Miko Kiiski

UTILIZING AGILE METHODS IN CONTINUOUS SOFTWARE DEVELOPMENT

UNIVERSITY OF JYVÄSKYLÄ
FACULTY OF INFORMATION TECHNOLOGY
2022

TIIVISTELMÄ

Kiiski, Miko
Utilizing agile methods in continuous software development
Jyväskylä: Jyväskylän yliopisto, 2022, 59 s.
Tietojärjestelmätiede, pro gradu -tutkielma
Ohjaaja: Abrahamsson, Pekka

Ohjelmat ja järjestelmät ovat osa kaikkien päivittäistä elämää. Teknologiat muuttuvat kiihtyvällä tahdilla, uusia innovaatioita syntyy ja toimintatavat voivat muuttua yhdessä yössä maailman tilanteen mukaan. Ihmisten ja organisaatioiden tarpeet muuttuvat nopealla syklillä. Muutokset markkinoilla ja liiketoimintaympäristöissä vaikuttavat siihen, että organisaatioiden on reagoitava nopeasti ja pystyttävä vastaamaan muutoksiin. Organisaatioiden on pystyttävä ylläpitämään ja parantamaan omia sovellus- ja ohjelmistokehitysprosessejaan, mikäli aikovat pysyä kilpailijoiden edellä. Organisaatiot pyrkivät vastaamaan muutoksiin hyödyntämällä ketteriä menetelmiä ohjelmistokehityksessä. Ketterien ohjelmistokehitysmenetelmien hyödyntämiseen on tarjolla useita erilaisia viitekehyksiä ja käytänteitä. Haasteita tuottaakin oikeiden tapojen valinta ja niiden hyödyntäminen käytännössä ohjelmistokehityksessä. Uusimpien menetelmien joukossa on jatkuva ohjelmistokehittäminen, jonka tarkoitus on tehdä ohjelmistotuotannosta yksi yhtenäinen sykli, joka kulkee organisaation läpi.

Tutkielmassa toteutetaan laadullinen tutkimus, jonka on tarkoitus selvittää ketterien menetelmien hyödyntämistä jatkuvan ohjelmistokehityksen saavuttamiseksi organisaatiossa. Empiirinen aineisto kerättiin teemahaastatteluilla. Haastateltavat työskentelivät eri rooleissa ohjelmistokehitystiimeissä.

Tutkimuksen tulokset osoittavat, että ketterien menetelmien suurimpia haasteita on niiden konseptien ja termien määrittely sekä kouluttaminen. Havaittiin myös, että vaikka viitekehykset tunnetaan hyvin, niin niiden hyödyntäminen käytännössä on haastavaa.

Asiasanat: ketterät ohjelmistokehitysmenetelmät, ketteryys, jatkuva ohjelmistokehitys, haasteet

ABSTRACT

Kiiski, Miko
Utilizing agile methods in continuous software development
Jyväskylä: University of Jyväskylä, 2022, 59 pp.
Information Systems, Master's Thesis
Supervisor: Abrahamsson, Pekka

Software and systems are part of everyday life. Technologies change at an accelerating pace, new innovations are born and operating methods can change overnight depending on the world situation. The needs of people and organizations change in a fast cycle. Changes in the market and business environment mean that organizations must react quickly and be able to respond to changes. Organizations must be able to maintain and improve their own application and software development processes if they intend to stay ahead of the competition. Organizations strive to respond to changes by utilizing agile methods in software development. Several different frameworks and practices are available for utilizing agile software development methods. Choosing the right methods and using them in practice in software development creates challenges. Among the latest methods is continuous software development, the purpose of which is to turn software production into one coherent cycle that runs through the organization.

In the thesis, a qualitative study is carried out, which is intended to investigate the utilization of agile methods to achieve continuous software development in the organization. Empirical data was collected through thematic interviews. The interviewees worked in different roles in software development teams.

The results of the study show that the biggest challenges of agile methods are defining their concepts and terms and training them. It was also observed that although the reference frameworks are well known, their utilization in practice is challenging.

Keywords: agile software development, agile, continuous software development, challenges

FIGURES

TABLES

TABLE ON CONTENTS

1 INTRODUCTION

The agile methods in software development have been a topic of conversation for a long time. Various methods, frameworks and practices have been created to support software development. New methods are also constantly emerging, as the ways of using software and the needs of customers and society change at an accelerating pace. It is difficult for organizations to choose the right methods and tools to support their own operations. We live at time where software is an increasingly important part of everyday life. However, software are getting bigger and more complicated to develop, this brings challenges to everyone working around software development (Jacobson et al., 2022).

1.1 Motivation

Agile and Lean software development methods have grown in popularity after release of the Agile manifesto in 2001. Organizations have taken steps to improve their processes in software and product development. The benefits of agile methods are beginning to be clear to all, however, their implementation and utilization is not easy. Successful use of agile methods allows an organization to improve productivity, accelerate the pace of development, and prepare the organization to face changes in a rapidly changing business environment. (Conboy, Coyle, Wang & Pikkarainen, 2011.)

Based on agile and lean, many practices and frameworks have developed, from which the organization and development teams have to choose the ones that suit them best. Scrum framework is one of the most used agile methodology. Elements of Scrum are to do software development in sprints. Activities that the Scrum team implements are always repeated within the sprint, such as daily scrum, sprint review and sprint retrospective. (Agh & Ramsin, 2021.) Scrum as such is intended for a small software development team. For scaling agile methods to a larger group, different frameworks are also available.

Scaled Agile Framework is one of the most used framework for scaling agile, it tries to bring all best practices together. (Ebert & Paasivaara, 2017.) Scaled Agile Framework collects the best practices, tools and principles for organizations to use. The basic idea of the framework is built based on lean software development, agile product development and system thinking. (Kalenda, Hyna & Rossi, 2018).

Although there are many options available, organizations and the development team do not find it easy to choose the right methods. Even if there was a certain popular method like Scrum in use, it is not easy to use it in practice. Essence has been developed to facilitate the selection of different methods and to enhance the selection of the necessary elements. (Jacobson et al., 2022.) If the organization succeeds in software development and chooses the best methods, then it is on its way to continuous software engineering. Continuous software development and engineering includes different aspects like continuous integration, delivery, testing and deployment (Fitzgerald & Stol, 2017).

1.2 Research questions

The goal of this study is to investigate utilizing agile development methods in continuous software development and try to find best practices to ensure continuous software development. The goal is to investigate what challenges are encountered when trying to use agile methodology in software development and what are the ways to overcome those challenges. This forms the first research question:

- How to adapt agile methods in continuous software development?

This research question is answered with the empirical part of the research. In addition to this, the research aims to understand what different methods, frameworks and practices are available to support agile software development. This forms the second research question:

- Which agile development methods can be used in continuous software development?

This research question is answered in the theoretical part of the research, by examining existing previous studies and scientific publications.

In addition to this, the aim is to understand what challenges there are in utilizing agile methods. This forms the third research question:

- What are the challenges in applying Agile methods?

This research question is answered in the theoretical part of the research, by examining existing previous studies and scientific publications.

1.3 Structure of thesis

This chapter reviews the structure of the thesis. Chapters two and three review the theoretical background of the research, this part of the research reviews previous studies and scientific publications that are relevant to answering the research questions. The research mainly tried to use peer-reviewed studies.

First we go through the background and history of Agile and lean software development. Then some of the most used frameworks are introduced. Next, scaled agile methods and their best-known frameworks will be reviewed. In addition, the challenges encountered in the utilization of the methods are described. After that, through ways to improve the methods used. Finally, we go through the elements of continuous software development and ways to succeed with them. The fourth chapter presents the research goals, research methods, data analysis and data collection and justifies them. In addition, the reliability and validity of the research is evaluated. The fifth chapter presents the empirical findings of the study and the conclusions drawn from them. The primary empirical conclusions are separated from the conclusions.

The sixth chapter discusses of the primary empirical conclusions from theoretical and practical view. The chapter seven reviews the conclusions and answers the research questions.

2 AGILE SOFTWARE DEVELOPMENT

Agile practices and methods have grown in popularity with companies that want to produce working high quality programs and products, but at the same time improve the company's speed and agility to respond to changes. A lot of different methods, practices, frameworks, tools, and models are built around agile development. This chapter describes how Agile and Lean appear in the software development. The chapter also presents the most popular frameworks for agile and lean development and their benefits. The challenges associated with the organizations transfer towards agile models are also discussed. This chapter also presents DevOps and BizDev.

2.1 Agile and lean

Agile and lean software development methodologies are both made to increase flexibility and speed up the development process to ensure that delivered products are high quality and working. The objectives and principles are very similar in both methodologies, so comparing them can be confusing. Agile and lean methods do not exclude each other, they can be part of the same production chain but operate in different stages.

Agile software development methods emerged after the publication of the Agile Manifesto in 2001 (Beck et al., 2001). The software development industry was already in great need of more customer-oriented and efficient methods and previous methods had long been criticized. Agile methods offer more flexible and efficient software development. Agile methods also emphasize customer-centricity.

The Agile Manifesto defines four values for agile development:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negation

- Responding to change over following the plan.

These values are the background for many agile methodologies. (Beck et al., 2001.)
The Agile Manifesto also presents twelve supporting principles that guide the nature of all agile methodologies:

1. Customer satisfaction through early and continuous software delivery
2. Changing requirements are welcome
3. Frequent delivery of working software
4. Collaboration of business and development teams
5. Motivated and trusted team works better and more efficiently
6. Information should be shared face-to-face
7. Working software is the main measure of progress
8. Agile processes support sustainable development
9. Attention to technical excellence and design enhances agility
10. Simplicity, developing just enough for this stage
11. Self-organizing teams emerge best architectures and designs
12. Regular reflections on how to be more effective and process improvements

All the values and principles aim to one result, that is delivering a working software for the customer.

The traditional software development includes extensive planning, heavy documentation, and well-defined processes and schedules. Agile software development focuses on the product and making the solution. The focus is on teamwork together with the customer. Changes and customer wishes can be responded quickly because time is not wasted on documenting and lengthy processes. The information is shared within the team and the relationship with the customer is strengthened. (Coleman, 2016.) Common to all agile methods is the use of iterative development and repeated deliveries of the product to the customer. Agile methods also aim to keep customer satisfaction high with early deliveries of the final product, instead of just delivering finished product (Misra et al., 2011). Other differences to traditional non-agile methods are self-directed teams which must remain small enough, continuous improvement of design and continuous integration. Agile methodology believes in motivated individuals that only need the right environment and support to succeed (Misra et al., 2011). Agile methods also prefer direct communication between team members and all the information should be shared conveniently face-to face, avoiding unnecessary heavy documentation. (Greer & Hamon, 2011.)

The customer collaboration enhances that the customer is sure to get the product they wanted. If the specification document is followed without cooperation and interaction with the customer, it can easily happen that the end product is something completely different from what the customer wanted. The customer's feedback during the project is one of the most important aspects in agile

development. Feedback reduces unnecessary work and makes it possible for customers to receive the desired end result. Often software project requirements also change during the project, the ability to quickly respond to changes is one of the benefits of agile software development. (Coleman, 2016.)

Agile methods are changing and evolving all the time. Organizations can follow the latest best practices, when striving to improve their own performance. Agile methods offer challenges and opportunities. There are also situations and projects where agile methods are not the best ones to use. The use of agile methods is most effective in a changing business environment where organizations need to react quickly to changes and stay ahead of competitors.

The roots of Lean thinking are in Toyota's automotive industry (Poppendieck & Poppendieck, 2003). The Toyota Production System was the basis for the lean thinking tool Kanban, that is used when an organization wants to develop its operations towards Lean. Kanban is explained in more detail in chapter 2.1.2.

Lean software development methodology was built to increase productivity and eliminate waste (Poppendieck & Poppendieck, 2003). The principles and practices of agile development described earlier fit into Lean Software Development. Lean Software Development also emphasizes the importance of customer focus and continuous improvement of the development process (Poppendieck & Poppendieck, 2003). However, there may be too heavy elements in Agile development when compared to Lean. Focus on Lean Development is to produce workflow that allows flexible and efficient development phases. (Poppendieck & Poppendieck, 2003.) According to Poppendieck and Poppendieck (2007) there are seven principles that can be applied to Lean Software Development. Poppendieck and Cusumano (2012) modified these principles that are used to frame the ongoing discussion about software and product development and how principles fit in unique situations. Lean Principles are shown in Table 1.

