HAMK
Hämeen ammatti-
korkeakoulu

LEADING MOBILE APP DEVELOPMENT IN A TELECOMMUNICATIONS COMPANY

Master's Thesis

Data Based Management and Smart Networks

Spring, 2024

Harri Määttä

HAMK
Hämeen ammatti-
korkeakoulu

Koulutus	Tietojohtaminen ja älykkäät palvelut	Tiivistelmä
Kampus	Hämeenlinna	
Tekijä	Harri Määttä	Vuosi 2024
Työn nimi	Leading Mobile App Development in a Telecommunications Company	
Ohjaajat	Markus Sihvonen	

TIIVISTELMÄ

Tässä opinnäytetyössä käydään läpi erään suomalaisen teleoperaattorin nykytilanne koskien mobiilisovellusten kehitystyön johtamista, mitä puutteita kehittäjät siinä nykytilanteessa näkevät sekä millaisin toimenpitein nykytilannetta voisi parantaa. Opinnäytetyö keskittyy kehitystyön johtamiseen ja jättää ulkopuolelle esimerkiksi henkilöstöjohtamiseen liittyvät kysymykset.

Opinnäytetyössä sivutaan myös ketteriä ohjelmistokehitysmenetelmiä sekä näihin liittyviä kysymyksiä kuitenkaan menemättä liian syvälle näiden menetelmien yksityiskohtiin. Näitä sivutaan vain sen verran, mikä on paras edesauttamaan lukijan ymmärrystä menetelmistä sekä niihin liittyvistä ohjaus- ja johtamishaasteista.

Opinnäytetyötä tehtäessä haastateltiin kahdeksaa mobiilisovellusten kehittäjää ja analysoitiin puolistruktuoidun, teemahaastattelun vastaukset. Näiden vastausten pohjalta lopulta pohdittiin mahdollisia parannuskohteita hyödyntäen myös samoja ongelmia sivuavia opinnäytetöitä ja näiden pohjalta luotua tapausesimerkkejä.

Lopputuloksena oli esimerkkejä teknisen velan käsittelyyn ohjelmistokehitysprojekteissa sekä niin sanottujen siilojen tuomat ongelmat ja kuinka niistä poispääsyä voisi lähteä ratkomaan.

Avainsanat	ohjelmistokehitys, ketterät menetelmät, ohjelmistokehittäjät, yhteisöviestintä, tiedonkulku
Sivut	51 sivua

Name of Degree Programme		**Abstract**
Data Based Management and Smart Networks		
Campus	Hämeenlinna	
Author	Harri Määttä	Year 2024
Subject	Leading Mobile App Development in a Telecommunications Company	
Supervisor	Markus Sihvonen	

ABSTRACT

This thesis analyses the process of leading mobile app development in the telecommunications company Elisa Corporation and how the current state of the process can be improved.

This thesis focuses on the actual process of leading mobile app development, excluding topics such as human resources issues.

This thesis also includes a few words about modern agile software development frameworks and raises some questions about what these methods bring up. However, this thesis does not delve too deeply into agile development methods. Going through the basics of these should help the reader to understand the nature of agile development and the challenges of leading this kind of development work. Understanding agile development also aids in understanding the roles and responsibilities in a modern software development team.

The outcome of this thesis focuses on handling technical debt, how silos in a company can cause issues, and how improving communication can help to break down such silos. At the end of the thesis, there are some open questions and ideas about how the process of leading mobile app development could be improved.

Keywords software development, agile methods, developers, company communication, sharing knowledge.

Pages 51 pages

Contents

Images

Attachments

1 Introduction

This section introduces the background to the research, the main questions being researched, details of the methods and some criticism of the sources.

The background for this thesis is to analyse and find improvements for leading mobile app development at Elisa Corporation. Mobile app developers develop multiple apps in various teams. The structures and number of developers in each team vary, and so do the development tools, programming languages and technologies that they use. Developing mobile apps has some features that are different from general programming projects.

1.1 Research questions

The primary enquiries and objectives of this thesis aim to address the following key points:

- What is the current state of leadership in mobile app development at Elisa Corporation?
- What are the primary challenges in management and leadership, as perceived by developers within the context of mobile app development?
- How can the issues of leadership in the mobile app development area that have emerged be fixed?

1.2 Research background

During my long career working in software development companies, I have noticed that some issues remain valid for long periods, and they either seem to be hard to fix, or fixing them never rises to a top priority in the team's next tasks. During these years, I have developed a theory that with improved leadership, these common and long-lasting issues could be solved, or at least fixed for the better.

In this thesis, I use research methods to test my theory that the telecommunications company Elisa Corporation could create better mobile apps for its customers, emply synchronised updates for common components, and use its resources more economically. Also, with more structured leadership, the company could create extra value for customers that the current fragmented teams and ways of working cannot create.

To find out how realistic the ideas mentioned above are, I engaged in interview sessions with developers, and I also researched case studies about the most common pain points to discover theories about development processes.

This thesis includes an explanation of why mobile app development is different to regular development, and I have also included some theory sections about agile development frameworks to help the reader understand how projects and development are led at the service level.

For security reasons, mainly to avoid phishing attempts, this thesis does not use accurate, current names for the company’s non-public services, teams and business units. Also, the names of colleagues, product owners, etc. are hidden, even when they were mentioned in the interviews. The developers that were interviewed are aware of this.

For the business unit that handles most software development for the consumer business area in the company, including the online store, mobile apps and UX design, this thesis uses the fictional name “Group of Developers” (GoD). The department that handles the company’s corporate business is called “CB”, and the department for retail customers is called “RC”.

In the current situation, mobile app development is spread around all areas of the company. Even though most developers are members of several teams, there are still a few professionals who are located far away from colleagues with the same profession – both in physical location and in the organisational stucture .

This kind of fragmentation is nowdays referred to as “silos”, even though there are no physical silos as in the infrastucture; the term relates to communication issues and various objectives in different teams. Silos are not issues only in software development. Study has found similar issues, as an example, between of marketing and internal communication departments (Neill & Jiang, 2017). According to the website Grammarly, department silos can effect to company's health negative way (Grammarly Inc., 2021). Instead of "company's health", in this thesis, I try focus to mobile application development and developers.

Each business service team focuses on developing their own main product. There is one person whose main task is to maintain mobile app stores and share common knownledge about new tools or practices who is not part of the same team as the mobile app developers. The writer of this thesis is that person.

1.2.1 Elisa Corporation

Elisa Corporation, the company that is the subject of this research is a corporation offering telecommunications and digital services. Elisa is over 140 years old and during these years it has grown an amout of customers near to three million. Customers are in Finland, Estonia and internationally (Elisa Corporation, 2023, p. A sustainable future through digitalisation.). Elisa Corporation has multiple apps available on public mobile app stores, such as Google Play for Android devices and Apple's App Store.

Elisa's mission is "a sustainable future through digitalisation", its vision is, "we are the global benchmark for generating value in communication and digital services. Our constant pursuit of excellence and innovation makes us better every day", and its strategy is, "we innovate digital services for customers in our own telecom footprint area and internationally with network ownership-independent services. We engage people in building excellence." (Elisa Corporation, 2023, p. Mission and values) From these corporate values, the importance of digital services and improving sustainable future and helping people to thrive is notable.

Due to the nature of this thesis, it is not reasonable to examine the telecommunications side of Elisa's business, even though it is a very interesting area in general. However, to help the reader of this thesis to better understand Elisa's digital services, the section below introduces some of the company's services that have, or might have, something to do with mobile app development. Also, interviewed developers migth be part of teams that develop mentioned services below or then from other teams.

Elisa Viihde is a consumer entertainment service with a selection of streaming movies and TV series. It also provides the possibility to watch and record TV programmes. Elisa Viihde also produces its own original series and films (Elisa Corporation, 2023, p. Services for private customers in Finland).

Elisa Kirja is Elisa's eBook and audiobook service in Finland, offering the widest selection of eBooks in the Finnish language. Customers can rent or buy eBooks and audiobooks individually, or read or listen to them with a monthly subscription (Elisa Corporation, 2023, p. Services for private customers in Finland).

In Estonia, Elisa also has an Android-based TV service called Elisa Elamus that works on smart devices. It includes the possibility to watch live TV and to rent videos and other entertainment from Google, Viaplay and Amazon, as well as the Elisa Hub service. The Elisa Elamus set-top box also lets customers download apps and games from the Play Store (Elisa Corporation, 2023, p. Services for private customers in Estonia).

1.3 Research setting

This thesis was made using qualitative research method. Although a cohort of mobile app developers exists at Elisa, I maintain that employing qualitative research remains the optimal approach in this context. The available sample size would not suffice for conducting a robust quantitative data analysis while ensuring high quality.

Eight developers were interviewed using themed interviews. The themed interviews offered some space for additional questions, if needed to clarify questions or answers. For those who were still studying at universities or universities of applied sciences while simultaneously working for the company, I had several extra questions about writing their thesis and joining the company as a newcomer.

At the beginning of the questionnaire, I included four quotes about leadership and management. These quotes simplify the difference between managing and leading, helping the developers to think about everyday management. The company's mission was also included mission in case some of the interviewees were unable to remember it correctly.

One of the questions was about the company’s mission and how it is implemented in mobile app development.

The interviews were mostly conducted remotely, using the videoconferencing and collaboration software Microsoft Teams. I shared my notes while writing so that the interviews could be fixed if I missed or misspelled something. One interview was conducted face-to-face at the office, and in this case, the developer being interviewed was able to see my notes while I was writing them. The interviews were mostly conducted in Finnish, but two of them were carried out in English. When the interviews were done, in the case of the Finnish developers, I translated the answers and sent them to the interviewees for checking. If any issues were found, they were fixed. When the interview sessions were conducted in English, then the translation process was not needed, and I could write notes in English from the beginning.

When reviewing literature sources, it was notable that some terms, like “technical debt” were surprisingly rare in academic sources and those rare sources were strongly focusing to software development. Instead of academic research, commercial sources were using these terms and had a lot of content to provide. Naturally commercially available data is not academically reviewed, and this has been taken into account when using these sources.

No AI tools were used in writing this thesis, with the exception of Word’s grammar checker.

1.4 Research method

For this thesis eight developers were interviewed, and answers were analysed using qualitative research method. All the interviewees are from GoD, and they all had at least some experience working in the company. Several of the interviewees are still studying while working under part-time contracts, and the interviews included some extra questions for these interviewees. These questions were related to doing thesis work while being employed at the same time, as well as how their first steps in working life had gone. The interviews were semi-structured, offering some room for additions, if there was a need for extra questions.

The theory behind this thesis is that if GoD were to expand the role of leading the company's mobile app development, its app developers could would be more productive, creating better apps for customers. Also, with improved communication it would reduce the threshold for developers to change teams, increasing knowledge sharing.

1.4.1 Themed interviews

The questionnaire can be found as Attachment 1. Leading and leadership is a complicated topic and there are thick books about the topic, but I tried to underline the main points by reading a few quotes at the beginning of the interviews. Also, in the Finnish language, there is no clear distinction between *management* and *leadership*, as both words translate to the same Finnish word: *johtaminen*.

One question was related to the company's vision. In case any of the interviewees had any trouble remembering the vision, it was added to the questionnaire template and was visible to all the interviewees.

When the interview sessions were done, I edited and tidied up the text and, for interviews in Finnish, translated the answers into English. When this was done, I sent my notes to the interviewees so that they could make sure that I hadn't missed anything and that my translation process had not changed the point or nature of their answers.

