z. (2014). A search-based approach for cost-effective software test automation decision support and an industrial case study. In *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops* (pp. 302-311). IEEE.

Amaricai, S., & Constantinescu, R. (2014). Designing a Software Test Automation Framework. *Informatica Economica*, *18*(1).

Ammann, P., & Offutt, J. (2016). *Introduction to software testing*. Cambridge University Press.

Bass, L. (2017). The software architect and DevOps. *IEEE Software*, *35*(1), 8-10.

Bucena, I., & Kirikova, M. (2017). Simplifying the DevOps Adoption Process. In *BIR Workshops*.

Callanan, M., & Spillane, A. (2016). DevOps: making it easy to do the right thing. *IEEE Software*, *33*(3), 53-59.

Cois, C. A., Yankel, J., & Connell, A. (2014). Modern DevOps: Optimizing software development through effective system interactions. In *2014 IEEE International Professional Communication Conference (IPCC)* (pp. 1-7). IEEE.

Collins, E. F., & de Lucena, V. F. (2012). Software test automation practices in agile development environment: An industry experience report. In *2012 7th International Workshop on Automation of Software Test (AST)* (pp. 57-63). IEEE.

Collins, E., Dias-Neto, A., & de Lucena Jr, V. F. (2012). Strategies for agile software testing automation: An industrial experience. In *2012 IEEE 36th Annual Computer Software and Applications Conference Workshops* (pp. 440-445). IEEE.

Cruzes, D. S., & Dyba, T. (2011). Recommended steps for thematic synthesis in software engineering. In *2011 international symposium on empirical software engineering and measurement* (pp. 275-284). IEEE.

Diel, E., Marczak, S., & Cruzes, D. S. (2016). Communication challenges and strategies in distributed devops. In *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)* (pp. 24-28). IEEE.

Ebert, C., Gallardo, G., Hernantes, J., & Serrano, N. (2016). DevOps. *IEEE Software*, *33*(3), 94-100.

Fitzgerald, B., & Stol, K. J. (2014). Continuous software engineering and beyond: trends and challenges. In *Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering* (pp. 1-9). ACM.

Fitzgerald, B., & Stol, K. J. (2017). Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software*, *123*, 176-189.

Forbrig, P. (2018). BizDevOps and the Role of S-BPM. In *Proceedings of the 10th International Conference on Subject-Oriented Business Process Management*. ACM.

Freeman, H. (2002). Software testing. *IEEE instrumentation & measurement magazine*, *5*(3), 48-50.

Gandhi, G. M. D., & Pillai, A. S. (2014). Challenges in gui test automation. *International Journal of Computer Theory and Engineering*, *6*(2), 192.

Garousi, V., & Elberzhager, F. (2017). Test automation: not just for test execution. *IEEE Software*, *34*(2), 90-96.

Garousi, V., & Mäntylä, M. V. (2016). When and what to automate in software testing? A multi-vocal literature review. *Information and Software Technology*, *76*, 92-117.

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

Hirsjärvi, S., Remes, P., & Sajavaara, P. (2009). Tutki ja kirjoita. 15. painos. *Helsinki: Tammi*, 202-204.

Hüttermann, M. (2012). *DevOps for developers*. Apress.

Karhu, K., Repo, T., Taipale, O., & Smolander, K. (2009). Empirical observations on software testing automation. In *2009 International Conference on Software Testing Verification and Validation* (pp. 201-209). IEEE.

Kasurinen, J., Taipale, O., & Smolander, K. (2010). Software test automation in practice: empirical observations. *Advances in Software Engineering*, *2010*.

Kit, E. (1995). *Software testing in the real world: improving the process*. Addison-wesley.

Leung, H. K., & White, L. (1990). A study of integration testing and software regression at the integration level. In *Proceedings. Conference on Software Maintenance 1990* (pp. 290-301). IEEE.

Lwakatare, L. E., Karvonen, T., Sauvola, T., Kuvaja, P., Olsson, H. H., Bosch, J., & Oivo, M. (2016). Towards devops in the embedded systems domain: Why is it so hard?. In *2016 49th Hawaii International Conference on System Sciences (HICSS)* (pp. 5437-5446). IEEE.

Lwakatare, L. E., Kuvaja, P., & Oivo, M. (2015). Dimensions of devops. In *International conference on agile software development* (pp. 212-217). Springer, Cham.

Lwakatare, L. E., Kuvaja, P., & Oivo, M. (2016). Relationship of DevOps to agile, lean and continuous deployment. In *International Conference on Product-Focused Software Process Improvement* (pp. 399-415). Springer, Cham.