Table 1 Lean Principles (Poppendieck & Cusumano, 2012)

Lean Principle	Summary
Eliminate Waste	Waste is anything that is not adding value to a product or is not useful for delivering working and high quality products to customers. Waste in software development can be unnecessary requirements or codes that are not useful right now and are slowing down the whole project.
Build Quality In	Quality cannot be overemphasized in software development. However, ensuring this is not always clear. Popular Lean development tools to ensure quality: Pair programming, enough testing, incremental development, workflow and automation. Every individual should also be responsible for quality.
Optimize the Whole	Optimizing means that a value stream to customers is the only thing that matters. Every action and business should be optimized to produce as much value as possible.

	Identifying the flow of value between teams is the first step to optimizing.
Learn Constantly	Development aims to create knowledge and make a product of it. Lean development approaches learning in two different ways. The first is to delay critical decisions to the last moment so that decisions can be made with the best knowledge and minimize change related costs. The second approach is the continuous learning, where frequent delivery and customer feedback are used to minimize unnecessary features and costs.
Deliver Fast	Faster delivery highlights the idea that software development should not be seen only as projects, but instead as a flow that goes through design, development and release of product. To ensure speed it is fundamental to deliver simple solutions that can be easily changed according to feedback.
Engage Everyone	The different departments and units should not be kept too separate. It should be possible for everyone who is responsible for the value development of the product to interact. Encouraging collaboration and driving the idea of self-organizing teams are elements of Lean thinking.
Keep Getting Better	Look for opportunities to be better. Every working method should be researched using scientific methods and if possible make improvements.

2.1.1 Scrum

Agile software development consists of different software development methods and frameworks. One of the most popular and widely practiced is Scrum. At least 75% of respondents of the 14th annual State of Agile report (2020) announce that they are practicing Scrum or a hybrid that includes Scrum (State of Agile, 2020). Scrum was first introduced in 1986 by Takeuchi and Nonaka in the context of product development. Takeuchi and Nonaka (1986) argue that this holistic approach consists of features: self-organizing teams, overlapping development phases, light control and transfer of learning. In the context of software development it was introduced and modified by Schwaber and Sutherland in 1995. The literature defines Scrum as a framework or methodology. Scrum offers tools for people to deal with complex problems, while delivering valuable products effectively (Schwaber & Sutherland, 2017). It is mainly used in product and software development. However, tools and practices that Scrum offers also work with other industries. (Gonçalves, 2018.) According to the 14th annual State of Agile report (2020), the five industries that used scrum the most: technology (27%), financial services (17%), professional services (7%), government (7%) and insurance (6%). Scrum works best in environments where organizations and teams have to respond quickly to changing situations. Organizations using scrum are

able to deliver products of high quality at a fast pace (Gonçalves, 2018). Scrum helps teams and organizations to make adaptive solutions to complex problems.

At the center of Scrum is The Scrum Team. Scrum Teams must contain individuals who are able to self-organize and have the necessary skills for development. These practices enable the team to produce quality and functional products. The goal is also to deliver a partially finished product to the customer for feedback. When the team has the necessary skills, it is easy to respond to feedback and improve or correct the product based on the feedback. Agility, flexibility and productivity are optimized by this model. Scrum team has different roles: Product Owner, Scrum Master and Development Team (Gonçalves, 2018). The Product Owner is responsible for the value of the product. Product Owner must take care that the customer's requirements for the product are followed and the customer gets the product they ordered. The Product owner is only one person and the other team members must listen and implement the features that the Product Owner prioritizes. Scrum Masters role is to offer guidance and training for the Scrum Team that everyone understands and follows the values, principles and rules of the framework. The Scrum Team works under the requirements that the Product Owner has set. Team consists of a small group of experts that are able to work without leaders. Team reports on progress to Product Owner and Scrum Master. The size of the Scrum Team recommended to be at its maximum of nine, any larger team will produce too much coordination. (Schwaber & Sutherland, 2017.)

The Scrum is a framework that contains development iterations that are called “Sprints” and every Sprint includes “Events”(Figure 1). Sprint is a short period, normally a month or less, any longer time period would threaten the project. Shorter time periods increase predictability through inspections and meetings. Sprints should be seen as small projects and a new sprint begins immediately after the end of the sprint. Each Sprint the team should deliver an increment of software, that ensures the achievement of the goal of delivering working software. Schwaber and Sutherland (2017) argue that during the Sprint: changes should not be done, especially if they risk the project, quality does not decrease and scope can be redesigned with the Product Owner as more is learned. Product Owners can also cancel Sprints if goals have become obsolete due to changes and start planning new Sprint. (Schwaber & Sutherland, 2017.)

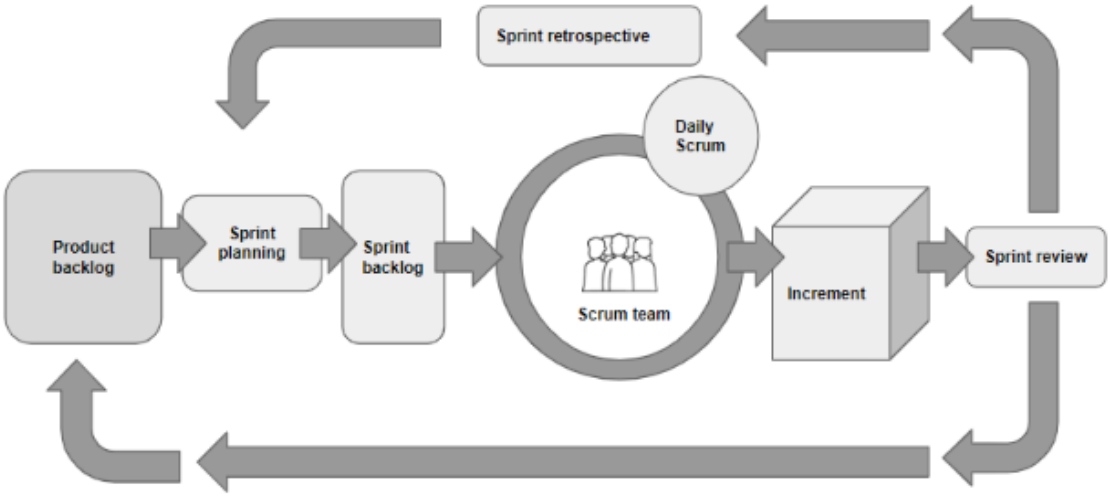

Figure 1 Scrum Framework according to Schwaber & Sutherland (2017)

As mentioned, each sprint includes four events that are precisely set to last a certain amount of time. Sprint starts with the Sprint Planning, the entire Scrum Team participates and for a one month Sprint the maximum duration should be only eight hours. Scrum master is responsible for planning the event and ensuring the timeframe. (Schwaber & Sutherland, 2017.) The goal of this event is to plan, what is the expected outcome for the Sprint and what parts of the product can be delivered? It is also necessary to plan how the goal will be achieved and whether the team has all the necessary resources. Another event that belongs to the Scrum is Daily Scrum. Every day during the Sprint includes a short meeting that takes 15 minutes. The Development Team is responsible for organizing the meeting and Scrum Master only gives guidance for the meetings. The goal of it is to plan tasks for the next 24 hours. Meetings should always be at the same time to reduce misunderstandings and clarify the process. At the end of the Sprint is an event called Sprint Review, this event will be attended by the entire Scrum Team. The purpose of the meeting is to discuss and inspect that is the goal of the Sprint achieved and what should be done next. It is also highlighted if problems or other issues were encountered and the event also looks at a possible completed part of the product. Timeframe for the Sprint Review should be four hours if the Sprint lasted a month. (Schwaber & Sutherland, 2017.) The last event before starting a new Sprint is Sprint Retrospective. The entire Scrum Team will also be participating in this event. The idea of this event is to offer an opportunity to all participants to inspect what went well and what could be improved. The Scrum Master is also responsible for events taking place and that everyone understands the purpose behind the event. Based on the development suggestions found, it is intended to ensure that the same situations are not encountered in the next Sprint. The Scrum Team is responsible for implementing the agreed improvements. (Schwaber & Sutherland, 2017.)

Scrum Artifacts are tools to manage the project and keep all the information transparent. Most important Artifacts are Product Backlog and Sprint Backlog. The Product Backlog contains everything from the features, requirements and fixes needed to finish and the product. It is also dynamic and evolves as requirements, markets, or technology causes changes, and the Product Owner is responsible for updating the Product Backlog. Features in Product Backlog are named user stories that describe the usefulness of features to the customer. Product Backlog can be described as a living artifact due to its nature. The Sprint Backlog contains a list of items that are selected from The Product Backlog by The Product Owner. These items are planned to be completed during the Sprint. Sprint Backlog presents all the work that is needed to be done to achieve the goal of the Sprint. Only The Development Team can make changes to the Sprint Backlog within a Sprint. (Schwaber & Sutherland, 2017.)

2.1.2 Kanban

The previous chapters introduced how Lean has become more popular and has influenced software development. This chapter introduces the tool and method Kanban at the heart of lean. Kanban was developed by Taiichi Onho in the 1940s, at the time he was working for Toyota. Kanban is a tool to reduce costs, increase productivity and improve management. Kanban in software development was introduced by David Anderson in 2004, at the time Anderson was working for Microsoft. Kanban can be described as a change management tool and it is not connected only to software development.

According to Anderson (2010) Kanban uses five core elements, which are needed to successfully implement Kanban in organization. These elements guide organization to Lean-thinking:

- visualize the workflow
- limit the work in progress
- measure and manage the flow
- make policies explicit
- use models to identify development opportunities

Kanban is based on an idea that work should be limited and new phases or projects should not be started before finishing the ongoing ones. The Kanban board helps to visualize the situation. A visual signal is produced to inform that new phases can be started. Kanban board can be digital or for example traditional whiteboard with sticky notes. The common practice is to produce both versions. Kanban provides transparency to all stakeholders and helps the development team to understand what is the next phase or work task. There are different variations of the Kanban board. Therefore, organizations should experiment, customize, recreate and optimize it to better fit in different situations that best suits

organization's practices and work environment. The team is responsible for updating the Kanban board and they decide what new tasks to fill in. All Kanban boards should have a section that includes work to do. These tasks can be transferred through different sections in the Kanban board. Other sections should show what tasks are under development and what is already done. The Kanban board can also show challenges and problems that projects face. Challenges and problems should be marked with different colors that will help them to pop out so that they are quickly responded to. Figure 2 shows variation of Kanban board.

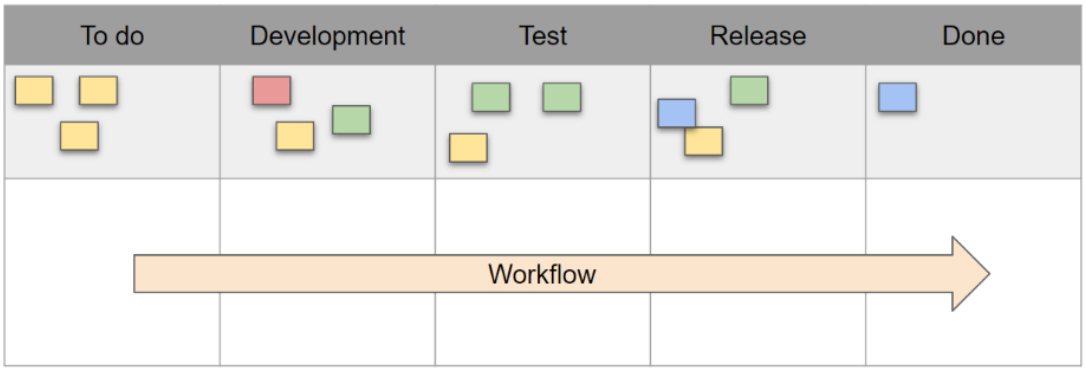

Figure 2 Kanban board according to Kniberg & Skarin (2010)

Studies have shown that Kanban changes the work environment to be more open, when everyone can see the stages of projects. Collaboration increases and the team is able to spot slowdowns and problem areas faster. Kanban evolves organization work culture to more agile ja lean. (Anderson, 2010.)

Anderson (2010) argues that Kanban should not be used as a project management tool, instead Kanban should be adapted to the already ongoing process of the organization which can be improved with Kanban.

2.2 Scaled agile

The agile methods presented in the previous chapters do not work as such on a large scale, so scaled methods are needed for organizations to use agile and lean practices efficiently.