1.5 Source criticism

When doing this research, it is important to note the characteristics of software development, mobile app development and creating a culture of evolution in companies. In addition, when interviewing developers, I tried to avoid manipulating the interviewees by having a strictly planned and structured questionnaire. Even so, I could not exclude other possible sources of bias or manipulation that could influence developers in background, such as a specific hot topic in their teams or ideas that the interviewees could have shared between themselves.

Besides, when researching the theorical topics from the technical and business perspectives, it was notable what a small amount of scientific research has been conducted in this area, in particular, when terminology that is similar to that used in business life, such as *silos*. This had the result that many commercial sources and posts by companies online were used as sources. This has naturally been taken into account when analysing reliability of sources. Personal blog posts were avoided, and original sources have been preferred where possible, such as scrum.org for issues related to Scrum.

With the correct keywords, it is quite common that posts on social media services will show up in the results. Especially when using keywords like *leading*, *software* and *development*, social media posts were quite often among the top results when searching around these topics online. No social media posts have been used as sources for this thesis.

2 Theory – technical

2.1 Uniqueness of mobile app development

One big point that should be raised in this thesis is the difference between “normal” development and mobile app development. Mobile app development variates from other kind of development (Alrabaiah & Medina-Medina, 2021). If we simplify “normal” development to only apply to web or backend development, the main difference from mobile app development is that in web and backend development, developers have almost all the power to do almost anything they can. In the mobile app development, its ecosystem, the app stores have owners who play a very strict role as gatekeepers. The image below illustrates these differences well.

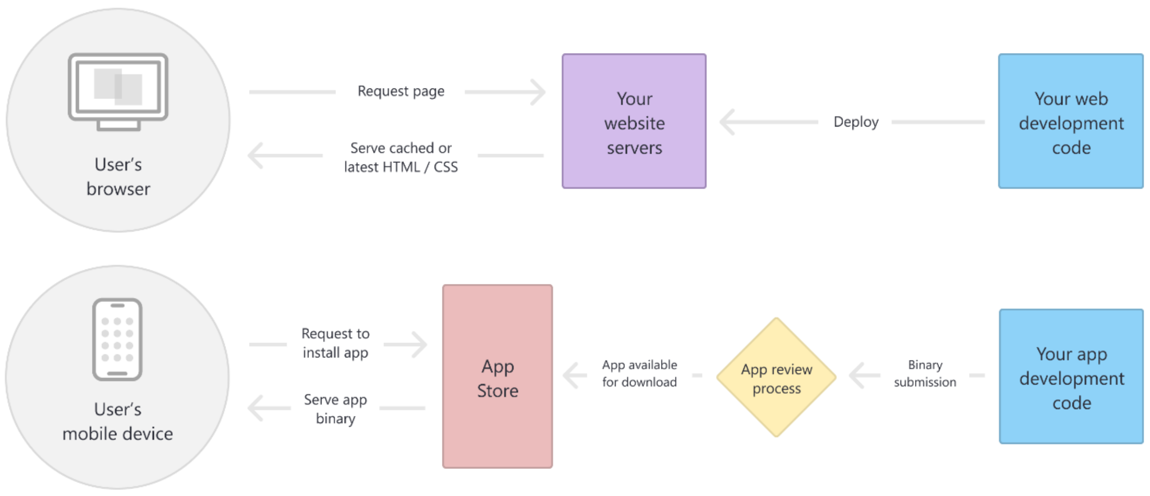

Figure 1. Deployment processes (Barrera, 2020)

Figure 1 shows the differences in ownership. In the case of a website, there is *your* web development code and *your* website servers (that users connect to). On the mobile app development side, the code is "yours", but from that point, there are some components owned by others. In the case of an iOS app, Apple owns the app review process and the App Store. On the Android side, Google is the owner. It is not necessary to examine software development more closely here, but Figure 1 shows very clearly that in mobile app processes, ownership is more complicated, and power is given to third parties in review processes. This means that they can modify your app, request that the developers make changes, reject updates, etc. Feedback from app stores – especially if they discover an issue – needs to be transferred to the developers or business teams. Every now and then, there is a need to make changes to apps or the related data to get them updated in app stores and available to customers.

2.2 Development models

Apps can be developed in various ways. The next sections present some simplified differences between in-house and outsourced development and white-label products.

Development work that is carried out by a company's own employees on company premises can be called *in-house development*. In this case, success or failure are in the company's hands. Creating and maintaining an in-house development team is a process. (STX Next, n.d.)

In-house development naturally has positive and negative aspects. The positive effects are that the company has full control over its employees. They are part of the company's culture. Also, communication is easier when all the resources are from the same company. On the other hand, the demanding recruitment process provides teams with hand-picked team members (STX Next, n.d.). Developing products or services "in-house" gives more power to design those to meet specific needs of organisation, business unit or team (Jackson & Brannon, 2018).

One negative side of in-house development is that it can result in increased costs. All resources – people, premises, etc. – result in costs. Also, creating a development team via recruitment takes time. Overall, limited resources can create other limitations.

Outsourced development is a practice where a company outsources software development to a team outside its organisation, with the goal of saving time and money. The benefits of outsourced development are increased efficiency. It makes it possible to set up a team and start working without a complicated recruitment process. Also, markets provide a large pool of experts for each specific area. Naturally, there are some risks when outsourcing development, such as potential communication problems, security issues and lack of control (Walburg, 2021). Nevertheless, research indicates that the stability of outsourced teams positively affects project management and customer satisfaction. (Narayanan, Balasubramanian, & Swaminathan, 2010).

A *white-label product* is a product that is purchased by a company from a service provider and rebranded as its own. White-label products can be labelled with the buyer's own brand icons or be unbranded. White labelling can be optimal approach for delivering a software solution without the need to develop it entirely anew is. White-label projects can be economical with time and financial resources. Also, eliminating the need for hired developers can be a huge cost saving (Paddle, n.d.). In other words, "white labelling involves a company buying software, customizing it, and presenting it to other people as their own" (Marin, 2022).

White-label products have several notable benefits. Naturally, cost savings are one, as it is not necessary to hire developers or buy outsourced developers. Also, time can be saved if the development part can be skipped completely and a finished product with a rebranded user interface can be used instead. The quality of the product should be known, as testing and improving is part of software development. This also covers security issues. White-label products also have benefits of scale if the needs of the markets are known and already tested by other similar products. Branding is easy if a white-label product is high quality, sells well and results in a positive user experience at every. Rebranding this kind of successful product with a company’s own logos creates positive value (Marin, 2022).

2.3 Technical debt and legacy code

Technical debt can be understood as work that has yet to be done and will affect the product, now or in the future. Existing problems will get worse over time. Even technical debt in software development that is consciously taken on can be a known risk. Knowing the risk can provide tools to perform development project, even that there can be known issues, as an example, with performance (Lutkevich, 2023). So, technical debt can be planned or unplanned. When technical debt accumulates, then “payback time” can be seen when the teams become less capable of developing new features or maintaining current ones. Also, in some cases, technical debt can affect other teams as well, especially in complex projects. The causes of technical debt vary, stemming from sources such as unclear project requirements, bad code, no training, poor tool implementation, lack of documentation and information silos (Lutkevich, 2023).

Team’s strategies to manage technical debt can variate. Study says that, in this case, some teams had strategy for debt management was only to fix debt when it escalated for too big number of issues for development. Conversely, certain teams under examination implemented a systematic approach to manage debt throughout the development. (Yli-Huumo, Maglyas, & Smolander, 2016).

The ability to reduce the technical debt naturally varies based on the origin of the debt, but the article *7 Simple but Effective Steps to Reduce Technical Debt* lists seven simple ways that it can be reduced (Schults, 2022) These steps are:

- Embrace automated testing.
- Adopt code review practices.
- Adopt pair/mob programming when it makes sense.
- Create a culture of constant refactoring.
- Track and improve the right metrics.
- Understand and treat the root cause (at the organisational level)
- Listen to your engineers.

Legacy code can be understood as a code that has given from the past. It has already reached or passed the end of its support cycle. The term can also refer to code, libraries or tools that are otherwise too complex, outdated, irrelevant or complex to maintain compared to modern tools (Gorbachenko, n.d.). Even though mobile app development platforms are quite strict and moderated by the platform providers, it may be that legacy code act as block for developers to move on newer techs and frameworks (Gorbachenko, n.d.). Also, code that is valuable and developer is too afraid to change, can be seen as a legacy cod (Carlo, n.d.).

To add new features, there might need to update the old codebase and related parts like libraries and configurations for compatibility. Legacy code is considered less stable and more fragile. Legacy code (base) might also create risks for security vulnerabilities. *Refactoring* can be an option for eliminating legacy code. Refactoring means modifying or rewriting the code and keeping its functionality as it. Idea of this is to upgrade and improve, as an example, its readability. (Gromenko, 2022).

2.4 Guides for mobile developers

Mobile app development has some features that are somewat different than, for example, web development. (See section 2.1 for more details.) The main mobile app platforms – iOS from Apple and Android from Google – have some guidelines for developers.

Apple is famous for its strict rules in app development. Apple's Apple Developer Documentation (Apple Inc, 2023, p. Apple Developer Documentation) is probably the best starting point for every app developer. Apple also has a lot of guides and examples for UI design, like the Human Interface Guidelines (Apple Inc, 2023, p. Human Interface

Guidelines), which need to be followed if developers plan to get their app accepted into the Apple App Store.

Also, based on my working history, one of the most-read guidelines is the App Store Review Guidelines (Apple Inc, 2023, p. App Store Review Guidelines). Apple has a complicated review process (Apple Inc., 2023, p. App Review) and, especially with new apps, it is more the rule than the exception that your app will be rejected before it is accepted for publication to a public audience.

Naturally, Google also has good, if complicated, guidelines for mobile app developers. The Android for Developers website is the official place to start. It also contains references for the Kotlin programming language as well as guides for how to create apps for cars or wearable devices. (Google LLC, 2023, p. Android for Developers).

One thing that is common to these non-native tools is that they do not have app stores. To release apps to customers, you must use the official app stores from Apple or Google. However, with these tools, developers will be creating similar app packages than with a native IDE.

For Flutter, the official website is at https://flutter.dev/. As can be seen in the architectural documentation (Google LLC, 2023, p. Flutter architectural overview | Flutter), Flutter has built some layers and libraries to support an idea that same code can be used in multiple environments. This means that a single app should be usable on both Android and Apple devices.

Kotlin Multiplatform technology aims to streamline the development of cross-platform projects, and it should reduce the amount of resources to develop same code for different platforms (Kotlin Foundation, 2023, p. Kotlin Multiplatform | Kotlin Documentation). With Kotlin Multiplatform, it is also possible to build full-stack web apps (Kotlin Foundation, 2023, pp. Build a full-stack web app with Kotlin Multiplatform | Kotlin Documentation).

React Native is one of the most common non-native tools to build webpages and mobile apps. React Native allows creating versions of components specific to each platform, so one set of code works across different platforms. (Meta Platforms, Inc., 2023).

In addition of official documentation there can be internal wiki pages. Study says that wiki pages has helped developers to share knowledge, improved using of agile processes and increased motivation to investigate autonomously. (García, Amescua, & Sánchez, 2011)

2.5 Mobile app technologies

The topic of this thesis is not mobile app technologies, but to understand the answers in the themed interviews, it is good to take something from these topics.

Developing mobile apps can be shared into two categories: native development and cross-platform development. Making this choose effect to development processes, used tools and user experiment. (Schmitt, 2023)

Native mobile apps are target either for Android or iOS. Depending on which operating system the project is targeted, there can be used recommended programming languages, like Kotlin or Swift.