Mathiassen, L., Ngwenyama, O. K., & Aaen, I. (2005). Managing change in software process improvement. *IEEE software*, *22*(6), 84-91.

Mohamed, S. I. (2015). DevOps shifting software engineering strategy-value based perspective. *International Journal of Computer Engineering*, *17*(2), 51-57.

Qu, S. Q., & Dumay, J. (2011). The qualitative research interview. *Qualitative research in accounting & management*.

Rafi, D. M., Moses, K. R. K., Petersen, K., & Mäntylä, M. V. (2012). Benefits and limitations of automated software testing: Systematic literature review and practitioner survey. In *Proceedings of the 7th International Workshop on Automation of Software Test* (pp. 36-42). IEEE Press.

Ramler, R., & Wolfmaier, K. (2006). Economic perspectives in test automation: balancing automated and manual testing with opportunity cost. In *Proceedings of the 2006 international workshop on Automation of software test* (pp. 85-91).

Riungu-Kalliosaari, L., Mäkinen, S., Lwakatare, L. E., Tiihonen, J., & Männistö, T. (2016). DevOps adoption benefits and challenges in practice: a case study. In *International Conference on Product-Focused Software Process Improvement* (pp. 590-597). Springer, Cham.

Scott, J. E. (2007). Mobility, business process management, software sourcing, and maturity model trends: propositions for the IS organization of the future. *Information Systems Management*, *24*(2), 139-145.

Senapathi, M., Buchan, J., & Osman, H. (2018). DevOps capabilities, practices, and challenges: Insights from a case study. In *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018* (pp. 57-67). ACM.

Shahin, M., Babar, M. A., & Zhu, L. (2017). Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices. *IEEE Access*, *5*, 3909-3943.

Smeds, J., Nybom, K., & Porres, I. (2015, May). DevOps: a definition and perceived adoption impediments. In *International Conference on Agile Software Development* (pp. 166-177). Springer, Cham.

Soni, M. (2015). End to end automation on cloud with build pipeline: the case for DevOps in insurance industry, continuous integration, continuous testing, and continuous delivery. In *2015 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)* (pp. 85-89). IEEE.

Stillwell, M., & Coutinho, J. G. (2015). A DevOps approach to integration of software components in an EU research project. In *Proceedings of the 1st International Workshop on Quality-Aware DevOps* (pp. 1-6). ACM.

Stolberg, S. (2009). Enabling agile testing through continuous integration. In *2009 Agile Conference* (pp. 369-374). IEEE.

Suomalainen, T., Kuusela, R., & Tihinen, M. (2015). Continuous planning: an important aspect of agile and lean development. *International Journal of Agile Systems and Management*, *8*(2), 132-162.

Todnem By, R. (2005). Organisational change management: A critical review. *Journal of change management*, *5*(4), 369-380.

Vaisala.com. (3/2020). Retrieved from www.vaisala.com on 1.3.2020.

Virmani, M. (2015). Understanding DevOps & bridging the gap from continuous integration to continuous delivery. In *Fifth International Conference on the Innovative Computing Technology (INTECH 2015)* (pp. 78-82). IEEE.

Vos, T. E., Kruse, P. M., Condori-Fernández, N., Bauersfeld, S., & Wegener, J. (2015). Testar: Tool support for test automation at the user interface level. *International Journal of Information System Modeling and Design (IJISMD)*, *6*(3), 46-83.

Wang, F., & Du, W. (2012). A test automation framework based on WEB. In *2012 IEEE/ACIS 11th International Conference on Computer and Information Science* (pp. 683-687). IEEE.

Wettinger, J., Breitenbücher, U., & Leymann, F. (2014). Devopslang–bridging the gap between development and operations. In *European Conference on Service-Oriented and Cloud Computing* (pp. 108-122). Springer, Berlin, Heidelberg.

Wiklund, K., Eldh, S., Sundmark, D., & Lundqvist, K. (2012). Technical debt in test automation. In *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation* (pp. 887-892). IEEE.

Yoo, S., & Harman, M. (2012). Regression testing minimization, selection and prioritization: a survey. *Software Testing, Verification and Reliability*, *22*(2), 67-120.

Zhao, J. (2003). Data-flow-based unit testing of aspect-oriented programs. In *Proceedings 27th Annual International Computer Software and Applications Conference. COMPAC 2003* (pp. 188-197). IEEE.

Zhu, L., Bass, L., & Champlin-Scharff, G. (2016). DevOps and its practices. *IEEE Software*, *33*(3), 32-34.