Scaled Agile Framework (SAFe) is one of the most popular frameworks for scaling Agile (Remta & Buchalcevova, 2021). According to the 14th Annual State of Agile report (2020), 35% of respondents are practicing Scaled Agile Framework to scale agile. Other popular frameworks for scaling agile are Scrum of Srums (SoS), Disciplined Agile (DA) and Large-Scale Scrum (LeSS) (Kalenda, Hyna & Rossi, 2018). This chapter introduces SAFe and LeSS in more detail.

2.2.1 SAFe

The Scaled Agile Framework was first released in 2011 and since then five versions have been released. The Scaled Agile Framework is a collection of agile and lean practices (Kalenda, Hyna & Rossi, 2018). The scaling agile frameworks are made for large organizations to manage agile and lean practices. Scaled Agile Framework helps organizations to reduce the time it takes to go to new markets and improves productivity and quality of services and products. (Remta & Buchalcevova, 2021.) SAFe has been built on the principles of agile and integrating them with lean best practices, to create scalability through products, teams and solutions in the organization. (Sreenivasan & Kothandaraman, 2019). Scaling in the context of agile means the continuous process that happens in the organization when knowledge is transferred, translated and transformed across different teams and persons. This also puts a lot of pressure on good communication between communities and units. (Kalenda, Hyna & Rossi, 2018).

SAFe is built around seven competencies which are: enterprise solution delivery, lean portfolio management, organizational agility, continuous learning culture, lean-agile leadership, team and technical agility and agile product delivery. (Scaled Agile Framework, 2022.) The core elements of the competencies are presented in the table 2.

Table 2 SAFe Competencies (Scaled Agile Framework, 2022)

Lean Portfolio Management	Describes how to coordinate with portfolio's value streams that are specific for business domains. Lean Portfolio Management is a modernized version to support the Lean-Agile way of working.
Organizational Agility	One competitive advantage is the speed that organization can respond to needs of its customers. Organizational agility must be built to ensure this capability.
Continuous Learning Culture	Continuous learning should be organization-oriented. Continuous improvement is also everyone's responsibility, because otherwise it is difficult to keep up with the pace of the market.
Lean-Agile Leadership	Describes how leaders can ensure that teams and individuals reach their full potential. Leaders must have a Lean-Agile mindset to succeed and lead by example. Adoption, success and improvement of Lean-Agile development in organization is the responsibility of leaders.
Team and Technical Agility	The focus is on agile teams that are cross-functional and high-performing. Teams should be able to use the best agile methods

	to produce solutions. Agile teams together with business teams strengthen the organization's business agility.
Agile Product Delivery	At the center of the product strategy is the customer-centric approach to decision making. All the decisions should be based on the customer, and decision-making is continuous throughout the entire development process. This enables continuous delivery ready for market demands.
Enterprise Solution Delivery	Describes how to adapt Lean-Agile methods to the largest and complicated softwares and systems. Coordinating and aligning the full supply chain.

SAFe evolves with new technology trends and offers the latest information for the organizations. SAFe version 5.0 was released in January 2020. SAFe 5.0 highlights the business agility even more than earlier versions of the SAFe. Business agility means organizations skills to respond to market changes and compete with innovative solutions. To deliver solutions organizations must participate in all the parties including development, IT operations, marketing, finance, support and others. The lean and agile practices are not only for the development teams. SAFe 5.0 highlights even more than previous versions the customer centricity and design thinking. Emphasizing customer centricity can be a big change in mindset that needs to be implemented throughout the organization. Organizations must understand the needs of the customer and make products and services to ensure that the customer receives value. SAFe offers four different configuration models for different sizes of organization and projects. Configurations fit in specific environments (Remta & Buchalcevova, 2021). Depending on configuration different levels and tools are added to SAFe. The Essential SAFe configuration is the basic version of SAFe. Essential SAFe contains the main practices of Agile Product Delivery and Technical Agility. It also includes key concepts like the Agile Release Train (ART). The Large Solution SAFe configuration offers the Enterprise Solution delivery ability that is needed for large projects that will need more than one ART and dozens of teams to deliver. Large Solution SAFe is mainly used in aerospace, defense, banking system, automobile and medical device projects with the government industries. One of the key concepts is the Solution Train that coordinates multiple ARTs and makes it possible to produce complex solutions that in failure could lead to economic or social effects. The Portfolio SAFe configuration is a collection of practices that makes possible the organization to reach Business Agility. It provides abilities that are Organizational Agility and Portfolio Management. Measure and Grow is the one of the key concepts to way portfolios evaluate their progress to the full business agility and decide the steps how to get there. The Portfolio SAFe also offers methods and strategies for lean budgeting. Full SAFe configuration includes all the abilities that are needed for business agility. Full

SAFe is mainly used by large organizations that need to manage large solutions. (Remta & Buchalcevova, 2021.)

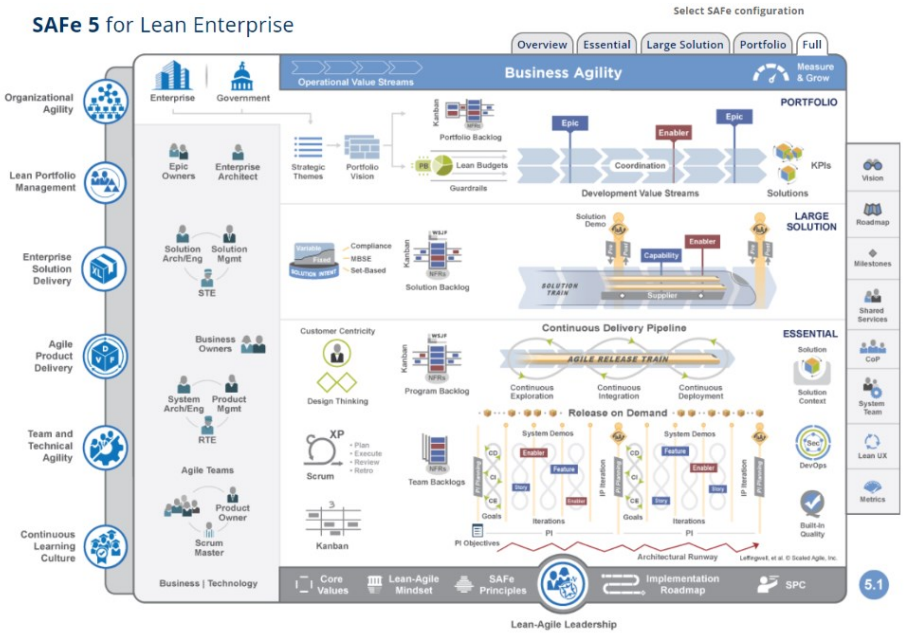

Figure 3 Full SAFe configuration. (Scaled Agile Framework, 2022)

2.2.2 LeSS

Large-scale Scrum (LeSS) is a version of Scrum that has been developed for large groups. With the help of Less, the strengths of Scrums agile development can also be used for larger teams. Less also tries to stay simple and includes the same elements as Scrum, such as sprints. The idea is that less is Scrum scaled. (The LeSS Company B.V., 2022.)

LeSS tries to bring out the deepest ideas of the Agile manifesto, which are that individuals and encounters are more important than tools and processes (Kalenda, Hyna & Rossi, 2018). There are two different versions of LeSS. LeSS, which is intended for management between a maximum of eight teams (eight people each team) and LeSS Huge, which can be used to manage several thousand teams and employees that work with the same product. There are many elements in common between LeSS and Scrum. Both have one product backlog because the teams work on one product, because of this LeSS removes the need for portfolio management (Almeida & Espinheira, 2022). Both include Sprints, but the difference in LeSS regarding sprints is that there is only one product-level sprint and no individual sprints for each team. Like Scrum, each team holds a daily scrum meeting, where the team organizes its tasks for the day. Other team members can participate in other teams dailys to ensure information sharing. (The LeSS Company B.V., 2022.)

In addition to the product owner, members of other teams also participate in the sprint planning meeting. Together, the teams decide the priorities of the product backlog and plan cooperation. After this first sprint planning meeting, team-specific planning meetings are held, which are almost similar to Scrum. A Product Backlog Refinement meeting is held halfway through the sprint. The purpose is to ensure that big things are shared, the situation of things is clarified, the work situation, risks and other essential factors are assessed. The meeting is carried out between several teams, it would be good to have people from all teams involved. Customers, experts and users can also be involved. At the end of the sprint, there is a sprint review, which again includes people from all the different development teams. The customer, user or other stakeholders are also usually involved. They review the part produced during the sprint and decide the next development directions for the project. (The LeSS Company B.V., 2022.)

Finally after the sprint, each team holds its own retrospective, in which it is estimated what challenges were encountered during the sprint and how they will be overcome in the future. Less presents yet another meeting, which is an overall retrospective. Its purpose is to discuss challenges at the organizational level, assess how teams work together and share the necessary information at the organizational level. (The LeSS Company B.V., 2022.)

2.3 Challenges

Scaling Agile in large organizations can be challenging. Recent studies have shown that coordination and communication are in an important position, to scale to be successful. The process needs the right people to be part of it. When implementing new scaling practices many different challenges can be met. The most common challenges are resistance to change and poor leadership skills. Adopting new practices means a major organizational change and the flow of information must be ensured for all parties. The training of personnel and involvement during change is key. (Kalenda, Hyna & Rossi, 2018). Resistance to change is one of the major challenges when transferring organizations methods to more scaled agile. The resistance can be also found in any levels of the organization, from development teams to the management level. Resistance in the higher levels is also even more complex, and it needs a professional change management to support it. The missing change management in described situations can lead to major problems. (Kalenda, Hyna & Rossi, 2018.)

There can be several reasons behind resistance. Agile methods often increase transparency through tools that are used, for example the Kanban board. While it is much easier for everyone to monitor the progress of work, some workers may experience it as negative because they feel that they are being watched more than before. Conboy et al,. (2011) also recognize the fear of transparency among developers that it will reveal level of competence. Agile methods for sure make it easier for the team to find out if someone faces problems, but team help is also available faster. Another reason for resistance can be that, Agile methods also

prefer self-organizing teams, some employees may experience it as challenging because they feel that responsibility is accumulating too much and problems need to be solved independently. (Inayat et al., 2015.) This challenge highlights the point mentioned earlier about the importance of finding the right people to use agile methods.

Agile development methods prefer face to face communication and inexhaustible collaboration between team members. It would be good for the team to work close to each other and allow for encounters. An overly decentralized team can cause problems and slow development. Sharing knowledge and learning from each other can be challenging if work is only done through online meetings. (Inayat et al., 2015.)

New ways of working can cause workload to increase and that can lead to pressure in the teams. The change of working habits should be scheduled at the right time. In rapidly changing environments that can be challenging. Problems can also show up when the whole transfer to Agile methods is not done properly. Developers do not have motivation to use new methods, because they feel learning new ways is too overwhelming. (Conboy et al., 2011.)

It is easy for people to misunderstand the responsibilities from new methods and critical testing or work phases can be missed. Agile transformation needs enough training, planning and coaching. If the transformation is done too quickly without support management , the lack of knowledge can be spread through teams. Teams that do not get enough information are not going to change their old habits and the whole transformation failed. Change is also difficult to implement if teams do not value Agile principles and find them futile. (Inayat et al., 2015.)

2.4 Essence

There are many different frameworks, methods and practices for software development. Previous chapters introduced some of the most used ones such as Scrum and SAFe. At best, different frameworks and practices improve the agility, quality and success of software development in the organization. Even though there is plenty of choice, organizations and development teams still find it difficult to choose the right practices. It is particularly challenging to utilize them in software development. One of the biggest challenges even today is developing really high-quality softwares and maintaining it. However, the problem is not that there are many practices, but that choosing the right practices for your own development team is difficult and there are few tools for making choices. These challenges were identified in 2009 when the research community decided it was time to expand people's understanding of software development. As a result of many years of hard work by the SEMAT (Software Engineering Method and Theory) community, a new framework called Essence was born in 2014, which allows us to look at development methods from new perspectives. (Jacobson et al., 2019)

2.4.1 The Kernel

From Essences point of view, all practices used contain some reusable parts. These can be, for example, a backlog, user stories, continuous development or a motivated self-directed team. With Essence, all of these parts should be able to be used effortlessly. The challenge was that many of the methods used are tied to the framework from which they originally came. They use a vocabulary that makes it difficult to combine different practices. The first goal of Essence was to standardize the vocabulary used in terms of how different practices are utilized. Based on these, the common ground of software development methods was created in Essence. This common ground is called the Kernel. (Jacobson et al., 2022.)