Android apps are built with either Java or Kotlin. Initially, Java was the primary language for Android development. However, in 2017, Google introduced support for Kotlin. Kotlin allows for both object-oriented and functional programming, while Java is restricted to object-oriented programming. Additionally, Apple unveiled Swift during their Worldwide Developers Conference in 2014. (Schmitt, 2023).

There are many blog posts about the good and bad sides of native and cross-platform development. One blog post on Net Solutions (Singh, 2023) lists the following as benefits of native app development and benefits were like rapid execution speed and improved security. Negative things were like slower development speed and need for specific area experts. Benefits of cross-platform development are better product maintenance and possibility to reuse codes. Negative side for cross-platform development were like issues with integration to platforms. (Singh, 2023).

Flutter is a framework by Google, it used a programming language called Dart, and it is open source. It is designed that you can use single source code to create Android and iOS apps.

Some developers might appreciate that it is type safe language. The programming language Dart is designed to help developer by ensuring that a variable's value matches the variable's static type. (Google LLC, 2023)

One alternative solution for Flutter is React Native. It promises that when developers have created their product, the end users get expected experiment by using native components given by the platforms, Android and iOS. (Meta Platforms, Inc., 2023)

3 Theory – business and corporation

3.1 Stakeholders

A stakeholder refers to an individual, group, or organization that has direct or indirect involvement in a project and can influence or be influenced by its outcome. They can provide data on software or product needs and suggest ideas. Stakeholders can be either i*nternal stakeholders* or *external stakeholders*. Internal stakeholders can be roles and persons like developers, product owners and managers. External stakeholders can be e.g. a customer. (Cherednichenko, 2021).

Developers are stakeholders themselves, but they also communicate with other stakeholders, like managers, product owners, customer care, etc. After all, stakeholders are suggesting ideas for features or problems. (Concepta Technologies LLC, 2022). Studies says that modern agile development frameworks, like Scrum, can increase confidence of stakeholders (Udvaros, Forman, & Avornicului, 2023).

3.2 Silos

In business, *silos* refer to a block on sharing resources between individuals in the corporation. With silos ideas and/or information is not optimally shared with other departments or employees, like mobile app developers in our case, or managers, leaders, or product owners. Blocks or delays in sharing of information can slow down development and moderate scaling up. (Indeed, 2022, p. What Are Silos in Business? Causes for Organizational Silos).

The reasons for organisational silos can be summarised in four points: leadership shortcomings, business structure, communication challenges, and unclear roles or expectations. Leadership flaws can impact to every team and individual. Leaders are in responsibility to create an atmosphere for developers to act like company values expects. (Indeed, 2022, p. How To Break Down Silos Within an Organization).

In terms of business structure, if a team or organisation is built in a non-supportive way, it can create silos. Especially silos can appear when employees believe they're on their own and there is lack of supporting for teamwork. (Indeed, 2022, p. How To Break Down Silos Within an Organization).

Difficulties in communication may lead to a silo mentality. And this might be done by isolating departments, teams, or developers and making them feel ignored. A successful method to handle this is by empower open communication among team members and offering tools to facilitate dialogue. (Indeed, 2022, p. How To Break Down Silos Within an Organization). A study about internal online communities revealed the emergence of dense subgroups with similar locations emerged and that it decreased engagement of employees of sharing knowledge across multiple knowledge domains (Hwang & Krackhardt, 2020). Lack of knowledge about roles and responsibilities might lead to misunderstanding of expectations or duplicated tasks (Indeed, 2022, p. How To Break Down Silos Within an Organization).

In eliminating silos, it is important to limit or remove the sections between developers or teams in an organisation. If departments have a silo mentality, there is a risk that they will not share information or resources with anyone outside their group. Eliminating silos improves collaboration. This supports decision-making, increases productivity, and fosters innovation (Indeed, 2022, p. How To Break Down Silos Within an Organization).

Destructing silos can be done by e.g. establishing the organisation's vision, providing incentives, promoting cross-departmental collaboration, implementing team-building practises and meetups, and break communication barriers (Indeed, 2022, p. How To Break Down Silos Within an Organization).

3.3 Projects vs agile development

Traditional project management can be seen as a linear. All the steps appears when previous is done and this creates predictability. The main idea is a planned project beforehand without any change for unexpected development requests. In traditional project management is assumed that time and cost can variate and that the requirements should be stable and known. (Narasimman, 2023).

Projects such as software development projects can have multiple issues. These can include frequent technology updates, upgrades, or other changes. Also, immature technology, non-agile development practices, and human resource changes. Software development projects might have specific issues, like navigating stakeholder ideas and that randomness, and communicating with end users, developers, and others clearly and understandable (Smartsheet Inc, 2023). On the other hand, studies says that social and political skills moderate the effects of complexity on software-projects performance (Zaman, Jabbar, Nawaz, & Abbas, 2019).

Smartsheet lists four different kind of software projects on their site. These can be new software tools created by software vendors or startups. Then these can be like updates or patches of existing software. Also, there can be mobile apps or internal support software solutions. (Smartsheet Inc, 2023)

Project management has five phases. These five phases are explained in detail in the following paragraphs. Each stage of the project life cycle has a distinct focus (Donato, 2023).

In the phase ‘initiation’, the goal is to check that the project is in line with requirements, and success criteria of the project is commonly understood by team members. The project goal will be the outcome of this phase. In this phase, the entire project team defines the project idea. (Donato, 2023).

The planning phase is dedicated to strategizing and organizing every task required for the team to address the scope, attain the deliverables, and accomplish the overarching objective. Team members also plans specific requirements, tasks and deadlines. The outcome is a collection of documents. (Donato, 2023).

While the execution phases the team will run tasks that are in project plan. In this stage, the project manager oversees all aspects of project delivery to maintain its trajectory, while the team concentrates on accomplishing the established objectives from earlier phases. (Donato, 2023).

In the monitoring and control phase, the focus is on tracking the performance and progress of the project. In the closing phase, the idea is to reach the goal, release resources and complete any related tasks. (Donato, 2023).

3.4 Software development life cycle

The software development life cycle (commonly referred to as SDLC) is a method through which development teams design and construct software, including mobile applications. The concept involves breaking down development into tasks that can be assigned, accomplished, and assessed. The main point of SDLC is to provide some version of the app at the any point or the development process. (Amazon Web Services, Inc. or its affiliates, 2023).

Amazon lists four benefits of SDLC. These benefits are like better visibility of the process for everyone, improved estimation and planning, better risk management and non-stop software delivery for customers to get better satisfaction. (Amazon Web Services, Inc. or its affiliates, 2023)

Depending on the source that is being used, SDLC is divided into either five or six steps. Amazon has split the cycle into six phases (Amazon Web Services, Inc. or its affiliates, 2023).

In phase called “plan” there should be done tasks like cost-benefit analysis, creating time window, resource estimation and allocation. Detailed plan should be written to estimate things like costs etc. When these are written, then those will be analysed by developers. This helps them to identify the best solutions to create the software and they can forward into *design* phase. The development team finds out the best way to create the new software. In the *implement* phase, the developers write the code, develop the app itself. In the *test* phase, the solution is tested for *bugs*. In the *deploy* phase, the idea is to share the developed app with customers. In mobile app development, this can mean that the app is uploaded to mobile app stores and shared with customers, or it might involve a smaller distribution to closed test groups. (Amazon Web Services, Inc. or its affiliates, 2023).

In the *maintain* phase, the developers keep the project running, updated and fix bugs. (Amazon Web Services, Inc. or its affiliates, 2023). However, there is some new concepts of next-generation variates of software development life cycle. One of those is open agile software development life cycle (OASDLC) and there are already studies how it can improve life cycle management (Misra & Singh, 2015).

3.4.1 Agile development

The point of *agile software development* is the ability to react to changes. Also it includes an idea to be able to deliver and make changes at any point when needed. (Davis, 2022). There is also the *Agile Manifesto* and its 12 principles, which are notable quideline for agile development. There is underlined an idea about constant changes and deliver value during development process. Also, it says politics of business and developers working together and how to motivate developers. (Beck, et al., 2001)

Due the topic of this thesis, principle number 4 is notable: “Business people and developers must work together daily throughout the project”. This underlines the importance of good communication. Communication is a critical component of any project’s or team’s success (ProductPlan, 2023). ProductPlan simplfies how this looks in practice into two points. First is that development team should include also developers, but also designers and business people. Second is a praise for daily meetings to keep up active communication. (ProductPlan, 2023).

Daily meetings and other agile development practices are discussed in the section “Theory – business and corporation”.

Principle number 5 is also notable. The main point here is about trust and autonomy and strengthening individuals and teams. It is also crucial that responsibilities are clearly defined before any project begins (ProductPlan, 2023).

Literature study says that mobile app development models are usually focusing strongly to requirements and design approaches. Also used model variates based on variables like size of the team. (Jabangwe, Edison, & Duc, 2018)

3.4.2 Scrum

Scrum is one way of handling agile development. Scrum is a framework for development work, not an exact process. This framework helps individuals and bigger groups, like teams, generate value. Despite employing adaptive solutions for complex issues (Scrum.org, 2022). The Scrum Guide in the address https://scrumguides.org/scrum-guide.html goes through very thoroughly everything that is documented about Scrum.

The main point about Scrum in this thesis is to observe one idea: that in this framework, planning work, implementation and reviews are carried out regularly. Also, understanding the basics of Scrum will help the reader of the thisis to understand how processes can be carried out in software development teams and how decision-making and hierarchies of decision-making power are linked to each other.

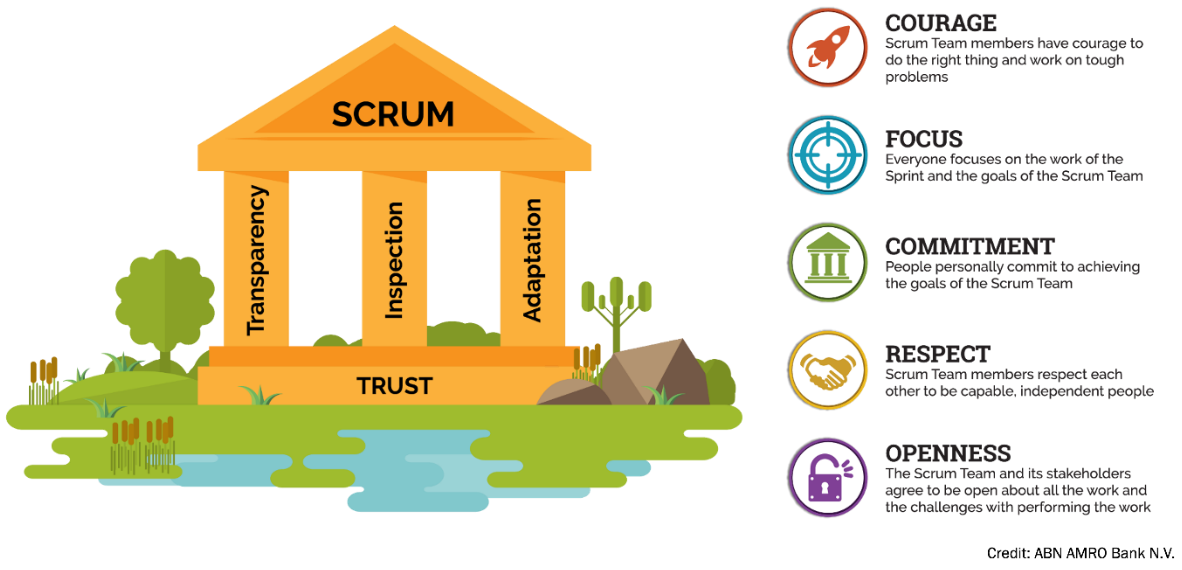
SCRUM
Transparency
Inspection
Adaptation
TRUST
COURAGE
Scrum Team members have courage to do the right thing and work on tough problems
FOCUS
Everyone focuses on the work of the Sprint and the goals of the Scrum Team
COMMITMENT
People personally commit to achieving the goals of the Scrum Team
RESPECT
Scrum Team members respect each other to be capable, independent people
OPENNESS
The Scrum Team and its stakeholders agree to be open about all the work and the challenges with performing the work
Credit: ABN AMRO Bank N.V.