ATTACHMENT 1 THEMES AND INTERVIEW QUESTIONS

1. Questions – Theme 1 – Current job description and challenges
    a. What kind of work your current role includes?
    b. How long have you been in the current role?
    c. What are the most critical challenges regarding your work?
        i. Three different challenges
        ii. Which one is the most important?
        iii. Opinion or based on fact?
2. Questions – Theme 2 – DevOps core capabilities
    a. Are the core capabilities visible in current work?
        i. How? Why?
    b. How could the target organization enhance these capabilities and when are they good enough?
    c. Is DevOps the only framework which has been considered?
3. Questions – Theme 3 – Test automation
    a. Visible in current work?
        i. If yes, how? If not, how could it affect?
    b. Impact on organizational processes?
        i. If yes, how? If not, how could it affect?
    c. Is test automation visible inside the organization?
        i. If yes, how? If not, how could it affect?

EXTRA QUESTIONS:

What are the most important DevOps capabilities?
- What affect do they have on your own work and organizational structures?

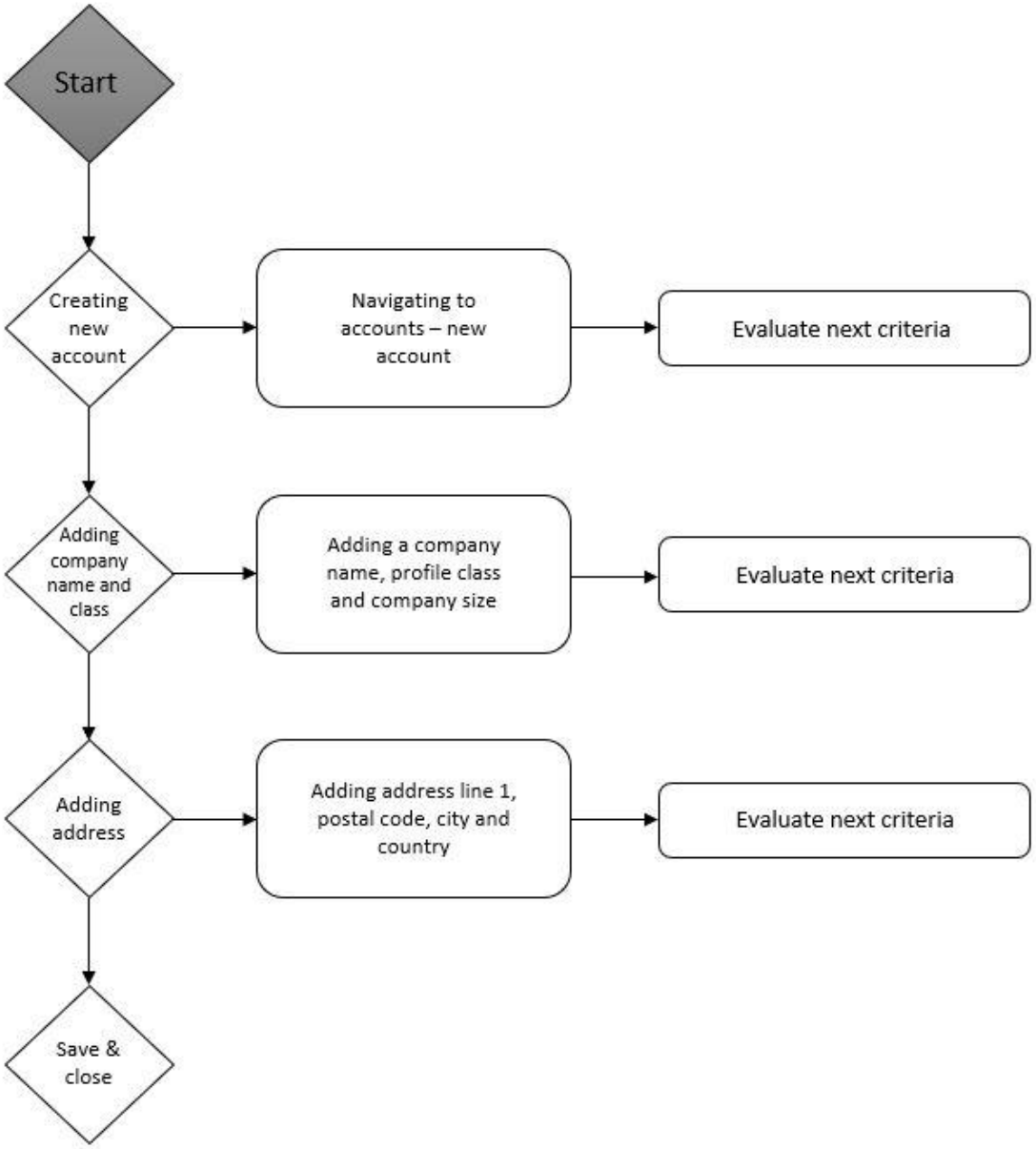
How would you describe the organizational operating model (vs. DevOps)?
- Similarities? Can something be done better?