Utilizing the Kernel makes learning, using and evaluating the best methods much easier. The software development team is able to constantly monitor its own performance and change it based on practices. The Kernel consists of three activity spaces. These activity spaces are: customer, solution and endeavor. The customer activity space includes things related to exploring opportunities, understanding the needs of stakeholders, meeting the needs of stakeholders and using the system to observe benefits. The solution activity spaces include things related to achieving requirements and development. It includes understanding the requirements, modifying the system to support easy use and maintenance, system implementation, testing, deployment and future maintenance. Endeavor activity space contains things related to development teams and ways of working. There is team building, work planning, support and monitoring. (Ivar Jacobson International, 2022.)

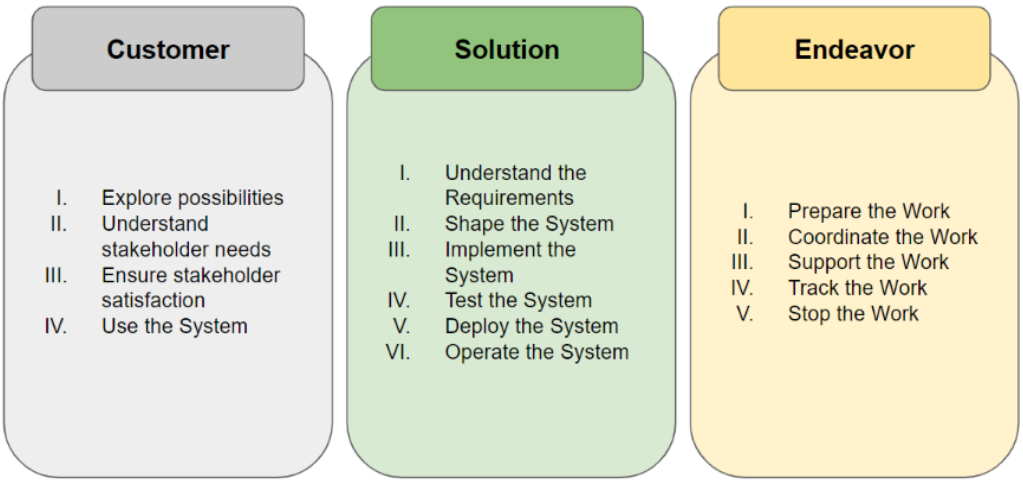

Figure 4 Essence Kernel Activity spaces (Ivar Jacobson International, 2022)

2.4.2 The language

Essence consists of two things: The Kernel and the language. Essence's language aims to describe the methods in software production in the most general way possible. Essence is a standard, so it cannot contain concepts that are too tied to

specific methods. Everything must be repeatable and clear for all. The language of Essence makes use of various easily recognizable forms that can be used to describe practices. According to Jacobson et al. (2022) the elements of Essence:

- **Alphas**, the key elements to monitor the progress of the development
- **Work Products**, the actual results of work
- **Activities**, a task or activity that needs to be performed
- **Patterns**, for describing other things and their relationships

With the Kernel and the language different methods can be Essentialized, this means that they are described using Essence. This enables the described practices to be enriched and mixed with new methods. New methods are available through Essence's common ground. This enables the developers to choose the development methods they feel are best for them, or actually the Essential parts of practices. (Jacobson et al., 2019)

The purpose of Essence is to be as approachable and easy to visualize as possible. Essence components are used in the form of cards. Each card describes the card's goals, status, checklists and prompts to complete the activities. These cards can be used in daily development work. The cards do not contain any difficult concepts, which is the whole idea of Essence. Even new employees can quickly get to where we are going and what their own goals are. (Jacobson et al., 2012.)

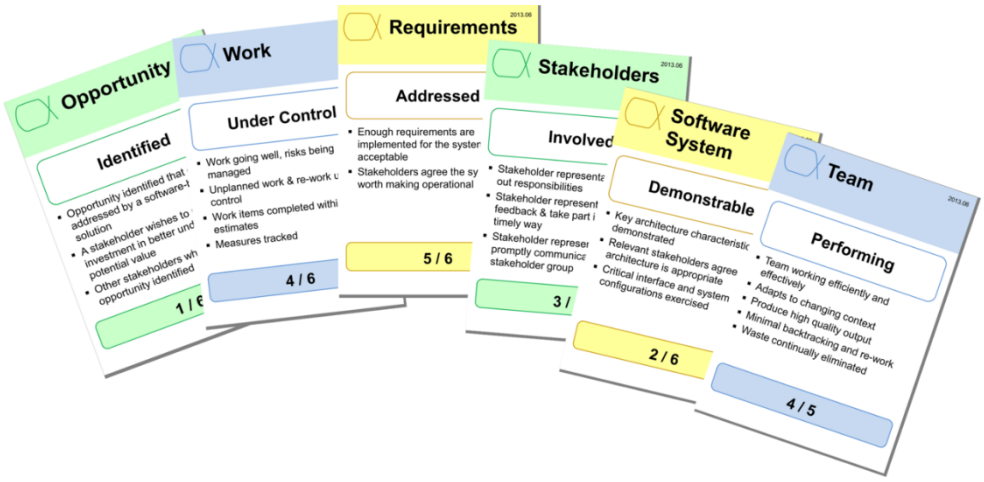

Figure 5 The Essence Cards (Ivar Jacobson International, 2022)

Using cards as part of development is not a new idea. Learning demanding methods is not very popular, that's why Essence offers a new kind of user experience based on the use of cards. The cards can be used to play games that help the development team get a better idea of where the development is going. Playing games with cards is the natural use of Essence.

2.4.3 Essence for Scrum

Essence contains a number of elements and terms, the learning of which is necessary for the utilization of Essence. SEMAT community emphasizes that Essence is not a new method among thousands of others, but a way to combine best practices. Essence provides tools and a library for this. Essence is still a fairly underused tool in agile development methods. Compared to other methods, quite a few studies have been published. In 2022, Jacobson et al., published the article "Better Scrum through Essence" on combining Scrum and Essence. The article describes the process of how to get the most out of Scrum with the help of Essence, Scrum is one of the most used methods in software development. Understanding the benefits of Essence in practice is easier when they are applied to the Scrum that everyone knows. Scrum and its elements were already discussed in earlier chapters. Next, the different elements of Scrum are described in the style of Essence.

The processing of Scrum starts with forming four categories with the help of cards. These four categories are: Alpha cards, the key elements of development, work product cards that describe the wanted results, activity cards that describe tasks and pattern cards that describe relationships and roles. The following table shows how the different scrum elements fit into these categories.

Table 3 Scrum essentialization categories (Jacob et al., 2022)

Alpha	Work product	Activity	Pattern
-Product backlog item -Sprint -Improvement	-Product backlog -Definition of done -Sprint backlog -Increment -Sprint goal	-Product backlog refinement -Sprint planning -Sprint review -Sprint retrospective -Daily scrum	-Self organization -Scrum values -Scrum team -Scrum master

Based on these different elements of Scrum, each can be made into its own card. The card can include a description, relationships, impact and work status. Essence scrum cards presented in figure 6.

Figure 6 Essence Scrum cards (Jacob et al., 2022)

When all the necessary elements of Scrum have been identified and the cards have been formed, the team can start reviewing its own practices. The cards are reviewed and an effort is made to place them on a scale with the most important things at the top and the less important at the bottom. Scale also includes information on how the team performs on them. If the cards have for example checklists, the team will discuss their status and goals. With the help of the scale, it is possible to identify the things that need to be prioritized. Prioritization is especially important in a situation where a lot of cards have been found to be important for the team, but at the same time it is challenging to perform them. Based on these, the team decides on measures to improve things. (Jacob et al., 2022) Everyone who participated in that process will probably get a good idea of the team's situation, even if they haven't been involved for a long time.

Essence offers Scrum ways to improve practices. Essence offers opportunities for many stakeholders. Employees understand the software development process better when it is described more clearly and down to earth. Application development teams can combine and modify existing methods by leveraging the guidelines, methods and elements compiled by Essence. With the help of cards, the whole team can be included in the development process. A common language facilitates the coordination and formation of groups. the entire sector also benefits as a whole from the fact that it has a common basis for bringing practices to the fore and their comparison is easy. (Jacob et al., 2022)

3 CONTINUOUS SOFTWARE ENGINEERING

Chapter describes the goals and elements of continuous software engineering. DevOps is also presented, the goal of which is to combine the activities of development and IT teams, which traditionally work very separately. After DevOps, BizDev is also introduced, which aims to combine business strategy and software development.

Agile methods provide the organization with a framework for software development, which can be used to improve the flexibility, efficiency and especially speed of the development work. Software must create value for its users, to create value the software or system must work, meet the requirements criteria and be up and running quickly. Because of this, organizations want to be able to produce new software quickly and efficiently. According to Dornenburg (2018) the popularity of DevOps rose at the point when technology became business.

Continuous software engineering is an emerging trend in research and practice (Shahin, Ali Babar & Zhu, 2017). The organization must take into account areas other than development in order to be able to operate with a continuous software engineering model. (Fitzgerald & Stol, 2017.) Around continuous software engineering, there is also a reflection on how security matters are handled when development is done at a fast pace (Rajapakse et al., 2022). Continuous software engineering refers to an organization's ability to develop software or its parts in several parallel processes. Continuous software engineering involves various development activities such as continuous integration, continuous deployment, continuous delivery and continuous testing. (Johanssen et al., 2019.)

3.1 DevOps

DevOps is a software development and delivery framework (Forsgren, 2018) and it was introduced in 2009 (Banica et al., 2017). It includes practices that enhance collaboration between software development and IT operations. DevOps approach increases organizations to deliver faster with higher quality products.

The idea of DevOps is to break barriers between teams that should communicate with each other. Many problems in software development are caused through poor communication that could be avoided. (Ebert et al., 2016.) The goal of DevOps is to speed the delivery of business value for customers (Lopez-Fernandez et al., 2021).

DevOps aims to bring the knowledge from end-users to development teams, so that teams can respond to changes rapidly. DevOps is all about flexibility and speed. DevOps can be described as an extension to Agile methods due to the efficiency and agility that it brings to organization. (Banica et al., 2017.)
DevOps tools and methods have reduced the old cultural divide between software development and IT operations. The new organizational structures have had a chance to be born. One example is virtual teams, that can be formed easily to include people from both teams. (Alnafessah et al., 2021.)
The DevOps practices highlight the importance of automation. DevOps aims to release products, applications and updates at a faster pace, because of this, many processes should be automated. Continuous development, continuous integration, continuous testing, continuous deployment and continuous monitoring are the key concepts of the DevOps lifecycle. (Alnafessah et al., 2021.)
The two basic pillars of DevOps are automation and interaction with people (Leita et al., 2020). According to Lwakatare et al. (2019), one of the challenges of implementing DevOps was identified as the situation where DevoOps is promoted from the bottom up. That is, the developers have to convince the managers of its benefits. DevOps lifecycle is presented in figure 7.

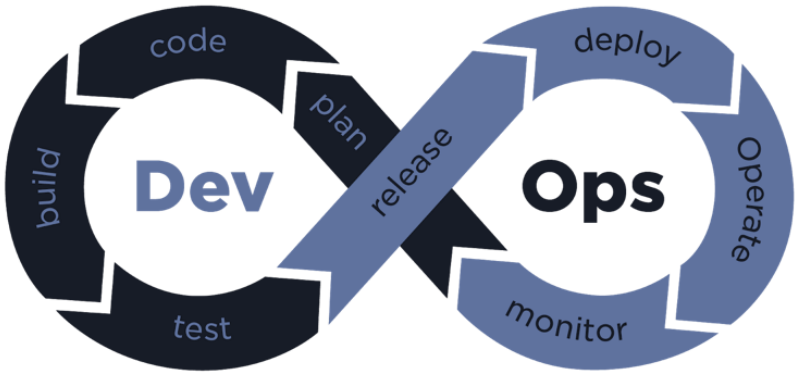

Figure 7 DevOps lifecycle (Alnafessah, 2021)

3.2 BizDev

The purpose of DevOps is to combine software development and IT operations. However, the idea of DevOps must be expanded to cover a larger part of the organization if the goals of software engineering are to be achieved. The framework BizDev has been developed for this purpose. (Fitzgerald & Stol, 2017.)