Figure 2. Scrum values (Scrum.org, 2022)

Figure 2 visualises the three pillars which Scrum is based: *Transparency*, *Inspection* and *Adaptation*. These values are based on Trust, and at the top of all of this is Scrum itself. On the right side of Figure 2 are the values of Scrum: *Courage*, *Focus*, *Commitment*, *Respect* and *Openness*.

The pillars of Scrum support the concept of working iteratively. They include the idea of working empirically, with small experiments, learning what has been done and adapting based on what is done and how it can be improved. The Scrum pillars are built on a platform of Trust. Lack of trust can create tension and bottlenecks in the way of getting work done. (Scrum.org, 2022)

The Scrum value of *Courage* means that team has courage to do what is needed – the boldness to do the correct things and work through faced issues (ScrumGuides.org, 2022). The Scrum Team should possess the courage to maintain honesty, openness, and transparency, both internally and with stakeholders. This includes having the courage to ask for help, to try new things, and to be able to engage in civilised dialogue with others (McAbee, 2022).

The value *Focus* underlines that Scrum Teams must focus their work on Sprints to make the best progress toward their goals (ScrumGuides.org, 2022). Participating in daily meetings helps developers to focus on their main tasks (McAbee, 2022). Handling amount of tasks for each or importance of each task can be done by the Scrum Master (McAbee, 2022).

It is required that developers trust each others in their tasks and believe that they are delivering to the best and to be ability to work together to achieve a teams shared goal. This can only happen when team members has focused to the team and the project (McAbee, 2022).

Scrum Team members – developers – should respect each other. This means that they see each other as capable, independent people, and that they are respected in the same way by the people they work with, like stakeholders (ScrumGuides.org, 2022). This also means recognising that no one person's contribution is more valuable than that of others (McAbee, 2022).

Openness, like transparency, comes about through continous communication. For developers, this includes (but is not limited to) being open about their resent challenges. Daily Scrum meetings focus on this, making them a good type of meeting to raise issues or impediments (McAbee, 2022).

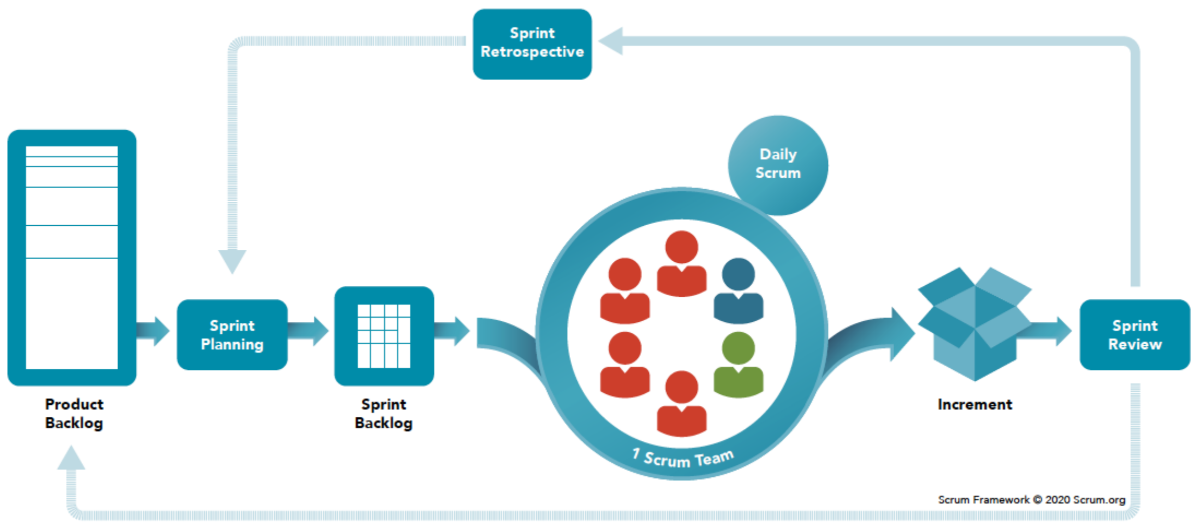

Figure 3. Scrum framework (Scrum.org, 2022)

Figure 3 displays the processes and the items *Product Backlog*, *Sprint Planning*, *Sprint Backlog*, *Daily Scrum*, *Sprint Review*, *Increment*, *Sprint Retrospective*. In the middle is also a *Scrum Team*.

Small group of developers is called to the *Scrum Team* and it contains a one Scrum Master, one Product Owner and one or more Developers. The Scrum Team should have all the knowledge needed to create value (or releasable software) in each Sprint. Teams are self-managing, meaning that they have the power and responsibility to decide who does what, when and how. The optimal size of the team is a maximum of 10 members. Larger teams might result in an increased risk of issues in communication and a lack of productivity (Scrum.org, 2022).

The *Product Backlog* includes items that need to be done to improve the product. It is a simple, single-column list that is undertaken by the Scrum Team. Every item in the Product Backlog can be done by the Scrum Team, and every item should be done in one Sprint. In refinement sessions, the idea is to break down (too) big Product Backlog items into smaller ones to increase transparency and make the development more precise. *Sprint Planning* is an event where the work is planned for the next Sprint. The resulting plan is created by the whole Scrum Team, who can also invite other people to provide assistance and to discuss Product Backlog items. Here, the Product Owner proposes the next set of tasks to increase the value of the product. Based on that, the whole Scrum Team collaborates and plans the next Sprint Goal to specify why the Sprint is valuble for stakeholders. This event also includes estimating how much work is required in the next Sprint. Naturally, the definition of “done” can (and should) be specified in this meeting. The Sprint Backlog includes those items that are planned to be executed during that Sprint. This plan is created by developers for developers. The Sprint Backlog should be visible to everyone, and it should be actively updated to offer the latest information (Scrum.org, 2022).

In *Daily Scrum* meetings, the idea is to quickly check the Scrum Team’s progress to the Sprint Goal. If needed, then the Sprint Backlog can be adjusted. The Daily Scrum is a meeting for developers and should take a max of 15 minutes. Keeping Daily Scrum meetings literally every day should improve communication, raise up blocking issues, empowering for quick decision-making if needed and cut the need for other meetings. In the *Sprint Review*, the idea is to go through what items from the Product Backlog the Scrum Team has done in the Sprint. Here, outcomes are inspected and the next actions are planned. The Scrum Team presents their achievements to the stakeholders. This is also a good place to go through that what has gone well in the Sprint that is ending and what could be done better (Scrum.org, 2022).

An *Increment* is the work that the Scrum Team has done so far. This might be an update to some software, for example. An Increment is not an idea or plan or theoretical presentation that how to reach the Product Goal. Increments should be usable by stakeholders, so fancy words are not counted as an increment. In a *Sprint Retrospective*, the main idea is to increase quality and effectiveness. The idea is to think about questions like “What went well?” (in the Sprint), “What could be improved?”, and “What shall we commit to improving?” (in the next Sprint) (Scrum.org, 2022).

If teams are following Scrum, they also use titles that are relevant for Scrum. This means that there is team called the Development Team, and personal roles like Product Owner, Developer and Scrum Master (ScrumGuides.org, 2022). It is natural that other business areas and teams that are not famliarq with development practices will use other titles.

This thesis is strongly focusing to software development, which means that terms and related items of Scrum is following software teams. However, Scrum can be used also in non-software industry. There is study where was introducted so called Scrum-Based New Product Development (SBNPD) and found that it provides some improvement over other approaches. (Cano, García-Camús, Garzás , & Moguerza, 2021)

3.4.3 "Waterfall" development

Compared to modern agile software development methods, one old method, originally copied from other industries, is "Waterfall" development. The name "Waterfall" comes from the, where each step cascades down to the next step.

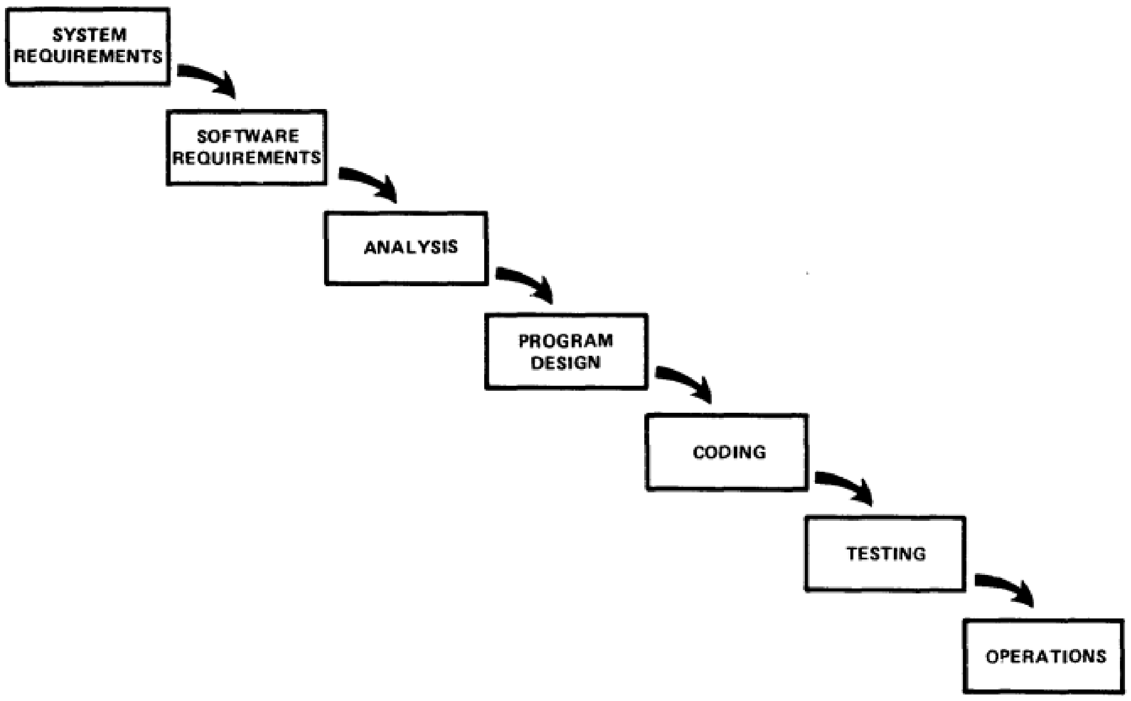

Figure 4. Waterfall method (Royce, 1970)

The idea here is not to examine development frameworks deeply, but Figure 4 describes well how each step follows the previous one. This means that if a development team uses this method, there would be a very narrow time window for tutoring, or there should be tutoring at every step. This kind of strict waterfall method is no longer used in modern app development teams, so there is no reason to go deeply into this.

Nevertheless, the study that compared software development modes, raises a point that a new software development model is needed. This new model should be beneficial for clients and simple for developers. It should help development teams minimizing risk within decided budget and quality. (Alam, Sarwar, & Noreen, 2022)

3.5 Roles and responsibilities in agile development models

The most relevant roles in this thesis from the business areas are Business Manager and Developers. To simplify, I have dealt with Business Managers by putting them in the role of Product Owners. Communication with Developers has been done based on the assumption that their role is similar to that of Scrum's Developers. The Scrum Guide says that Developers are always accountable for creating the Sprint Backlog, refine a Definition of Done, adapting plan to reach the Spring Goal and appreaciate other professionals (ScrumGuides.org, 2022).