How should the test automation framework be built?
- How to implement, which resources are needed, possible benefits, barriers?

How could test automation strategy affect the current organizational operating model?
- Is it visible? If yes, how?

Effect of test automation in organizational processes?
- Change management processes? Benefits?

ATTACHMENT 2 NEW ACCOUNT PROCESS



ATTACHMENT 3 SCRIPT RESULTS

Salesforce New Account creation

Test nr	Date	Time	Result	Comments
1	24.3.2020	10:00	Test crash	Crashing because of Chrome extension
2	24.3.2020	11:06	Success	OK
3	24.3.2020	12:10	Success	OK
4	24.3.2020	13:02	Success	OK
5	24.3.2020	14:15	Success	OK
6	25.3.2020	14:05	Success [8895643033720790XXXXX]	OK
7	25.3.2020	15:07	Success [1873869147549XXXXX]	OK
8	25.3.2020	16:18	Success [7555978684963912XXXXX]	OK
9	25.3.2020	16:55	Success [7418206386937412XXXXX]	OK
10	25.3.2020	17:49	Success [4727916369282652XXXXX]	OK
11	25.3.2020	19:29	Success [8406906069869475XXXXX]	OK
12	25.3.2020	22:37	Success [7381328499776292XXXXX]	OK
13	26.3.2020	9:46	Success [8644288582460052XXXXX]	OK
14	26.3.2020	10:49	Success [7100468007664822XXXXX]	OK
15	26.3.2020	11:43	Success [7289949653492XXXXX]	OK
16	26.3.2020	12:48	Test crash	Crashing because of Chrome extension
17	26.3.2020	12:52	Success [7664705006867XXXXX]	OK
18	26.3.2020	13:21	Success [5243722118931XXXXX]	OK
19	26.3.2020	14:34	Success [5758569440814902XXXXX]	OK
20	26.3.2020	15:29	Success [5764377732889XXXXX]	OK
21	27.3.2020	12:50	Success [3047617885806652XXXXX]	OK
22	27.3.2020	14:09	Success [2912330041367642XXXXX]	OK
23	27.3.2020	14:45	Success [5753745457052652XXXXX]	OK
24	27.3.2020	15:41	Success [6683804394748562XXXXX]	OK
25	28.3.2020	13:45	Success [7544166006868982XXXXX]	OK
26	29.3.2020	14:30	Success [7739407614116252XXXXX]	OK
27	29.3.2020	14:42	Success [4131877526691912XXXXX]	OK
28	29.3.2020	15:19	Success [4614023587125042XXXXX]	OK
29	29.3.2020	15:36	Success [8446409580144522XXXXX]	OK
30	29.3.2020	16:19	Success [6474628645237422XXXXX]	OK
31	29.3.2020	17:07	Success [3896721129002302XXXXX]	OK
32	29.3.2020	18:23	Success [4163905952220202XXXXX]	OK
33	29.3.2020	19:03	Success [3504089207901412XXXXX]	OK
34	29.3.2020	19:39	Success [6109386865652822XXXXX]	OK
35	29.3.2020	20:19	Success [7453950765179422XXXXX]	OK
36	29.3.2020	21:02	Success [2272665222388682XXXXX]	OK
37	29.3.2020	21:45	Success [2398859274107782XXXXX]	OK
38	29.3.2020	23:15	Success [6390435228386022XXXXX]	OK
39	30.3.2020	0:01	Success [6789754498367452XXXXX]	OK
40	30.3.2020	8:02	Success [5739992668466622XXXXX]	OK
41	30.3.2020	8:31	Test crash	Crashing because of Chrome extension
42	30.3.2020	8:36	Success [8170372738285522XXXXX]	OK
43	30.3.2020	9:08	Success [8862153093772022XXXXX]	OK
44	30.3.2020	9:40	Success [1088839103440432XXXXX]	OK
45	30.3.2020	11:06	Success [3769235530933622XXXXX]	OK
46	30.3.2020	12:09	Success [4597816375843002XXXXX]	OK
47	30.3.2020	12:40	Success [1650484074743022XXXXX]	OK
48	30.3.2020	13:43	Success [4467561502182222XXXXX]	OK
49	30.3.2020	14:21	Success [8489944438172022XXXXX]	OK
50	30.3.2020	15:04	Success [4564802446125222XXXXX]	OK