Continuous software engineering contains the full software development life-cycle, that can be divided into three different phases: Business Strategy & Planning, Development, and Operations. All phases contain different continuous activities. (Fitzgerald & Stol, 2017.)
Fitzgerald and Stol (2017) introduce the term BizDev to describe the link between business strategy and software development. Development or business strategy cannot take place in silos, the transparency of everything improves the success of continuous software engineering. Today's softwares and systems are becoming so complex that the traditional model of software development is not functional. There will always be changes, both through the market, the law, and customers. Communication and cooperation between the teams must be enabled in the initial phase of the application development life-cycle and must continue throughout the entire life-cycle. BizDev complements the mindset of DevOps by expanding it to include business strategy and development. The continuous activities of business strategy, development and operations described in figure 8.

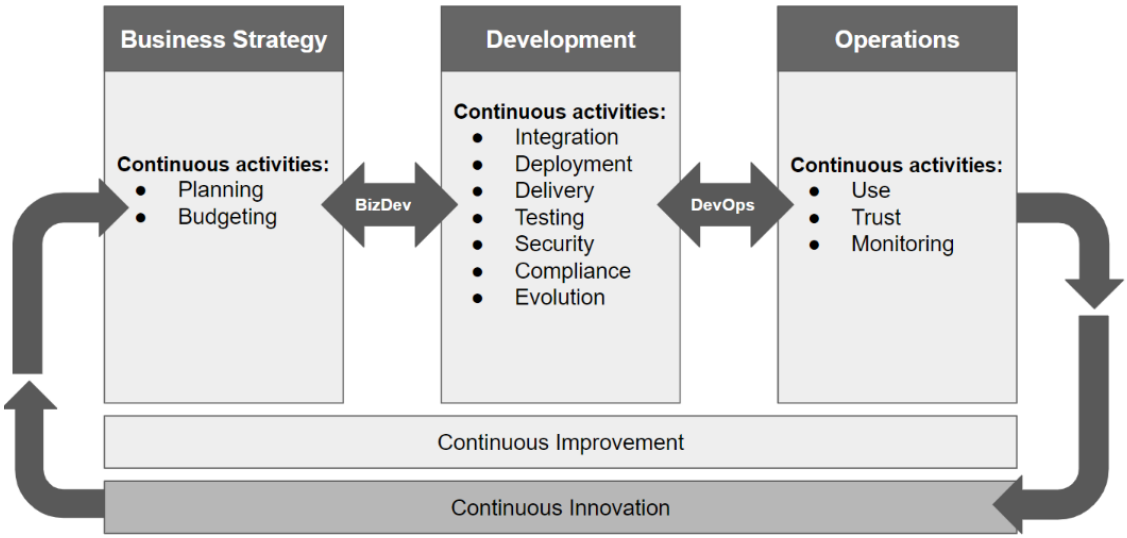

Figure 8 DevOps and BizDev (Fitzgerald & Stol, 2017)

The business strategy activities identified by BizDev are continuous planning and continuous budgeting. A characteristic of continuous planning is that the plans are constantly under review. Changes are made to justify what changes are happening in the market or business area. Plans should be dynamic and transparent. It is not worth making plans too far into the future that are challenging to change. Continuous budgeting means that budgeting should be as flexible as possible, so that it is easier to prepare for changes. Resources should be able to be distributed during the year as well. Combining these activities with development activities improves the business team's understanding of the software development requirements and helps development teams understand the reasons why development is done from a customer and market perspective. The main

thing in everything is effective communication between teams. (Fitzgerald & Stol, 2017.)

According to Fitzegrald and Stol (2017), with the success of BizDev and DevOps, the end result is continuous improvement and then continuous innovation. Continuous improvement refers to the fact that in development one knows how to critically examine the points that act as slow-downs to rapid development and aims to prevent them. If the processes and operating methods are correct, operations should improve continuously. With the help of sustainable procedures, the organization can respond to changes effectively and remain competitive. Continuous innovation describes when an organization has opportunities to design ways to produce added value for customers. The possibility of continuous innovation requires that all phases of DevOps and BizDev work.

4 RESEARCH DESIGN

This chapter reviews the goals of the study and chosen research method. In addition, the selected data collection method and background information about the interviewees are presented. The chapter also explains how the interviewees were selected and how the material was analyzed. Finally, the reliability and validity of the study will also be reviewed.

4.1 Goals of the empirical research

Today's software development is fast-paced and those who fear change cannot keep up with the competitors. Many different methods have been developed to support agile software development and surely many of them know at least the name. Current methods differ from those previously used in that they make it easier to react to changes. The publication of the Agile manifesto in 2001, started the change (Beck et al., 2001).

This research aims to investigate agile development methods in the continuous software development and identify which frameworks are in use and identify the challenges associated with utilizing them. The benefits of agile methods are already becoming clear to everyone, but how to implement them in practice is difficult.

The case company of the study provides SaaS-based systems for organizations of various sizes. With the help of systems, organizations manage payroll, human resources and accounting. The company wants to improve the processes between software development and business operations. Both units contain many different teams from different areas of expertise. Every team is needed to ensure that the journey of a new feature or product in an organization is always successful from the design stage to the end product for the customer. Successful processes also ensure that the customer feels that they have gained added value.

A qualitative case study was chosen as the research method. The choice of research methods was influenced by the fact that the research focuses on the

activities of one organization. In qualitative research, the aim is to understand the phenomenon under investigation, it aims to understand the meaning and purpose of the phenomenon. With the help of a case study it is possible to obtain more detailed information about the phenomenon under investigation. (Guest at el., 2012.)

4.2 Data collection

The empirical data for the study were gathered from semi-structured interviews. The interviews used a predefined questionnaire, which was built between different themes. Themes must be based on the theoretical framework of the study, and they must contain what is already known about the studied phenomenon. (Tuomi & Sarajärvi, 2002.) When using semi-structured interviews, the interviewer can ask more specific questions about themes, if things come up from the interviewee that you want to open up in more detail. In the semi-structured interviews the questions are the same for all the interviewees but there is also room for improvisation. In semi-structured interviews, it must be remembered that the interviewees are quite capable of choosing what to tell and what to leave unsaid. The interviewees also think much more about what the interviewer is aiming for with the questions. (Alasuutari & Alasuutari, 2011.) The advantage of an interview is that the interviewer can ask more detailed questions if the answers do not immediately answer the questions (Tuomi & Sarajärvi, 2018).

The goal of the study was to investigate the phenomenon of the use of agile methods in continuous software development. For this reason, the interviewees were chosen from roles that worked mainly in software development. In the selection of the interviewees, the aim was to ensure that the selected interviewees would comprehensively represent the entire development group. Some of the employees refused the interview and suggested another colleague for the interview. All interviewees work on the same product, but in slightly different positions based on the job description. More experienced and also newer employees were selected. There are also different roles, from software developers to team managers and scrum master. Background information of eight interviewees presented in table 4.

Table 4 Title and work experience of the interviewees

Interviewee	**Work title**	**Work experience**
Interviewee 1	System Analyst	30 years
Interviewee 2	UI Developer	5 years
Interviewee 3	Team Manager	8 years
Interviewee 4	Solution Architect	8 years

Interviewee 5	Scrum Master	24 years
Interviewee 6	Team Manager	25 years
Interviewee 7	Software Developer	10 years
Interviewee 8	Product Owner	20 years

All interviews were conducted remotely using the organization's own tools. The interviewees had moved to work from home, due to the COVID-19 pandemic. The interviewees were asked their favorite way to conduct the interview, and for all of them it was remotely. The interviews took place in June 2021. The interviews were conducted in Finnish and 50 minutes were reserved for each interview. The interviews were recorded for transcription.

4.3 Data analysis

The analysis of the interview material starts when the recorded interviews are transcribed. The transcribed material forms the research material. The accuracy of transcription is affected by the goal of the study and the method of analysis. It is not necessary to transcribe all pauses or sounds in every study. (Hyvärinen et al., 2017.) In this study, no filler words were included in the transcription. A total of 48 pages of transcribed material were collected.
The challenge of qualitative research is that many interesting things often emerge from the material that the researcher did not even expect. That is why it is important to be able to accurately choose what to focus on when analyzing the material. (Tuomi & Sarajärvi, 2002.) Thematic analysis was used to analyze the data. It can be used to study themes based on qualitative data. Thematic analysis is a frequently used method in qualitative research. Based on thematic analysis, the aim is to identify recurring themes and patterns in the data. Categorization and coding were utilized in the thematic analysis and also in this study. Themes can be based on a literature review or appear completely new. (Guest at el., 2012.) In this study, most of the themes were obvious and expected.

4.4 Realiability and validity

The quality of scientific research is often evaluated through reliability and validity. Reliability describes the repeatability of research and plays an important role in the evaluation of research results, and its function is to describe the research's ability to give similar, non-random results. Can the results be independently repeated, a reliable study has been conducted in such a way that a study based on the original study gives the same result. (Tuomi & Sarajärvi, 2002.) In this study,

an effort has been made to improve reliability by accurately describing the phases of the thesis, such as the selection of the interviewees, data collection and analysis. Also the questionnaire used in the interviews has been added to the appendices.

The validity of the research, on the other hand, means the validity of the research, it describes the ability of the research method to study exactly the subject area or characteristic that is intended to be studied. Results and conclusions are evaluated on the basis of validity. (Tuomi & Sarajärvi, 2002.) The interviewees for the study were chosen in such a way that the material collected based on it is well suited to answering the research questions. The interviews were also conducted within a short period of time, so that the results cannot be influenced too much by the researchers own goals.

5 EMPIRICAL FINDINGS

This chapter presents and reviews the empirical findings of the study. The most important findings are reported as empirical conclusions (EC) and primary empirical conclusions (PEC). Data were collected by interviewing eight employees utilizing semi-structured interviews. The goal is to understand what are the best practices for continuous software development, with an emphasis on agile and lean. The results are presented according to the themes of the thematic interview.

5.1 Development process

The first theme in the interviews was the development process. Interviewees were asked about the development process and how it is planned to take place. The purpose was also to find out if the interviewees could name the different methods they use on a daily basis. Interviewees were also asked about strengths and challenges based on their development process. Six out of eight interviewees were able to somehow define which methods or frameworks were used.

> We don't have any kind of real scrum. In a way, however, we use the same style, deliveries dictate the schedule of what is ever done. For bigger things, for example, interfaces then have their own schedules. It is intended to operate within delivery cycles. We do not have daily Scrum meetings, but every day is a team meeting. And that is where I go through open questions and what the pool has. -Interviewee 1

> There is talk of scrum, but it is not pure scrum. More Kanban spiced with scrum-related dailies and sprint changes. There is really no kanban and no scrum. Methods will be visible in such a way that it will be possible to roughly handle the activities that are to be developed. -Interviewee 7

The answers show that some of the interviewees do not know how to name the actual methods, but are aware that they use elements of the Scrum, for example. These findings form empirical conclusion EC1.

EC1: Used methods and frameworks are known, but their use is not systematic.

However, some of the interviewees were also able to describe very precisely the elements of Scrum and what is in use.

> We have Scrum, done in three-week cycles, so that the development process stays on schedule. Every other month another app delivery and every other month another app. We have daily meetings every morning for half an hour. There are a few more people out there for what Scrum's recommendation is. We have about 15 participants every morning. Two minutes for everyone to speak for themselves. The speeches are short and concise. After the sprint, the breakpoint will go through the previous sprint and plan whether the same thing will continue for the next sprint. If it is a larger whole then it will not be completed in three weeks and it will take many months and several sprints. -Interviewee 2

These findings show that the use and know-how of processes and methods is possessed by only a few. This forms the empirical conclusion EC2.

EC2: The knowledge of the practices used has been emphasized only for the leaders of the processes.

PEC1: The currently used frameworks of agile methods are known, but use in development is not clear to all involved.

Although the knowledge of the practices seems to be concentrated in a few people, the other interviewees were also able to praise Scrum and how it fits the current situation well. The interviewee 5 described that Scrum is suitable for the current work situation because the backlogs used are long. Kanban for example would not be useful.

> Scrum has been right for us because we have long backlogs and it is useful to take things in a different order. That is, Kanban would not work for us as such. There is always an assessment of that next episode and what kind of entities there are. The Agile world and from there Scrum has been the frame of reference where it is delivered. -Interviewee 5

These findings form the third empirical conclusion EC3.