Based on the Scrum Guide, the Product Owner is in response to maximise the value from Scrum Team. How this is done may vary widely. (ScrumGuides.org, 2022). Besides that, the Product Owner is accountable also about backlog management. These tasks can be developing and explicitly communicating the what the "finished" product is, known as a "product goal" (ScrumGuides.org, 2022).

Product Owners should create and communicate the next tasks and backlog items, as well as ordering them. One communicaiton task for Product Owners is to be sure that the backlog items are understood, visible and transparent. With this burden of tasks and responsibilities, the entire organisation have to respect these decisions. (ScrumGuides.org, 2022).

3.6 Agile development in a large organisation

The purpose of this thesis is not to plan for more or new Product Owners or classic "bosses" in a hierarchical company. The aim is to look at the roles and responsibilities that fit well for the company and its software development department, GoD. The website less.works simplifies the modern role of leadership quite well into two points. First one is that manager has not control of what the team is working on and second on is that teams has power to choose that how team is working to reach the goal. (The LeSS Company B.V., 2022).

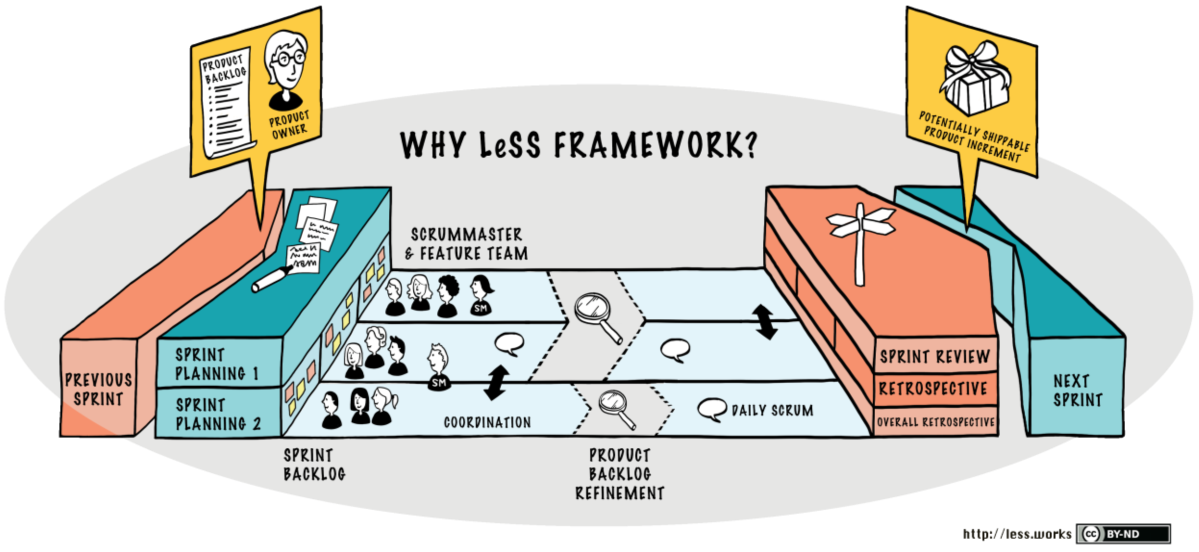

Figure 5. The LeSS framework (The LeSS Company B.V, 2022)

The same page has a clear statement which clarifys a role of the middle management. They are in responsibility to improve the organization's ability to create excellent products. However, Figure 5 shows the basics of the LeSS framework. The main point in this case is to notice the idea of sprints. On the left side is the previous sprint, and on the right side, the next sprint. Figure 5 illustrates well the “endless” queue of springs. When scaling agile in large organization, the study reveals that key determinants of successful agile scaling include company culture, prior experience with agile and lean methodologies, support from management, and alignment of values. (Kalenda, Hyna, & Rossi, 2018)

3.7 Communication in agile development

One aspect, probably the most crucial thing for success, is good communication in development teams. As mentioned in the section about Scrum, some ways of communicating are included in the framework, but in my view, these cannot be the only ways of communicating. The team has other ways to communicate than just planned meetings.

To help companies to improve communication in development teams, a company called 21Twelve Interactive has created a list for this (21Twelve Interactive, 2019).

These ten methods are:

1. Dwelling on mistakes. The idea here is to bring mistakes out into the open as soon as possible so that they can be solved quickly.
2. Non-verbal communication. This means how people behave and interact with others.
3. Defining roles from the very beginning. A lack of clearly defined roles might cause a gap between roles and responsibilities, but naturally, an agile team should be able to change or update roles as needed.
4. Feedback. Giving and receiving feedback frequently is important – meaning both positive and negative feedback.
5. Intercultural communication. It is crucial to understand cultural differences, and handling this makes it possible to collaborate as a single unit.
6. Intro-Retrospectives. Retrospective meetings are one way of communicating.
7. Encouraging questions. Dismissing questions is not recommended.
8. Varying the approach. In agile teams, some old techniques from yesterday might not work anymore today.
9. Enhancing transparency. Development team members should be encouraged to promote transparency.
10. Uniting individuals with purpose. Engaged employees will also be active in communication.

On the same page, there is a good list of where lack of communication can lead in teams (21Twelve Interactive, 2019). Therefore, this thesis does not focus on communication between individuals in development teams; rather, a list of challenges to show that many different problems can easily be created. There are issues like trust erosion, conflicts, us versus them feelings and leaving people, just a few to mention. (21Twelve Interactive, 2019)

3.8 Management in software development

Even though the aim of the research in this thesis is about leading rather than managing, it is good to understand the needs of management as well.

Software development management can be anything related to organising, coordinating and directing creation, from the concept to completion. This requires controlling resources, budgets and timetables, as well as communication with stakeholders like clients, project managers developers, end users and others. Important functions in software development management are things like strategic planning, resource management, project management, team management, risk management and quality assurance. In strategic planning, the aim is to set the objectives and course for the development team. This work can include, as an example, the creation of a roadmap for the project. In resource management, the aim is to handle an endless lack of resources, such as time, innovations and a spending plan. (Tutorials & Examples, 2023).

With project management, the aim is to ensure that projects are finished on time, within budget and to the appropriate quality. This requires supervising the daily operations of the development team. Defining the project requirements, estimating timetables, creating project plans, monitoring progress, and correcting course, if needed, are crucial parts of project management. Team management means creating cohesive software development teams, and it requires good leadership, communication and coaching abilities. Everything also needs to be in line with the organisation's vision and values. Risk management is the process of analysing and reacting to existing risks and avoiding new ones in every possible way. (Tutorials & Examples, 2023).

To handle quality assurance in a team, the team needs knowledge of the methods surrounding development and other methodologies tied to quality assurance. In software development management, carrying out quality assurance work requires both seeking out and eliminating any bugs, safety issues or problems in the software (Tutorials & Examples, 2023).

3.9 Importance of leadership in software companies

There are many good books about leading development in general. On Amazon.de alone, there are 20 pages of similar books (Amazon.com, Inc. or its affiliates, 2023). The aim of this thesis is not to delve deeply into leadership. This thesis focusses on leading software development, especially the development of mobile apps at Elisa Corporation.

3Pillar Global lists ten traits for modern software development leaders. They see that perfect balance of technical expertise and emotional intelligence is needed for leading a software development team. Also focusing to related stakeholders and resources is important. (3Pillar Global, 2023). Ten traits by 3Pillar Global contains points like resilience, emotional intelligence and skill of delegation. (3Pillar Global, 2023)

Customer-centricity means having the mentality to focus on the client and end users. The idea is to keep the customer front and centre throughout the development cycles, keeping user experience at front of mind. *Adaptability* is important because unexpected requirements can arise from customers and stakeholders, and in app development, also from platforms and updates for mobile operating systems. Good leaders can adapt to rapid changes. Also, agile software development methods like Scrum help in this mindset. (3Pillar Global, 2023).

Having a *learner mentality* means having a mindset and readiness for endlessly learning new things. Leading a software development team well requires curiosity and a passion for learning to be able to deliver innovative solutions before competitors do (3Pillar Global, 2023). *Resilience* means the ability to “raise back on ride” after something unexpected has happened (Cambridge University, 2023). A leader of software development needs to have good resilience. (3Pillar Global, 2023).

Emotional intelligence is the skill of managing human emotions. This means e.g. understanding what the end user wants. Also empowering individual contributors, like developers, to deliver the best possible results. Development leads should have *deep technical knowledge* of various technologies. This might be a slightly utopian idea, but in a perfect world, people leading software development understand everything from the technical stack to design, testing, etc. *Strong communication skills* help in communicating each team members. With good communication skills, leaders can pass requirements from clients to the development team. *Servant leadership* means the idea that leaders are not just for blocking things and increasing organisation. They should be enablers, empowering people and removing impediments to progress (3Pillar Global, 2023).

Delegation is not a software development-related skill. One classic issue among leaders is the lack of ability to delegate tasks. If this is not handled well, it can cause problems in modern, fast-moving and evolving teams. *Big-picture focus* means that, while individual developers can be focused on an extremely specific issue (such as a user interface issue with older iOS devices in the app that their team are developing), leaders also need to be able to focus on the big picture. This requires an understanding of the relationships and how they work together between each functionality (3Pillar Global, 2023).

Interesting found is that in the study case is found that most organizations are aren't ready for today's technology trends. Only a quarter them are successful at it today, showing a big "execution gap." (Lesser & Ban, 2016)

4 Case studies

In this section, there are a few examples of how similar cases are handled in other companies. This section utilises a thesis and other scientific sources to gain a better understanding than random blog posts usually give.

The first case, "Lahden Seudun Kehitys LADEC Oy" is an example of Finnish company that had issues with a silo mentality and its organisational structure. The second case is about legacy code and how large code base was upgraded in Enkora Oy. Third case study is done for unnamed fintech company in Norway, and it is about technical debt and how to manage it in big organisation with multiple teams.

4.1 Case Study - Company silos

The aim of the master's thesis *Silos as a Manifestation of Organisational Culture* by Riina Kivekäs was to study organisational culture and how it is connected to the implementation of a company strategy within a specialist organisation. The master's thesis was conducted as a qualitative case study at a regional business development company, Lahden Seudun Kehitys LADEC oy (Kivekäs, 2014).

In the thesis, Kivekäs has a proposal for decreasing silos and reducing the culture of silos. Kivekäs's thesis is in Finnish, so these main points and others as well are translated by the writer of this thesis.

These instruments were mentioned as breaking silos in the case study company:

- Leading
- Communication and interaction
- Trust (Kivekäs, 2014)

Leading is divided into three points:

1. CEO leading with a focus on openness and collaboration.
2. Support and tools from administration and board of directors to execute strategic work.
3. Understanding different cultures in the organisation

Communication and interaction are divided into two points:

1. Consistency of communication from administration
2. Actions that support openness and collaboration

Trust is divided into three points:

1. Improved recognition of employees and exploiting the expertise of specialists
2. Removing hierarchies in the organisation by involving specialists in everyday decision-making
3. Clear authorisation for collaboration inside organisations

In the summary, Kivekäs splits the practical recommendations that can be proposed to Lahden Seudun Kehitys LADEC oy into two main points. Here are abstracts of those points:

1. The strategy and values of the company need attention. The structure of the organisation, its mental and physical condition and its incentives need to be strengthened to support internal collaboration. Communication needs to be convergent throughout the organisation. Also, projects must be in line with the company's goals to strengthen operational collaboration to achieve new, more effective operating models and to increase customer utility.

2. There is a need for increased trust. To enhance trust within the organisation, attention must be paid, on one hand, to clarifying the organisation's goals while acknowledging the differing perspectives between various work cultures in the process of forging a common future. On the other hand, it is essential to increase opportunities for interaction among the members of the organisation to effectively harness the organisation's expertise. The administration needs to have confidence in employees' judgment and the mandate for collaboration between the operations to increase the effectiveness of the organisation (Kivekäs, 2014).

4.2 Case Study - Legacy code

In the master's thesis *Framework change for modernization of webservice* by Alice Thomas, the aim was to find out how code modernisation can be performed on a huge amount of code lines and to understand its effects. *Code modernisation* means, in this case, that CoffeeScript- and Backbone-based code was converted to TypeScript to implement the Angular framework. Moving to TypeScript was seen to provide a more homogeneous frontend solution. The case study company, Enkora Oy, creates software and hardware solutions to clients in Finland (Thomas, 2020).

Even though that thesis is about web development, we can assume that in handling legacy code as a phenomenon and an issue of processes, web development is similar enough to mobile app development.

The case study company's code base was huge. The company is over 10-year-old. Their business is in software solutions. The scope of the thesis was to understand the effects of transitioning from one framework to another (Thomas, 2020).

The thesis also includes a section called "Modernization strategies", which includes a set of ways to handle legacy code. The four main strategies are *replacement*, where old code is replaced with the newest (or newer) components, *wrapping*, where the legacy system is encapsulated, *redevelopment*, where the functionality is redeveloped and can be represented a in new form, and *migration*, where the legacy system is moved to a new environment (Thomas, 2020).

In the section “Modernization methods”, four different methods are listed: Chicken Little, Cold Turkey, Renaissance and Butterfly. In *Chicken Little*, the primary concept involves achieving the goal through the gradual migration of legacy software in small increments. Every step takes a tiny investment and a short time and produces a reasonable result. In *Cold Turkey* an idea to write everything from scratch. *Renaissance* methodology includes four main phases: evolution planning, evolution implementation, delivery and deployment. In *Butterfly*, the idea is to separate data migration from development phases. Once the development is completed, then data is migrated over from the legacy system (Thomas, 2020).

In the section, “Modernization Plan”, Alice Thomas ends up recommending that Enkora Oy use the Chicken Little method for their modernisation plan. This choice was made because it was essential for both the legacy system and the new target system to operate concurrently. In the thesis, Alice Thomas listed an 11-step plan for the modernisation phases.

4.3 Case Study - Technical debt

In the thesis *How to get away with technical debt: An explorative multiple-case study on autonomous teams and technical debt management*, Karl Omar Skeimo says that technical debt can slow down developers for creating value. (Skeimo, 2021). Skeimo’s thesis investigated how development teams actively manage technical debt, and it includes numerous case examples.

Skeimo’s master’s thesis includes a lot of good and deep background on technical debt, how to manage it and the research behind of all of this. It is not in the scope of this thesis to delve deeply into technical debt; it is more meaningful is to compare the founding of development teams.

Half of the teams in the research didn’t use specified tools to follow technical debt; the other half used monitoring tools actively where were visualized of technical debt. Every team documented their technical debt in their document tool to document to different extents. There weren’t processes in most of team to follow technical debt; however, one development team had improved processes also for this. The case company did not make it a standard practice to monitor technical debt. (Skeimo, 2021).

Technical debt measuring was not in the top priority of any team. Mostly decision was based on hunch. All the teams engaged in technical debt prioritisation at some way, and most of them seemed to put prioritisation as an everyday part of their development (Skeimo, 2021).

Teams primarily addressed tech debt by refactoring or rewriting the code. Also, most of the teams had reviews and guidelines that were followed. These practices should prevent creating new technical debt in their development. (Skeimo, 2021).

While Skeimo's instructions for different teams were specified for each team in the thesis, it does not make any sense to list them here.

5 Questionnaire results

In this thesis, the idea was to research how to improve the leadership of mobile app development at Elisa Corporation. To do this, one research method was themed interviews. This section analyses the answers for each question. The questionnaire itself can be found as an attachment. The questionnaire included a topic about which areas leadership is at a good level and where it is not, which included few subcategories. The goal here was to ask for ideas about how developers are feeling now. This question was split into three detailed questions.

When examining the responses, two points stand out:

- People did not feel the leadership was sufficient and thought there was a lack of support from upper levels of the organisation. Some raised the question of whether there was any leadership at all.

- Developers must make all decisions by themselves. This was seen as an agile way of doing things, but it also added some amount of extra weight to everyday development. On the other hand, some responses mentioned that developers making choices by themselves could also be seen as an opportunity to make bigger changes, and this way of working was seen as being agile.

Also, in one answer, the role of Product Owner was mentioned. The response stated that Product Owners lead product development, but no one leads app development or the necessary upgrades for its legacy parts.

Technology choices. There was some variance in the responses to the question about technology choices. In some teams, developers can make all the necessary decisions themselves, and this was seen as a benefit. Also, being given time to learn new things was raised as a benefit. Individuals taking responsibility was mentioned when technological choices are made. These might be also bad decisions, looking over the entire lifetime of a project. On the negative side, some developers feel that they are stuck with some legacy systems and there is no clear path away from them. Many developers felt that they have made the best choices and picked the best tools for their project, so there was no need to outsource leadership. A few stated that they have got some support from the team's architect.

Architectural/platform questions. The platforms' strict rules were raised as a guideline for many choices. Some respondents felt that strict rules about what the platform supports do not provide space for broader decisions. In one team, the team members had recently concluded that there should be more shared code and libraries between teams and products, but no one has taken any action to bring this about. Their architectural choices support this kind of activity. Also, if a single developer in a larger team has been willing to think about architectural questions, there has been room to raise this role and effect as well. One developer felt that they received good support when starting a project, and that architectural and platform-related questions were topical.

How well does leading "contribution to the success of an organisation" work at the moment? Many respondents thought that this was a difficult question to answer. Some said directly that they didn't have any clue that how they created value for the company, or they thought that creating value for the organisation's success was too abstract for everyday work. Some respondents mentioned that the way projects are led is strongly business focused, and they felt the development point of view was lacking in the leadership. Tracking customer satisfaction and metrics for measuring it (such as Net Promoter Score) were mentioned as ways of monitoring success.

A few felt that a lack of transparency, including inside their own business unit, blocks this. No cross-team experiments have been carried out. Some knowledge spreading has taken place, but this was related to individual developers and their activities. Data about what happens in different teams (e.g. from the developers' point of view) can be very limited, if it exists at all. On the other hand, the ways in which designers work and share knowledge for better value creation were mentioned as an inspiring example.

How mobile app development is led to support the company's vision. The company's vision was reviewed before any answers were recorded. Some developers felt that their product focuses on small features and minor details, so there is no pressure to have them as a global benchmark. One respondent also said that developers are in survival mode and trying to cope with being overloaded with legacy code. Handling legacy code and tools received multiple mentions and were perceived as obstacles to making improvements. Even when developers are creating a product just for the local market, they did not perceive and obstacles to the aim of being a global benchmark.

Silos were also mentioned in these responses. One developer raised the issue that even though they are trying to improve their products and working habits, these good learnings are not spread within the company, in their business unit, or even within their service team. One respondent suggested that increased activity in mobile app communities supports the implementation of this vision.

Some of the responses included questions, such as whether there is any leadership at all in this area. One suggestion was that maybe having a shared vision for mobile app developers could help those developers onto a shared path. The company's vision was also seen as abstract, with some developers not feeling that it has any effect on their everyday lives.

The topic for the next set of questions was about who leads in making choices about what things are taken into use, how they are used, what is the goal in their selection and how to maximise value creation. This topic was split into separate subtopics, which are presented here in their own sections.

Support tools in this case means CI/CD, release processes, GitHub Action, etc. In one response, the developer felt that they had to spend too much time on these kinds of support tools and proposed that another team/person should have an increased role in maintaining these tools. On the other hand, some respondents felt that the current method – where SRE teams provide some tools and some guidelines – works well, and they were satisfied with it. Multiple responses also raised the idea that developers or teams can try out new tools for themselves. Also, it was unclear who has chosen the support tools in use and based on what facts. Working in silos was also raised in these responses. Developers are not aware of which tools are in use among another teams.

Value Conflicts. The question was: When short-term goals are in opposition to long-term goals (if they exist), who resolves these issues and based on what policy/values? Some developers have not observed this kind of situation and were unable to give answers. At the same time, other developers felt that they had to struggle with value conflicts in their everyday lives and tried to get by doing this. Many answers raised the role of Product Owner and its responsibilities. In larger teams, developers felt able to discuss with others and have discussions with each other, and try to resolve conflicts this way, especially in cases where there is no Product Owner in the team.

The most generic question in questionnaire was: Are there any other things where you feel that you would like to have more leadership? It was quite common that developers wanted to raise something that they have also already said in previous responses. One pointed thing is a lack of vision about what is good development and what is not. One developer felt that they were forced to make a big decision in their development work without any guidelines or leadership. Also, silos are a serious issue, and there are difficulties in moving to other teams when the technology choices vary so dramatically between teams.

The issue was also raised that some developers do not have time to upgrade the technologies they are using to the latest ones, and business pressure might result in technical shortcuts. One respondent also mentioned an example project, where developers were just chasing the business goals, and after a while, the code base was so full of legacy code that they had to start a new project from scratch. Increasing technical debt was mentioned in multiple answers. The increasing number of out-of-date components in projects was raised as a risk for the future. There was also frustration that no one takes the lead in making these big, but at some point, necessary, choices, and prioritisation of items for the backlog was mentioned. One developer raised the concept of green coding and sustainability, believing that their current project, with a huge amount of legacy code, is far from meeting these trends and targets.

Overall, leading mobile app development and technical choices was seen as being invisible. At the same time, some developers felt that there have been good changes and improvements in communication between mobile app developers, and this has lowered some barriers to cooperation.

Some of the respondents were studying at the same time and working part-time. As they are transforming from students into employees, the following questions were added to obtain their current feelings about that process.

The first question was “How have your ‘first steps in app development’ been led?”, with clarifying follow-up question, “Are there any areas where leadership is completely absent?” The answers were quite positive. The interviewees were enjoying the atmosphere and the way they had begun their journeys and believed it was stress-free. Also here, legacy code was mentioned, as it was seen as making joining and understanding a project more difficult. Technical leadership was not mentioned, but project managers received good feedback. A lack of leadership was not seen as a particularly big issue at this point.

The next question was, “Are your vision for your master’s thesis and your next steps in education being supported?” The interviewees gave good feedback about taking the first steps in their careers. Also, when developers are having discussions about thesis ideas, the company has a good list of potential topics. However, some ideas had been added to the list by people at upper levels of the organisation, and getting developer support for those was difficult. This was observed in that prioritisation of topics and tasks could change quite rapidly. Here, the students would like to have some technical support and leadership.

6 Research results

This thesis examines three research questions:

- What is the current state of leadership in mobile app development at Elisa Corporation?

- What are the primary challenges in management and leadership, as perceived by developers within the context of mobile app development?

- How can the issues of leadership in the mobile app development area that have emerged be fixed?

To get answers to the first question, the goal was to figure out the current state and feelings about leadership of mobile app development at Elisa Corporation. To find out the current state, I interviewed mobile app developers to get a clear view of this.

Findings:

- Mobile app developers felt they were not getting any leadership and that there was a lack of support from upper levels of the organisation.
- Feeling a lack of leading raised the question of whether there is any leadership at all.