EC3: Scrum is well suited for development when used backlogs are long.

It is clear from the responses that the interviewees do a lot of work over teams. Interviewee 3 described that back and front end development teams are using Scrum or its variation. Cloud team uses mainly Kanban related methods.

> The development partly uses Scrum for the back and front End. Daily meetings (daily Scrum) are in use, so some parts of Scrum. In the cloud, we have a Kanban that works better because the deliveries are not actually sprints. However, the work is guided by development requests and cases coming through tickets. -Interviewee 3

Teams are formed for projects, so different practices of operation are also dependent on them. This can partly explain why there are differences between the interviewees in terms of how familiar they are with different methods like Scrum.

Interviewee 6 described that they do not use Scrum, but they will be involved in the Scrum team if there is going on larger projects.

> We do not have Scrum, but we will be involved in Scrum if we have larger projects ahead. However, there are team sessions, always in the mornings, where things related to implementations are reviewed. Get answers to open questions quickly. The aim is to go through things on a daily basis, so that there will be no meeting in a week's time where you can go through things. Let's see what is urgent and what is going on. -Interviewee 6

> EC4: People get involved in scrum when they are added to the project team.

Respondents were also asked how long the current methods have been in use and the answers were a bit difficult to interpret. In general, based on the responses, Scrum ideology has been in use for years, but in fact, Scrum has only been used for a few years. That is, the teams are quite young in the use of Scrum, who may also be involved in explaining why not everyone has an idea of what Scrum is and how it is reflected in the use of the methods.

> Scrum ideology has been in use for years, Sprint design has been in use. Whenever the sprint ends, there is a demo that introduces the products internally to different teams. After the sprint there are retros and we go through what went well and where to develop it then the next round starts. A lot of dealing with customers, weekly meetings with customers. -Interviewee 8

> Scrum has been in use for a few years mainly in development. There are retros after the sprint and it looks at what went well and what should be developed. -Interviewee 3

> There is a way we applied for Scrum. We have been introduced to agile, sometime in 2008. We have been working on this collaboration for so long, inevitably there will be points where we deliver to the team's advantage. -Interviewee 5

These findings form the fifth empirical conclusion EC5.

> EC5: The elements of Scrum have been in use for a long time, but the actual Scrum ideology has only been around for a few years.

Interviewees were also asked about the other frameworks and whether they were used. Some of the interviewees mentioned and recognized SAFe, but have not really used it with this current work team. Three interviewees out of eight recognized SAFe. Interviewee 7 had previously used SAFe and could say that Scrum is not enough for enterprise level development and SAFe includes elements that would help to work more efficiently. The current Scrum is not enough because things have been picked out that are not really part of Scrum. With SAFe they could manage bigger projects more efficiently.

> We have challenges, the basic Scrum is no longer suitable for this kind of enterprise level development. There should be some other tools, for example Scaled Agile framework. SAFe should be picked for their delivery slices. In the previous company, I used it and worked well when the outputs of several organizations had to be ready and tested. There is no person who will handle all the teams delivered by a non-release train engineer.

> However, SAFe is a pretty large package, but if you picked them up a few good ideas for this one. -Interviewee 7

SAFe has not been talked about much, and it has not even been considered in terms of development, these findings provide the sixth empirical conclusion EC6.

> EC6: There are challenges with Scrum and it is not completely suitable for enterprise level of work. However, there is no mention of other options, such as SAFe.

Interviewees were also asked about Agile and Lean. All eight interviewees recognized terms and had sometimes heard about them. When asked in more detail how they appear at work, there was a lot of scatter in the answers. Half of the respondents could not say whether Agile and Lean show up in their daily work. The other half of the respondents indicated that it appears this way because Scrum is enabled. There were also mentioned by three interviewees that the teams work approach is inherently agile.

> There has been talk, but it has not been taken into account. However I think that way we work is inherently Agile. -Interviewee 4

> Big development projects are when it is known that there are Scrum meetings. The idea is to strive for rapid development and to bring new things to the interim deliveries. -Interviewee 6

It seems that the terms Agile and lean were very familiar, but when asked in more detail how they are seen at work, one cannot directly answer. These findings form the seventh empirical conclusion EC7.

> EC7: Agile and lean were well known as terms, but describing them in development work was difficult.

Interviewees were also asked about the roadmap and how much it affects development. All interviewees could tell that larger entities come through the roadmap. However, following a roadmap on a schedule is not always easy, as production often becomes another busy task. It reportedly had no effect on daily activities. The roadmap is also visible to customers, so customers can also wait for certain features to come into production. That is why it is important to stay on schedule for the roadmap as well. However, one avoids putting overly precise schedules in the roadmap, because unpredictable things in the development can always affect them.

> Larger entities we are moving towards. The roadmap tells you what you want the product to do in the future. The vision is far into the future and we will move towards it with the help of a roadmap, however, there will be a lot of things outside of the roadmap that will go beyond it. However, something new for customers is always desired for every delivery. From the backlog you can pick up bigger and smaller things for development. -Interviewee 8

The purpose of the roadmap is to show the direction of development and does not show exact dates or schedules. This forms empirical conclusion EC8.

> EC8: The roadmap influences the development, but its purpose is to act more as a direction towards which we want to go. Little effect on daily work.

Almost all interviewees recognized the term DevOps, only one interviewee could not describe it. It is clear from the answers that although the term is familiar, it is not included in the daily work, even if some of the interviewees felt that it could be useful. It was also noticeable that there were some differences in the interpretation of the term DevOps and what it really meant. Interviewees were also well aware that current work habits do not support DevOps. It was felt that everyone is working in silos and the skills are very limited to certain people. Everyone has their own things to do and knows things that others hardly know.

> The term is familiar, not present in the work of our team. We have our own cycle to move forward with. Within our team, everyone works alone with their own area of strength. -Interviewee 4

> Traditionally, there has been a lot of separation between product development and production. Product development does something and then production takes over. That is, it completely jumps out of product development. The production team will then support the software in future production. However, we should go towards DevOps. -Interviewee 3

According to Ebert et al. (2016), the purpose of DevOps would be to dismantle these silos and improve cooperation between different teams. Bad communication or the lack of it altogether is one of the most common challenges in software development. Interviewees felt that they should go more in the direction of DevOps. These findings form the following empirical findings EC9 and EC10.

> EC9: Current work methods do not support DevOps.

> EC10: The work is done in silos and it would be important to dismantle them.

> PEC2: Current working methods do not support DevOps, everyone has their own areas of expertise, which affects the fact that work takes place in silos.

Interviewee 5 also said that there is need for DevOps in the development, but it would need so much work to change current work habits that it will take time to achieve.

> Our readiness is not at that level at the moment. It is traditionally done to have delivery dates announced to customers and developers to test and testers to test. We are packaged and exported to the world at pre-arranged intervals. DevOps would require a lot from us and the process, you will be able to evolve and change your operations. Efforts would be needed at the moment, but we are moving towards that. -Interviewee 5

The weaknesses of the current methods are known, but based on the answers, changing the current methods is perceived as such a heavy process that there is no motivation for it.

> EC11: Changing current methods is perceived as such a heavy process that there is no motivation for it.

Even though DevOps seemed like a very familiar term among the respondents, it was felt that not enough is known about it and that there would be a need for a new role that would be able to guide the teams work towards DevOps. Interviewee 7 felt that there is not enough information about DevOps.

> It is a familiar concept for a few people, but we do not have any DevOps engineers. There is no designated person to use DevOps, so its use is quite limited. It would be useful, because now you have to manually move all the delivery packages, why even need to manually move them could be automated. -Interviewee 7

> EC12: There is no defined responsible person or role that would be responsible for DevOps.

> PEC3: The need to break down silos and change working methods has been identified, but the change is perceived as too difficult to implement and no one is responsible for it.

It seemed that there were many differences between individuals in how they perceive current methods to work in the development process. Some think the current methods are good and have always been. On the other hand, there is a need for change to make it more effective. According to the interviewees, there were many strengths in the current methods. Flexibility was highlighted and the fact that when monitoring the work situation on a daily basis it is transparent and allows for a quick response to changes.

> Flexibility, on a daily basis when looking at work situations and being able to react to whatever comes, is perhaps the biggest thing that comes to mind. Teamwork, it can then be done by a group when you look at it so you can find help and opinions when everyone is clearly aware of things. -Interviewee 1

> EC13: Flexibility and transparency in working methods were perceived as the most important strengths.

According to Misra et al. (2011), agile methods need self-directed and motivated people to work successfully and efficiently. Interviewee 2 explained that the team is highly skilled and everyone knows their role, that is why it is easy to agree on things. Also there is not too much pressure from the managers and there is no hierarchical setup and everyone will get along well.

> Skilled and motivated team. The team knows their job, what they do is monitored daily, a morning meeting of what everyone is doing and the project manager and scrum master see how things are going. There is no pressure at the supervisor level and no hierarchical setup and we get along well. -Interviewee 2

Based on the answers, it seems that the organization relied on the skills and high motivation of individuals. A good and motivated team can produce results even if the methods themselves are not clearly defined. These findings form empirical conclusion EC14.

> EC14: A skilled and motivated team was perceived to be able to produce good results, even if the practices were not clear and functional.

> PEC4: A motivated and skilled team can achieve results as long as the working methods are transparent and flexible. Although the development framework is not clearly defined.

Interviewees said they would like to guide development more into way of Scrum ideology. However, development suggestions and repairs from customers often clutter the schedule. Interviewees recognized many challenges, four out of eight interviewees raised these same challenges regarding insufficient resources. Based on the roadmap, everyone has long-term development targets that need to be completed. Every day, however, there are urgent production cases that must also be handled by the same people. Means that issues that need to be fixed or other bugs or development suggestions emerge from the production with the tickets. Developers do not have time to focus only on new development, but time is spent on a lot of other things.

> We are involved in many things and there are big development tasks coming from the roadmap that take up work time and planning and other things. Then there is also everyday life that needs to be produced. - Interviewee 1

> It is challenging when product development cannot focus only on development. When it is noticed that production needs to be taken care of, it must be decided immediately whether the developer will stop his own work and start working on a new case. -Interviewee 5

A solution to the resource challenge was needed, some of the interviewees pointed out that some kind of division or process was needed. Which could be used to clearly determine how the development items according to the roadmap and the cases from the production side would be handled. These findings form empirical conclusion EC15.

> EC15: The most significant challenge was felt to be that when product development can't focus only on development instead, time and resources are spent on repairs coming from production.

> PEC5: Bug fixes and improvements from production related to system maintenance prevent continuous development and improvement. There is no clear practice for prioritizing and using resources.

According to Schwaber & Sutherland (2017), the scrum teams size should be a maximum of nine, so that there is enough time and meetings do not become too

heavy. Some of the interviewees identified the large size of the Scrum team as a challenge. Meetings may have up to 15 people and there is not always enough time for everyone.

> I have sometimes thought about that when there are 15 people in the daily scrum and it takes half an hour, so it's not quite optimal. You have to listen to the things of 14 other people that don not concern you. If there were fewer people in the daily, it could be good. -Interviewee 2

> EC16: A Scrum team with too many people involved was perceived as difficult and a smaller team would work more efficiently.

5.2 Agile teams

The one most important goal of all Agile and Lean methods is to improve an organizations ability to respond to change. According to Misra et al. (2011) for example, the flexibility of roles and operating methods improves responsiveness. Therefore, the interviewees were asked how the current methods can help react to changes. The answers show that changes are coming from several different directions. A change in the law can cause a lot of reforms and changes in the rules. New orders and repair requests from customers also cause a need for resourcing. The system is also constantly being developed to work better and include more functions. Changes in third parties can also have a significant impact on ongoing developments. A lot of interfaces have been made to communicate with these third party applications. Prioritizing and implementing all of these changes is a daily challenge, which all interviewees recognized. For five out of eight interviewees, it was clear who is responsible for decision-making in prioritization situations. Interviewee 2 described that the product owner or scrum master is responsible for deciding the delivery schedule.

> The priority question for the product owner or scrum master is whether to make an intermediate delivery if, for example, something is not successful at all in production. It is classified as a high priority bug for the next delivery. If it is a matter of customer wishes, then they will be included in the plan for next sprints. -Interviewee 2

> EC17: In terms of resourcing, prioritization was the task of either the Scrum master or the product owner.