When thinking about how to fix these, some steps were raised:

- If there are some leadership, guidelines or other visions that every developer should be aware of, this can be improved by more active communication. Section 3.2 of this thesis briefly lists issues that having silos in a company and between key players (which in this case can include leaders, managers and developers) might cause.
- In sections 4 and 4.1, there is a good example and case study of how a company improved their situation. Riina Kivekäs' master's thesis "Siilot

organisaatiokulttuurisena ilmentymänä" (Kivekäs, 2014), has a very good section about company silos and how the case study company was guided to remove them. To do this, she raises the issue that alongside improved communication, the mundane work of the strategy and values of the company needs attention. The structure of an organisation as well as the mental and physical condition of and incentives for employees need to be strengthened to support internal collaboration.

Issues related to questions about what the primary challenges in management and leadership are as perceived by developers within the context of mobile app development were raised strongly in the questionnaire, and fixes to these problems are more or less in everyday development. Also, some of the proposed fixes are closer to management than leadership, but naturally, management needs leading, and there were other tasks as well.

Findings:

- Making larger decisions within a project was felt by a few developers to be a burden. When conducting interviews and going through the questionnaire with developers, it was particularly notable that teams with a remarkably small number of developers or where all or most of the developers were at a junior level were the ones that raised these kinds of issues. As an example, one big question was about what level of support there should be in the products being developed for old devices or operating systems and how to sell the idea of upgrading the required hardware to business executives, given that it could result in losing some customers.
- Some developers felt that having responsibility for technological choices gave them enough space to make good decisions, while others felt that this was an extra task.
- In everyday work, projects are so strongly business-led that there is no simultaneous technology or development leadership at all.
- Some developers felt that whether they contribute to the success of the organisation is quite a difficult question to answer. Some said directly that they do not feel that they contribute to the larger group.
- There is no visibility over other developers, sometimes even within the same product development team.
- Silos were mentioned in multiple answers. Silos creates similar issues than listed in section 3.2 Silos.

As noted in section 3.3, no methods or project management tools provide a perfect way to handle technical debt or architectural questions. With improved communication and some guidelines, or support from developers with the required seniority or software architects, junior developers could get help with these questions.

For the question of how issues of leadership that have emerged in the mobile app development can be fixed, the paragraphs below list development areas to improve the situation in mobile app development.

Ideas how to improve current situation:

- Creating a companywide policy for details like this could also provide some basis for discussion within product teams. Having these kinds of strict but widely accepted policy or architecture guides could also help developers to raise issues about maintaining legacy code and get tools to change the priority of items in the backlog.
- Suggestive guidelines for technological choices could be beneficial. When developers felt that they have good components that they should share with others but just have not done so, I understand that this is now more of an issue of the company culture than one of individual laziness. With the correct tools, this is technically quite easy to set up, but creating a culture with an open-source mindset and keeping it breathing takes some effort. I see that this could be worth the effort and could help to break down silos. (See section 3.2.)

For the fact that some developers felt that they were outside the larger organisation, I see this also as a communication issue. Here, organisational silos were also raised as a blocking element.

How to fix:

- Section 4.1 includes a case study about how these can be analysed and fixed. In particular, if a mobile app development team has created an organisational silo within the team, I would cease development and make it a top priority to fix this issue. Once this is done, then product development can continue through removing obstacles and eliminating legacy code or other technical debt.

When we were thinking about how mobile app developers' work supported Elisa Corporation's vision, I could see the pain of the developers as they attempted to handle all the conflict between the everyday reality and high-level vision of the company.

Findings:

- Some developers felt that they are focusing on minor details and that their product is not even trying to be "global benchmark".
- The growing amount of legacy code was mentioned, and developers felt that they were struggling to keep a falling tower in one piece. Some of them mentioned that they were in survival mode.

Section 3.1 explains how many different kinds of stakeholder's mobile app developers need to satisfy, which might result in conflicts.

In reaction to this alarming message from developers – that they are in survival mode and are not able to do their best – I would expect some instantaneous action from the leaders of the business unit to identify the root causes of this and plan remedial action.

If product teams are satisfied with just being a local player and not even trying to focus on being a global benchmark, then the main issue is somewhere in the product management, which this thesis does not cover.

There were also some unexpected findings. When discussing supporting tools, there was some uncertainty about who had chosen them and based on what values. Also, some developers felt that they needed to spend too much time getting these tools to work properly. Generally, there was quite positive feedback about handling supporting tools. In any event, when selecting tools that effect to mobile app developers, improving communication about processes, policies, values and expected outcomes would help developers who are struggling with workloads.

The feedback from part-time workers was mostly positive, but a lack of technical leadership in the mobile app area was also raised. I got the idea that app development guidelines can somehow be cross-pollinated with a list of topics for academic theses. This could make it easier to check that what ideas are relevant right away, where these ideas have appeared and what policy they are leaning towards.

6.1 Open questions

Based on the research results, I have some questions in my head that can be left open in this thesis. These questions may help leaders at Elisa Corporation to improve the state of mobile app development or help students to find topics for their thesis, or they can be left as open questions.

- How are mobile app development teams structured within Elisa Corporation? Are they able to create value, as descripted in the section about Scrum or in other sections about agile methodologies. Does the current structure support developers in creating world-leading apps in the way that company's vision expects?
- How are decisions made within mobile app development teams and why are some developers unhappy about this? Could other developers help in any way?
- How do leaders foster innovation within mobile app development teams? Do teams have resources for innovation?
- How does Elisa Corporation support leaders in the area of mobile app development? Is there something that can be improved? Do managers have all the information and tools they need for success?
- Are architects and lead architects getting the leadership they need to do their best to support mobile app development?

6.2 Reliability and validity

As noted in the "Research setting" section, the most difficult part of preparing this thesis was the lack of scientific sources: it is notable that commercial sources and sources such as blog posts play a prominent role in the references and citations. It may be that terms such as "silo" are commonly used in working life and in communication within corporations and between developers but have not yet become very common in theses. However, I have tried to avoid personal blog posts, and I have removed irrelevant material from commercial posts.

On the other hand, I have been able to use original sources, such as scrum.org for Scrum and agilemanifesto.org for Agile principles. There sources provide the best and latest information about these topics, without any one individual's interpretation. Most likely, the principles of Agile development and related topics will not be outdated any time soon. Also, classic issues in corporations (such as communication silos) will not disappear either. With good fortune, some issues raised by developers could already have been fixed by changes at Elisa Corporation.

6.3 Reflection on the research process

I started studying at HAMK in autumn 2021, and this thesis was written in winter 2023/2024. This took too much time. Due to the nature of modern software companies, the people and processes as well as the tools and services used are changing and evolving all the time. It may be that while I have been writing this thesis, some things have already changed and improved. On the other hand, I believe that not everything has yet been fixed and implemented in every part of every process for everyone who participates in development work.

Also, one issue is that I am employee of Elisa Corporation, I know the mobile app developers, and they know me. I hope that already knowing me did not affect the responses from the participants, at least not in terms of holding back their opinions.

When writing this thesis, I have also noticed that it is very difficult to take a position as a reader with little knowledge of mobile app development and/or Elisa Corporation. It may be that I have missed some topics or questions that should be dealt with, but that I thought would be clear for everyone.

7 Bibliography

21Twelve Interactive. (2019, November 27). *0 EFFECTIVE WAYS TO IMPROVE COMMUNICATION IN AGILE TEAMS*. Retrieved from 0 EFFECTIVE WAYS TO IMPROVE COMMUNICATION IN AGILE TEAMS: https://www.21twelveinteractive.com/improve-communication-in-agile-teams/

3Pillar Global. (2023, January 28). *10 LEADERSHIP TRAITS FOR MODERN SOFTWARE DEVELOPMENT LEADERS*. Retrieved from 10 Leadership Traits for Software Development Leaders | 3Pillar Global: https://www.3pillarglobal.com/insights/10-leadership-traits-for-modern-software-development-leaders/

Alam, I., Sarwar, N., & Noreen, I. (2022, April 1). Statistical analysis of software development models by six-pointed star framework. *PloS one, 17*(4). doi:https://doi.org/10.1371%2Fjournal.pone.0264420

Alrabaiah, H. A., & Medina-Medina, N. (2021). Agile Beeswax: Mobile App Development Process and. *Sustainability (Basel, Switzerland)*, 33. doi:https://doi.org/10.3390/su13041909

Amazon Web Services, Inc. or its affiliates. (2023). *What Is SDLC (Software Development Lifecycle)?* Retrieved from What Is SDLC (Software Development Lifecycle)?: https://aws.amazon.com/what-is/sdlc/

Amazon.com, Inc. or its affiliates. (2023). *Results*. Retrieved from Amazon.de : leading development: https://www.amazon.de/s?k=leading+development&crid=1QQBR3Z1X5X70&sprefix=leading+development%2Caps%2C105&ref=nb_sb_noss

Apple Inc. (2023). *App Store Review Guidelines*. Retrieved from App Store Review Guidelines: https://developer.apple.com/app-store/review/guidelines/

Apple Inc. (2023). *Apple Developer Documentation*. Retrieved from Apple Developer Documentation: https://developer.apple.com/documentation/

Apple Inc. (2023). *Human Interface Guidelines*. Retrieved from Human Interface Guidelines: https://developer.apple.com/design/human-interface-guidelines/guidelines/overview

Apple Inc. (2023). *App Review*. Retrieved from App Review: https://developer.apple.com/app-store/review/

Barrera, I. (2020, June 24). *Optimizing the mobile release process.* Retrieved from Optimizing the mobile release process: https://www.runway.team/blog/optimizing-the-mobile-release-process

Beck, K., Beedle, M., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., . . . Bennekum, A. v. (2001). *Manifesto for Agile Software Development*. Retrieved from Manifesto for Agile Software Development: https://agilemanifesto.org/

Cambridge University. (2023). *Meaning of resilience in English*. Retrieved from RESILIENCE | English meaning - Cambridge Dictionary: https://dictionary.cambridge.org/dictionary/english/resilience

Cano, E. L., García-Camús, J. M., Garzás , J., & Moguerza, J. M. (2021, July–September). A Scrum-based framework for new product development in the non-software industry. *Journal of Engineering and Technology Management, 61*. doi:https://doi.org/10.1016/j.jengtecman.2021.101634

Carlo, N. (n.d.). *What is Legacy Code? Is it code without tests?* Retrieved from What is Legacy Code? Is it code without tests?: https://understandlegacycode.com/blog/what-is-legacy-code-is-it-code-without-tests/

Cherednichenko, S. (2021, November 5). *Stakeholders in the Software Development Process: Identifying and Distributing Responsibilities*. Retrieved from Stakeholders in the Software Development Process: Identifying and Distributing Responsibilities: https://medium.com/mobindustry/stakeholders-in-the-software-development-process-identifying-and-distributing-responsibilities-637a4b3c2672

Concepta Technologies LLC. (2022). *How to Define Stakeholders for Your Software Development Project*. Retrieved from How to Define Stakeholders for Your Software Development Project: https://www.conceptatech.com/blog/how-to-define-stakeholders-for-your-software-development-project

Davis, L. (2022, March 22). *What Is Agile Project Management? The Ultimate Guide*. Retrieved from What Is Agile Project Management? The Ultimate Guide: https://www.forbes.com/advisor/business/what-is-agile-project-management/

Donato, H. (2023, July 31). *5 Phases of Project Management Life Cycle You Need to Know*. Retrieved from 5 Phases of Project Management Life Cycle You Need to Know: https://project-management.com/project-management-phases/

Elisa Corporation. (2023). *A sustainable future through digitalisation*. Retrieved from About Elisa - Elisa: https://elisa.com/corporate/?_ga=2.87151514.365937514.1694957904-1858593870.1694957904

Elisa Corporation. (2023). *Mission and values*. Retrieved from Elisa.com: https://elisa.com/corporate/about-elisa/mission-and-values/

Elisa Corporation. (2023). *Services for private customers in Estonia*. Retrieved from Elisa.com: https://elisa.com/consumer-services/private-customers-estonia/

Elisa Corporation. (2023). *Services for private customers in Finland*. Retrieved from Elisa.com: https://elisa.com/consumer-services/private-customers-finland/

García, J., Amescua, A., & Sánchez, M.-I. (2011, August). Design guidelines for software processes knowledge repository development. *Information and Software Technology, 53*(8), 834-850. doi:https://doi.org/10.1016/j.infsof.2011.03.002

Google LLC. (2023). *Android Mobile App Developer Tools – Android Developers*. Retrieved from Android for Developers: https://developer.android.com/

Google LLC. (2023). *Dart overview*. Retrieved from Dart overview: https://dart.dev/overview

Google LLC. (2023). *Flutter architectural overview*. Retrieved from Flutter architectural overview | Flutter: https://docs.flutter.dev/resources/architectural-overview

Gorbachenko, P. (n.d.). *What is Legacy Code and How to Deal with It [Updated 2021] | Enkonix*. Retrieved from Everything You Need to Know About Legacy Code: https://enkonix.com/blog/legacy-code/

Grammarly Inc. (2021, June 3). *5 Actionable Ways to Improve Communication Between Departments*. Retrieved from Grammarly: https://www.grammarly.com/business/learn/improve-communication-between-departments/

Gromenko, A. (2022, 5 10). *Complete Guide to Legacy Application Refactoring (Part 1): Meaning, Benefits, and Code Examples*. Retrieved from Complete Guide to Legacy Application Refactoring (Part 1): Meaning, Benefits, and Code Examples: https://code-care.com/blog/complete-guide-to-legacy-application-refactoring/

Hwang, E. H., & Krackhardt, D. (2020, January). Online Knowledge Communities: Breaking or Sustaining Knowledge Silos? *Production and operations management, 29*(1), 138-155. doi:https://doi.org/10.1111/poms.13098

Indeed. (2022, June 25). *How To Break Down Silos Within an Organization (With Steps)*. Retrieved from How To Break Down Silos Within an Organization (With Steps): https://www.indeed.com/career-advice/career-development/breaking-down-silos

Indeed. (2022, July 22). *What Are Silos in Business? Causes for Organizational Silos*. Retrieved from What Are Silos in Business? Causes for Organizational Silos: https://www.indeed.com/career-advice/career-development/silos-in-business

Jabangwe, R., Edison, H., & Duc, A. N. (2018, November). Software engineering process models for mobile app development: A systematic literature review. *Journal of Systems and Software, 145*, 98-111. doi:https://doi.org/10.1016/j.jss.2018.08.028

Jackson, S., & Brannon, S. (2018). In-house Software Development: Considerations for Implementation. *The Journal of academic librarianship, 44*(6), 689-691. doi:10.1016/j.acalib.2018.10.008

Kalenda, M., Hyna, P., & Rossi, B. (2018, MAy 16). Scaling agile in large organizations: Practices, challenges, and success factors. *Journal of software : evolution and process, 30*(10). doi: https://doi.org/10.1002/smr.1954

Kivekäs, R. (2014). *Siilot organisaatiokulttuurisena ilmentymänä.* Retrieved from Siilot organisaatiokulttuurisena ilmentymänä: https://lutpub.lut.fi/handle/10024/96815

Kotlin Foundation. (2023). *Build a full-stack web app with Kotlin Multiplatform*. Retrieved from Build a full-stack web app with Kotlin Multiplatform | Kotlin Documentation: https://kotlinlang.org/docs/multiplatform-full-stack-app.html

Kotlin Foundation. (2023). *Kotlin Multiplatform*. Retrieved from Kotlin Multiplatform | Kotlin Documentation: https://kotlinlang.org/docs/multiplatform.html

Lesser, E., & Ban, L. (2016). only a quarter of the companies studied are successful at it today, so there exists a significant “execution gap.”. *Strategy & Leadership, 44*(1), 44-47. doi:https://doi.org/10.1108/SL-11-2015-0083

Lutkevich, B. (2023, February). *What is technical debt?* Retrieved from What is technical debt?: https://www.techtarget.com/whatis/definition/technical-debt

Marin, G. (2022, 11 3). *White label software: Definition, benefits, and types*. Retrieved from White label software: Definition, benefits, and types: https://inevent.com/blog/tech-and-trends/what-is-white-label-software.html

McAbee, J. (2022, June 2). *What Are the 5 Scrum Values?* Retrieved from Understanding the 5 Scrum Values | Wrike: https://www.wrike.com/blog/scrum-values-guide/

Meta Platforms, Inc. (2023). *React Native*. Retrieved from React Native · Learn once, write anywhere: https://reactnative.dev/

Meta Platforms, Inc. (2023). *React Native · Learn once, write anywhere*. Retrieved from React Native · Learn once, write anywhere: https://reactnative.dev/

Misra, S. C., & Singh, V. (2015). Conceptualizing open agile software development life cycle (OASDLC) model. *The International journal of quality & reliability management, 32*(3), 214-235. doi:https://doi.org/10.1108/IJQRM-08-2013-0127

Narasimman, P. (2023, April 21). *Agile vs Traditional Project Management [Top Differences]*. Retrieved from Agile vs Traditional Project Management [Top Differences]: https://www.knowledgehut.com/blog/agile/agile-project-management-vs-traditional-project-management

Narayanan, S., Balasubramanian, S., & Swaminathan, J. M. (2010). Managing Outsourced Software Projects: An Analysis of Project Performance and Customer Satisfaction. *Production and operations management, 20*(4), 508-521. doi:https://doi.org/10.1111/j.1937-5956.2010.01162.x

Neill, M. S., & Jiang, H. (2017, November). Functional silos, integration & encroachment in internal communication. *Public Relations Review, 43*(4), 850-862. doi:https://doi.org/10.1016/j.pubrev.2017.06.009

Paddle. (n.d.). *7 types of white label SaaS products & mistakes to avoid*. Retrieved from White-label SaaS products guide to stay competitive: https://www.paddle.com/resources/saas-white-label

ProductPlan. (2023). *What are the 12 Agile Principles? | Definition and Overview*. Retrieved from Agile Principles: https://www.productplan.com/glossary/agile-principles/

Royce, D. W. (1970). *MANAGING THE DEVELOPMENT OF LARGE SOFTWARE SYSTEMS.* Retrieved from MANAGING THE DEVELOPMENT OF LARGE SOFTWARE SYSTEMS : https://blog.jbrains.ca/assets/articles/royce1970.pdf

Schmitt, J. (2023). *Native vs cross-platform mobile app development*. Retrieved from Native vs cross-platform mobile app development: https://circleci.com/blog/native-vs-cross-platform-mobile-dev/

Schults, C. (2022, March 11). *7 Simple But Effective Steps to Reduce Technical Debt*. Retrieved from LinearB: https://linearb.io/blog/7-simple-effective-steps-to-reduce-technical-debt/

Scrum.org. (2022). *What is Scrum?* Retrieved from What is Scrum? A Better Way Of Building Products: https://www.scrum.org/resources/what-is-scrum

ScrumGuides.org. (2022, May 15). *Scrum Guide | Scrum Guides*. Retrieved from The 2020 Scrum GuideTM: https://scrumguides.org/scrum-guide.html

Singh, S. (2023, January 30). *Native vs Hybrid vs Cross Platform – What to Choose in 2023?* Retrieved from Net Solutions: https://www.netsolutions.com/insights/native-vs-hybrid-vs-cross-platform/

Skeimo, K. O. (2021). *How to get away with technical debt: An explorative multiple-case study on autonomous teams and technical debt management.* Retrieved from How to get away with technical debt: An explorative multiple-case study on autonomous teams and technical debt management: http://hdl.handle.net/11250/2826958

Smartsheet Inc. (2023). *Guide to Software Project Management | Smartsheet*. Retrieved from Software Project Management: Unique Skills for Highly Complex and Ever-Changing Projects: https://www.smartsheet.com/content/software-project-management

STX Next. (n.d.). *In-House Development vs. Software Development Outsourcing: A Comparison of Pros and Cons*. Retrieved from In-House Development vs. Software Development Outsourcing: A Comparison of Pros and Cons: https://www.stxnext.com/blog/in-house-development-vs-software-development-outsourcing-comparison/

The LeSS Company B.V. (2022, 12 4). *LeSS Framework*. Retrieved from LeSS Framework - Large Scale Scrum: https://less.works/less/framework

Thomas, A. (2020, December). Framework change for modernization of.

Tutorials & Examples. (2023, 6 19). *Role of Management in Software Development*. Retrieved from Role of Management in Software Development - TAE: https://www.tutorialandexample.com/role-of-management-in-software-development

Udvaros, J., Forman, N., & Avornicului, S. M. (2023). Agile Storyboard and Software Development Leveraging Smart Contract Technology in Order to Increase Stakeholder Confidence. *Electronics (Basel), 12*(2), 426. doi:https://doi.org/10.3390/electronics12020426

Walburg, M. (2021, March 25). *In-House Development – advantages and disadvantages*. Retrieved from In-House Development – advantages and disadvantages: https://binarapps.com/in-house-development-advantages-and-disadvantages/

Yli-Huumo, J., Maglyas, A., & Smolander, K. (2016, October). How do software development teams manage technical debt? – An empirical study. *Journal of Systems and Software, 120*, 195-218. doi:https://doi.org/10.1016/j.jss.2016.05.018

Zaman, U., Jabbar, Z., Nawaz, S., & Abbas, M. (2019, April). Understanding the soft side of software projects: An empirical study on the interactive effects of social skills and political skills on complexity – performance relationship. *International Journal of Project Management, 37*(3), 444-460. doi:https://doi.org/10.1016/j.ijproman.2019.01.015

Attachment 1: Interview base – Quotes and Background Information

How to improve and increase GoD´s role in leading mobile app development.

"Leaders, focus instead on working to generate a certain value that is over and above that which the team creates”.

"Leadership, on the other hand, is the ability of an individual to motivate, influence, and enable other employees to make a contribution to the success of an organization."

"A leader is someone who always takes the initiative and invests a great effort to accomplish the company’s vision. That is the only reason why people around start following them. "

"The role of management is to control a group or group of individuals in order to achieve a specified objective. Leadership is the ability of an individual to influence, motivate, and enable others to contribute to the organization's success."

All quotes from https://www.simplilearn.com/leadership-vs-management-difference-article

Attachment 2: Questionnaire

In which area leading is in good level and were not.

- Professional support in development
- Tech choices
- Architectural/platform questions

How mobile app development is led to support the company’s vision?

Who lead choices that 1) what are taken is use? 2) how those are used 3) what is the goal choosing these and 4) how to maximize value creation:

- Supporting tools: (CI/CD, releasing processes, GHA, cloudification)
- Value Conflicts. When short time goals are against long time goals (if exists). Who solved these and based to what policy/values.

Any other thing where you feel that you would like to have more leadership? Do you see some things that are completely out of sign or there are risks for big failure, if something is’ not done in nearby future?

(For part time workers whose studies at same time)

- How "first steps in app development" in has been leaded? Is there some area that is completely out of leading.
- Visions for master thesis, next steps (in educations) supported?