Interviewees said that customers can report bugs and make orders through the ticketing system. Ticketing is closely monitored and quickly noticed if one needs to react. Especially after the new version release, tracking is even more accurate. When it is noticed that there are things in production that need to be addressed, a quick meeting with developers and other experts is held. The meeting divides the tasks and evaluates how quickly the change or fix can be implemented. Sometimes, of course, there may be situations where you are not immediately sure where the situation is and a lot of work needs to be done before the situation can

be fixed. It is important that everyone is aware of their own job responsibilities. Interviewee 3 said that sometimes it is more important to prioritize things that are more visible for the customers.

> The response varies. Prioritization should be improved, we invest in everything that is visible to the customer. There is a clear division of which areas go to the different teams. -Interviewee 3

> It depends on how quickly if something comes into production, it will be determined immediately who will start doing it. Critical issues are fixed and tested quickly. -Interviewee 8

It seems that with the current methods teams were able to solve the prioritization issues. Especially the most important cases could be dealt with quite quickly. However, a clear process is missing in decision-making and it was clearly needed for people. These findings form an empirical conclusion EC18.

> EC18: Current methods have worked in urgent situations. However, the process should be improved so that everyone knows for sure how to act when it is a really important case.

Interviewee 6 explained that backgrounds must be known before any changes are made to the system. Changes that are too fast can cause even more problems, so you need to know the nature of the case. Even if this would result in a delay in repair or delivery.

> Whether it is a change in the law or a customer order, there must be information about what it is based on. The changes can cause major changes in resourcing and work tasks, for example, changes in the law in particular. -Interviewee 6

> EC19: The identification of changes and their impact on many different entities must be clarified in cooperation with other teams.

In the case of a change proposed by the customer, the customer is also involved in the development process. The customer is contacted several times and they go through exactly how they want the new subscription to work, for example. Product users are often best placed to tell what all the different cases can be encountered when testing.

> When an order comes to us, a specification has been made for it to be submitted to the customer for approval. It may be shared several times and clarified questions will be sent. Big changes also involve customers in piloting, which is really good because in production there is not enough imagination for all that can be met and it is good when piloting already gets something out of it. -Interviewee 1

> EC20: Customers were included in the development process.

It seems that interviewees considered they have a good ability to respond to changes. When changes are noticed from the ticketing system or other channels it is clear what happens next. However, there is no specific formula or operating model that all interviewees could have explained. However, the responses

indicated that a team would be assembled fairly quickly to determine the severity of the situation in the event of a bug or system error, for example. If there is more time for changes, the process is also clearer in many people's words, i.e. it is time to plan for which sprint the change will be implemented.

> PEC6: When development has to react quickly to things coming from production, everyone has to know their role and the process has to be clear throughout the organization. Otherwise, it is impossible to solve the problems.

5.3 Software quality

The DevOps practices recommend automation in all possible stages of software development, with the help of automation it is possible to improve the quality of the software. With automation, the organization can release products, applications and updates at a faster pace. (Alnafessah et al., 2021.)
Interviewees were about the quality management of the continuous development in their teams and organization. The purpose was to find out how to keep the quality of the system and the work high. In the case organization the development is continuous and new versions of the system are delivered monthly, sometimes smaller deliveries are made even weekly. However, all deliveries must be designed, developed and tested before they can be put into production. Almost all of the interviewees, seven out of eight, mentioned that there should be more testing, but its design and implementation are still challenging. The system being developed has become so demanding over the years that testing is also hard and time consuming. It is not possible to verify all test cases.

> The system is so complicated. This side where there are accounting rules. Testing is hard, one should know a lot of things that can make a difference. Testing is the first step. When something is done then you should be able to test that it works properly in every situation. -Interviewee 1

Based on the answers of the interviewees, the software is very large and contains many different rules that affect each other. All this makes testing and quality assurance difficult. These findings form following empirical conclusions EC21 and EC22.

> EC21: The size and complexity of the system affected the success of testing and quality assurance.

> EC22: Making even small changes to the system is challenging, because everything has to be tested for several different scenarios.

Five interviewees also recognized that automated testing could help to improve quality. Automated testing was already in use, but respondents felt it should be increased considerably. However, the testing cannot be based on automated

testing alone, because the situation of customers is so different that manual testing is needed. Automated test cases must be created manually and their aging is also rapid.

> Investments have been made in test automation, not everything needs to be tested manually. However, all automation tests have been manually coded, customers have such different situations. There can be a lot of situations that go unnoticed. Test automation ensures quality and even more for manual testing. -Interviewee 2

> Testing is done by manual testing. Automated testing should be increased and it would improve quality. An automated test could be run at all times to detect broken spots. At the very least, it would make it easier to monitor quality. -Interviewee 3

Automated tests were partially in use, but their utilization was perceived as challenging. Automated tests need a lot of manual maintenance to work. with new version updates, the tests should also always be reviewed. These findings form empirical conclusion EC23.

> EC23: The maintenance of automatic tests was perceived as challenging, because they always have to be updated manually before they are actually useful.

Interviewee 5 explained that in addition to testing, code reviews are also used to ensure the quality of deliveries. Code reviewing usually means reviewing the code with at least one other developer. Reviewing helps to share knowledge and unify developing styles. Furthermore, Interviewee 5 explained that developers also make unit tests in their code.

> There is a review, the codes are viewed through other eyes and in layers. The aim is to ensure quality through testing. Unit tests, encoders perform unit tests on their own codes. - Interviewee 5

Interviewee 5 also described that when delivery approaches there will be the system testing phase that includes double checking of the tests and regression testing.

> As delivery approaches, there will also be a separate system test phase during which duplicate checks will be performed. That is, we check the things that have been coded and do regression testing. The tests go through different functionalities in different browsers than what is being done now. -Interviewee 5

Interviewee 7 explained that more of the functions should be tested automatically and robot tests would help to ensure quality. However tests do not last long and tests would need to be updated often.

> There should be a greater proportion of functions automatically tested.
> Robot tests or integration tests and regression tests. However, the tests are technically obsolete and they no longer test what they should. -Interviewee 7

It seems that all of the interviewees understood the importance of the testing. Testing should not be forgotten, indeed, the challenges lie more in how to increase the number of tests and at the same time improve their quality. It is difficult to model all test cases because the system to be developed is extensive and some of the entities work differently between customers. Automated testing emerged from the responses several times. Increasing automated testing would affect the quality of product according to responses. However, tests become rapidly old with new versions of the product and would take a long time to update. In addition to this, manual testing should be at least as much as it is today.

> PEC7: Testing is key to ensuring quality. Automated and continuous testing improves consistency, but also requires a lot of manual work to work correctly, when changes are constantly made to the system.

5.4 Summary

This chapter summarizes all the empirical conclusions and primary empirical conclusions based on them. Based on the interview data, a total of 23 empirical conclusions and seven primary empirical conclusions were formed, that are presented in the table 5.

Table 5 Primary empirical conclusions

Identifier	Primary empirical conclusion
PEC1	The currently used frameworks of agile methods are known, but use in development is not clear to all involved.
PEC2	Current working methods do not support DevOps, everyone has their own areas of expertise, which affects the fact that work takes place in silos.
PEC3	The need to break down silos and change working methods has been identified, but the change is perceived as too difficult to implement and no one is responsible for it.
PEC4	A motivated and skilled team can achieve results as long as the working methods are transparent and flexible. Although the development framework is not clearly defined.
PEC5	Bug fixes and improvements from production related to system maintenance prevent continuous development and improvement. There is no clear practice for prioritizing and using resources.
PEC6	When development has to react quickly to things coming from production, everyone has to know their role and the process has to be clear throughout the organization. Otherwise, it is impossible to solve the problems.

PEC7	Testing is key to ensuring quality. Automated and continuous testing improves consistency, but also requires a lot of manual work to work correctly, when changes are constantly made to the system.

6 DISCUSSION

This chapter discusses the theoretical and practical implications of the study. Theoretical implications are derived from the research results, which are compared to the theoretical framework. Practical implications are also reviewed, and suggestions are tried to be found for them.

6.1 Theoretical implications

The most important empirical findings of the study and their relation to existing research are presented in the table 6.

Table 6 Primary empirical conclusions and relation to existing research

Identifier	Primary empirical conclusion	Relation to existing research
PEC1	The currently used frameworks of agile methods are known, but use in development is not clear to all involved.	There are many challenges in agile transformation, enough training and planning is needed so that everyone understands the agile methods (Inayat et al., 2015).
PEC2	Current working methods do not support DevOps, everyone has their own areas of expertise, which affects the fact that work takes place in silos.	Working in silos is a recognized challenge (Ebert et al., 2017). Cross-functional teams ensure high-performance (Remta & Buchalcevova, 2021).
PEC3	The need to break down silos and change working methods has been identified, but the change is perceived as too	Meets the challenges that occur in the implementation of continuous software engineering (Fitzgerald & Stol, 2017).

	difficult to implement and no one is responsible for it.	
PEC4	A motivated and skilled team can achieve results as long as the working methods are transparent and flexible. Although the development framework is not clearly defined.	Meets the challenges that occur in the implementation of continuous software engineering (Fitzgerald & Stol, 2017).
PEC5	Bug fixes and improvements from production related to system maintenance prevent continuous development and improvement. There is no clear practice for prioritizing and using resources.	Meets the challenges that occur in the implementation of continuous software engineering (Fitzgerald & Stol, 2017). The goals of DevOps is to ensure continuous software development lifecycle (Alnafessah et al., 2021).
PEC6	When development has to react quickly to things coming from production, everyone has to know their role and the process has to be clear throughout the organization. Otherwise, it is impossible to solve the problems.	Meets the challenges that occur in the implementation of continuous software engineering (Fitzgerald & Stol, 2017).
PEC7	Testing is key to ensuring quality. Automated and continuous testing improves consistency, but also requires a lot of manual work to work correctly, when changes are constantly made to the system.	Lean development tools to ensure quality: enough testing, incremental development, workflow and automation. (Poppendieck & Cusumano, 2012).

The first primary empirical conclusion (PEC1) states that the teams understand and know the frameworks. Almost everyone was familiar with Scrum and Kanban and had also worked with them. however, their use in own development work is unclear. It is felt that it is not certain which methods and their parts are actually in use. The findings are in line with previous scientific literature and research. There are many challenges in Agile transformation, enough training and planning is needed so that everyone understands the Agile methods (Inayat et al., 2015). One of the biggest challenges in using agile methods is describing concepts and terms in a way that everyone can understand them (Conboy & Carroll, 2019).

The second primary empirical conclusions (PEC2) states that the current operating methods of the teams do not support the DevOps operating model, everyone has their own specific skills. Competence in certain areas is concentrated in a few or even one team member and because of this work is done in silos. The

findings are in line with previous scientific literature and research. Working in silos has been identified as a challenge in several different studies and theories. Working in silos is a recognized challenge (Ebert et al., 2017).
The third primary empirical conclusion (PEC3) states that dismantling the silos has been identified as a goal, but the measures and tools to implement it are missing. The fourth primary empirical conclusion (PEC4) states a motivated and skilled team can succeed in development even if the frameworks, practices and operating models are incomplete. The fifth primary empirical conclusion (PEC5) states that, maintaining production and developing new features creates challenges. Resourcing is not clear and there is no operational model for it. The sixth primary empirical conclusion (PEC6) states that the organizations success in responding to change is significantly influenced by how the processes and operating models are planned in advance. These four primary empirical conclusions are in line with previous scientific literature and research. Meets the challenges that occur in the implementation of continuous software engineering (Fitzgerald & Stol, 2017). The goals of DevOps is to ensure continuous software development lifecycle (Alnafessah et al., 2021).The Adaption of DevOps needs cultural and technical transformations in the organization (Zhu & Champlin-Scharff, 2016).
The seventh primary empirical conclusion (PEC7) states that testing is one of the most effective ways to ensure the quality of applications. Automatic and continuous testing are at the core, but their implementation and maintenance is largely manual work. The findings are in line with previous scientific literature and research. One of the tools of lean development is built-in quality. Lean development tools to ensure quality: enough testing, incremental development, workflow and automation. (Poppendieck & Cusumano, 2012).

6.2 Practical implications

Based on the findings of the study, the practical implications are presented in table 7. The practical implications are based on previous research and theory.

Table 7 Practical implications

Identifier	Implication for practice
PEC1	Clarifying the used methods such as Scrum with the help of Essence.
PEC2	Changing the structure of teams, sharing expertise more widely in the organization. Cross-functional teams to ensure high-performance
PEC3	Organizational working methods towards DevOps, responsibility to the team, instead of one person.

PEC4	With the right methods of operation, the possibility of further improvement.
PEC5	The elements of continuous software engineering are missing.
PEC6	DevOps and BizDev to support the organization's ability to react quickly.

Practical implications have been formed on the basis of primary empirical conclusions. First found that even though teams use popular methods such as Scrum. The elements of Scrum are certainly very familiar to the Scrum master, but it may still be unclear to the other participants why the current method is used and which method it is. It is always necessary to clarify the goals and the frames to be used. Essence has been used for this purpose in previous studies.

The second and third practical implications relate to team work in silos. Even if the challenges in terms of silos are recognized, and one would like to get rid of them, measures are still not taken. This is certainly related to the fact that the team does not know how to dismantle them. The team recognizes the need for a role that would make the necessary changes to working methods. However, the core of DevOps is the team's central operation and decision-making.

The fourth practical implication relates to skilled and motivated team. It is felt that even though the practices, frameworks and methods of operation are not exactly in order, the team is able to produce results. Behind the success of many agile models is often a motivated and skilled self-managing team. This provides a good starting point for continuous improvement.

The fifth and sixth practical implications relate to guiding working methods in the direction of DevOps and BizDev. The teams should produce programs and features whose maintenance would also be possible easily and partially by the same people who have been developing them. If the organization wants to be quick to react to changes, the entire structure must be built around it.

7 CONCLUSIONS

The purpose of this chapter is to compile the theoretical and empirical part of the thesis. This chapter presents the results of the thesis by answering the research questions. In addition, the chapter considers the limitations of the thesis and possible topics for further research.

7.1 Answers to the research questions

The goal of the research was to find out the utilization of agile methods in software development, the goal of which is continuous software engineering. In addition to the main research question, two research questions were formed, the answers to which were to be found in previous studies and to examine what can be found on the subject in the scientific literature. The first research question to which the study is trying to find an answer from the scientific literature was:

- Which agile development methods can be used in continuous software development?

Answer to this research question were found in previous studies and scientific publications. The theoretical part described the most used methods such as Scrum, Kanban, SAFe and LeSS. In addition, the benefits of Essence were described.

The second research question tried to understand the challenges of agile methods:

- What are the challenges in applying Agile methods?

Answer to this research question were found in previous studies and scientific literature. The answer described the most relevant challenges that organizations and teams can face when using agile methods.

The goal of the study was to answer the main research question:

- How to adapt agile methods in continuous software development?

This question was answered with the empirical part of the study, where members of the development teams were interviewed. Based on empirical conclusions, several different challenges were identified. Mainly the challenges have already been identified in existing literature and research. It is important to note that those involved in software development are well aware of the different models and frameworks available.

The goals and elements of successful software development are also known. However, the biggest problem is how to use these in practice. In terms of the success of continuous software development, it is important that practices are effective from the first planning meetings all the way to production. It is also important to identify areas for development and strive to improve operations in the next cycle.

7.2 Limitations

The research was carried out for one software development unit of one organization, this limits the generalizability of the results. When comparing the results, it must be noted that there would be a similar unit in the comparison. A thematic interview was used for data collection, which gave the interviewees some space to tell about things that interest them. The roles of the interviewees were very central to software development, however, the number of interviewees could have been increased so that different roles could have been included even more comprehensively.

7.3 Further research

There is a lot of research on agile methods. Scaled agile development methods, on the other hand, are a fairly new phenomenon and research is still minimal. Regarding Essence, there are also very few studies. The theoretical part of the research dealt with the utilization of Essence with Scrum. The research topic of this seemed interesting and there are not many studies about it. However, several examples have been found in the literature where organizations used Essence to develop training and software implementation (Park, Jang & Lee, 2018). As a follow-up study for the organization, the benefits of Essence could be reviewed and how its use really helps teams. This style of research has been done before. (Ng, 2015.) The research could be carried out so that Essence tools, such as cards, would be offered to the team for use and observations would be collected based on it.

REFERENCES

Agh, H., & Ramsin, R. (2021). Scrum metaprocess: A process line approach for customizing Scrum. Software quality journal, 29(2), 337-379.

Alasuutari, P. & Alasuutari, P. (2011). Laadullinen tutkimus 2.0. 4. uud. p. Tampere: Vastapaino.

Al-Baik, O. & Miller, J. (2015). The kanban approach, between agility and leanness: A systematic review. Empirical Software Engineering, 20(6), 1861-1897.

Almeida, F., & Espinheira, E. (2022). Adoption of Large-Scale Scrum Practices through the Use of Management 3.0. Informatics (Basel), 9(1), 20.

Alnafessah, A., Gias, A. U., Wang, R., Zhu, L., Casale, G. & Filieri, A. (2021). Quality-Aware DevOps Research: Where Do We Stand? IEEE access, 9, 44476-44489.

Banica, L., Radulescu, M., Rosca, D. & Hagiu, A. (2017). Is DevOps another Project Management Methodology? Informatica Economica, 21(3), 39-51.

Coleman, G. (2016). Agile Software Development. Software Quality Professional, 19(1), 23-29.

Conboy, K., & Carroll, N. (2019). Implementing Large-Scale Agile Frameworks: Challenges and Recommendations. IEEE software, 36(2), 44-50.

Conboy, K., Coyle, S., Wang, X. & Pikkarainen, M. (2011). People over Process: Key Challenges in Agile Development. IEEE Software, 28(4), 48-57.

Dornenburg, E. (2018). The Path to DevOps. IEEE software, 35(5), 71-75.

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

Ebert, C. & Paasivaara, M. (2017). Scaling Agile. IEEE Software, 34(6), pp. 98-103.

Fitzgerald, B. & Stol, K. (2017). Continuous software engineering: A roadmap and agenda. The Journal of systems and software, 123, pp. 176-189.

Forsgren, N. (2018). DevOps delivers. Communications of the ACM, 61(4), 32-33.

Gonçalves, L. (2018). Scrum. Controlling & Management Review, 62(4), 40-42.

Greer, D. & Hamon, Y. (2011). Agile Software Development. Software-Practice & Experience, 41(9), 943-944.

Guest, G., MacQueen, K. M., & Namey, E. E. (2012). Applied thematic analysis. SAGE Publications.

Hyvärinen, M., Nikander, P., Ruusuvuori, J., Aho, A. L., & Granfelt, R. (2017). Tutkimushaastattelun käsikirja. Vastapaino.

Inayat, I., Salim, S. S., Marczak, S., Daneva, M. & Shamshirband, S. (2015). A systematic literature review on agile requirements engineering practices and challenges. Computers in Human Behavior, 51, 915.

Ivar Jacobson International. (2022). Essence Activity Spaces. Retrieved on 3.8.2022 from: https://www.ivarjacobson.com/publications/articles/essence-activity-spaces

Jacobson, I., Ng, P-W., McMahon, PE., Spence, I., Lidman, S. (2012). The essence of software engineering: the SEMAT kernel. Commun ACM.

Jacobson, I., Ng, P-W., McMahon, P. E., & Goedicke, M. (2019). The essentials of modern software engineering: free the practices from the method prisons!

Jacobson, I., Sutherland, J., Kerr, B., & Buhnova, B. (2022). Better Scrum through Essence. Software, practice & experience, 52(6), 1531-1540.

Johanssen, J. O., Kleebaum, A., Paech, B., & Bruegge, B. (2019). Continuous software engineering and its support by usage and decision knowledge: An interview study with practitioners. Journal of software : evolution and process, 31(5), e2169-n/a.

Kalenda, M., Hyna, P. & Rossi, B. (2018). Scaling agile in large organizations: Practices, challenges, and success factors. Journal of Software: Evolution and Process, 30(10), n/a.

Kniberg, H. & Skarin, M. (2010). Kanban and Scrum-making the most of both.

Leite, L., Rocha, C., Kon, F., Milojicic, D., & Meirelles, P. (2020). A Survey of DevOps Concepts and Challenges. ACM computing surveys, 52(6), 1-35.

Lopez-Fernandez, D., Diaz, J., Garcia-Martin, J., Perez, J., & Gonzalez-Prieto, A. (2021). DevOps Team Structures: Characterization and Implications. IEEE transactions on software engineering, 48(10), 1.

Lwakatare, L. E., Kilamo, T., Karvonen, T., Sauvola, T., Heikkilä, V., Itkonen, J., . . . Lassenius, C. (2019). DevOps in practice: A multiple case study of five companies. Information and software technology, 114, 217-230.

Misra, S., Kumar, V., Kumar, U., Fantazy, K. & Akhter, M. (2012). Agile software development practices: Evolution, principles, and criticisms. International Journal of Quality & Reliability Management, 29(9), 972-980.

Ng, P. (2015). Integrating software engineering theory and practice using essence: A case study. Science of computer programming, 101, 66-78.

Park, J. S., Jang, J., & Lee, E. (2018). Theoretical and empirical studies on essence-based adaptive software engineering. Information technology and management, 19(1), 37-49.

Poppendieck, M. & Cusumano, M. (2012). Lean Software Development: A Tutorial. Software, IEEE. 29. 26-32. 10.1109/MS.2012.107.

Poppendieck, M. & Poppendieck, T. (2003). Lean software development: An agile toolkit. Addison Wesley, Boston, Massachusetts, USA.

Poppendieck, M. & Poppendieck, T. (2007). Implementing Lean Software Development: From Concept to Cash. Addison-Wesley, Boston, Massachusetts, USA.

Rajapakse, R. N., Zahedi, M., Babar, M. A., & Shen, H. (2022). Challenges and solutions when adopting DevSecOps: A systematic review. Information and software technology, 141,106700.

Remta, D. & Buchalcevova, A. (2021). Product Owner's Journey to SAFe®—Role Changes in Scaled Agile Framework®. Information, 12(3), 107.

Scaled Agile. (2021) What is SAFe? Retrieved on 10.4.2021 from: https://www.scaledagile.com/enterprise-solutions/what-is-safe/create-a-learning-culture/

Scrum. (2020). What is Scrum? https://www.scrum.org/resources/what-is-scrum

Schwaber, K. & Sutherland, J. (2017). The Scrum Guide. The Definitive Guide to Scrum: The Rules of the Game.

Shahin, M., Ali Babar, M., & Zhu, L. (2017). Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices. IEEE access, 5, 3909-3943.

Sreenivasan, S. & Kothandaraman, K. (2019). Improving processes by aligning Capability Maturity Model Integration and the Scaled Agile Framework®. Global Business and Organizational Excellence, 38(6), 42-51.

State of Agile. (2020). 14th annual State of Agile report. Retrieved on 12.3.2021 from: https://stateofagile.com/#ufh-i-615706098-14th-annual-state-of-agile-report/7027494

Takeuchi, H. & Nonaka, I. (1986). The new product development game. Harvard business review, 64(1), 137-146.

The LeSS Company B.V. (2022). Why LeSS Framework. Retrieved on 20.9.2022 from: https://less.works/

Tuomi, J. & Sarajärvi, A. (2002). Laadullinen tutkimus ja sisällönanalyysi. Helsinki: Tammi.

Tuomi, J., & Sarajärvi, A. (2018). Laadullinen tutkimus ja sisällönanalyysi (Uudistettu laitos.). Kustannusosakeyhtiö Tammi.

Valli, R. & Aarnos, E. (2018). Ikkunoita tutkimusmetodeihin: 1, Metodin valinta ja aineistonkeruu : virikkeitä aloittelevalle tutkijalle (5., uudistettu painos.). Jyväskylä: PS-Kustannus.

Zhu, L., Bass, L., & Champlin-Scharff, G. (2016). DevOps and Its Practices. IEEE software, 33(3), 32-34.

APPENDIX 1

Themes and interview questions:

1. Theme: Background information
   - What is your role/work title/job description in the organization?
   - How much do you have work experience?

2. Theme: Development process
   - How is the development process planned to be realized?
   - What methods are used?
   - How long have the current methods been used?
   - How do the plans (roadmap) guide the work?

3. Theme: Responding to changes
   - How to consider changing requirements?
   - How are changes communicated?
   - What would you improve in the current operating model?

4. Theme: Challenges and strengths
   - What strengths do you see in current practices?
   - What challenges do you see in current practices?

5. Theme: Quality of software
   - How is it considered that the system remains of high quality?
   - What factors do you think enable the quality to remain high?
   - How would you improve the quality?