UNIVERSITY OF TARTU
Institute of Computer Science
Conversion Master in IT Curriculum

Erki Kilu

Software Process Improvement Using Agile Methods in Financial Institutions. LHV Bank Case

Master's Thesis (15 ECTS)

Supervisor(s): Fredrik Payman Milani, PhD

Tartu 2018

Software Process Improvement Using Agile Methods in Financial Institutions. LHV Bank Case

Abstract:

Large financial institutions and fintech companies have fundamentally different ways of working. More clearly than anywhere else, this is seen in their product development cycles. There is significant difference in time to market for new products and in speed of software development. The problem where many financial institutions find themselves now is the need to respond faster to the changes in the business environment and have faster product and software development processes.

Large financial institutions have historically relied on waterfall-inspired methods for software development. These methods have delivered great value for a long time, but are not corresponding to the current changing needs in the business environment. A larger shift from waterfall towards agile software development in these organizations has taken place just in the last five years due to the changes in the competition, where the new generation fintech companies have relied purely on agile development.

In light of this context, this thesis addresses the research question of how agile software development process can be scaled up within the context of financial institutions. This is achieved by means of a case study carried out on LHV Bank software development process. The current processes at LHV Bank are mapped, suggestions for changes are derived through review of existing research on agile methods and from in-depth interviews. Based on the analysis, the findings that are the most important for benefiting from agile development are identified and suggested for improving the software development process at LHV Bank.

There are eight key recommendations for improving the process at LHV Bank. On the organizational level, the agile methods and management culture should be introduced in larger scale, including the management, business, product and software development, with relevant trainings to be organized. On the teams level, to increase the efficiency of the development, concrete product teams should be assembled, common objectives set for team members and more autonomy given to the teams. On the process level, the agile development method used with its components should be reviewed by learning from the latest best practices and experience the organization has collected during the last five years when implementing agile practices. On the technical level, the release process should be automated and modular system architecture and microservices should be used more.

Keywords:

software process improvement, agile methods, financial institutions, LHV

CERCS: P170 Computer science, numerical analysis, systems, control

Tarkvara arendusprotsessi parendamine kasutades väledaid meetodeid finantsinstitutsioonides. LHV Panga juhtum

Lühikokkuvõte:

Töökorraldus suurtes finantsinstitutsioonides ja finantstehnoloogia ettevõtetes on rajatud erinevatele alustele. Selgemalt kui kusagil mujal on see näha nende ettevõtete tootearenduses, kus tuleb välja oluline erinevus uue toote turule toomise kiiruses ja tarkvara arendusprotsessis tervikuna. Paljud finantsinstitutsioonid püüavad lahendada probleemi, mis on seotud tootearenduse ja tarkvara arendusprotsesside kiirendamisega, et vastata ärikeskkonnast tulenevatele muutustele.

Ajalooliselt on suured finantsinstitutsioonid tuginenud oma tarkvara arendusprotsessides kose-meetoditele, mis tõid varem häid tulemusi, kuid mis ei vasta enam muutunud ärikeskkonnast tulevatele vajadustele. Suurem üleminek kose-meetodil toimivalt tarkvara arendusprotsessilt välk-meetodil toimivale tarkvara arendusprotsessile on nendes organisatsioonides toimunud alles viimase viie aasta jooksul. Uue põlvkonna finantstehnoloogia iduettevõtted on aga rajanud kogu oma tegevuse välk-meetodil põhinevale tarkvara arendusprotsessile.

Magistritöö eesmärgiks on leida vastus küsimusele, kuidas skaleerida väledaid tarkvara arendusprotsessi meetodeid finantsinstitutsioonides. Selleks viiakse läbi LHV Panga tarkvara arendusprotsessil põhinev juhtumiuuring. Magistritöös kirjeldatakse LHV Panga olemasolev tarkvara arendusprotsess, analüüsitakse läbi teoreetiline kirjandus ja viiakse läbi praktilised intervjuud. Võttes arvesse analüüsi tulemusel kogutud tähelepanekuid, pakutakse LHV Panga näitel välja ettepanekud, kuidas kiirendada tarkvara arendusprotsessi finantsinstitutsioonis.

Magistritöös tuuakse välja kaheksa ettepanekut protsessi kiirendamiseks LHV Pangas. Kogu organisatsiooni tasemel tuleb väledad meetodid ja juhtimiskultuur tervikuna kasutusele võtta laiemalt nii juhtimises kui ka äri-, toote- ja IT-arenduses. Selleks tuleb korraldada vajalikud koolitused. Meeskondade tasemel tuleb arenduse kiiruse tõstmiseks moodustada konkreetsed tootemeeskonnad, seada ühised eesmärgid kõikidele meeskonnaliikmetele ja anda meeskondadele suurem otsustusvabadus. Protsesside tasemel tuleb üle vaadata hetkel kehtiv tarkvara arendusportsess ning täiendada seda viimaste praktikate ja organisatsiooni enda poolt viimase viie aasta jooksul kogutud kogemustega. Tehnilisel tasemel tuleb automatiseerida tarkvara kasutuselevõtmisprotsess ja kasutada rohkem väiksematest osadest koosnevat infosüsteemi ülesehitust.

Võtmesõnad:

tarkvara arendusprotsessi parendamine, väledad meetodid, finantsinstitutsioonid, LHV

CERCS: P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimis-teooria)

Table of Contents

1. Introduction

Large financial institutions have historically relied on waterfall-inspired methods for software development. Such methods have delivered great value for a long time, but are not corresponding to the current needs in the changed business environment anymore. A larger shift from waterfall towards agile software development in these organizations has taken place just in the last 5-10 years due to the changes in the competition, where the new generation financial technology (fintech) companies have relied purely on agile development [1].

The problem where many financial institutions find themselves now is the need to respond faster to the changes in business environment and have quicker product and software development processes [1]. The financial institutions are increasingly investigating software process improvement with agile methods in order to compete with fintech companies. The old-generation software development models slowed down the processes of launching new products and services in large financial institutions. How to speed up and improve the software development process and compete with fintech companies with agile models, has risen the main question for financial institutions in the last few years [1].

However, there is no pure and perfect agile software development model suitable to every financial institution. The model should be adapted to take into account the constraints and regulations that heavily supervised financial institutions are following, and which makes these different from fintech companies. The question is, how financial institutions can learn from the agile techniques used in the fintech companies and find the best suitable model in order to compete with smaller organizations in the speed of software development.

Large financial institutions and fintech companies are having fundamentally different ways of working. More clearly than anywhere else it is seen in their product development cycles. There is significant difference in time to market of new products and in speed of software development between these two types of companies. Financial institutions with longer history have already large number of products that are based on legacy infrastructure, whereas fintech companies are focusing only on a small number of niche products using most up to date infrastructure [2]. Fintech companies are competing with existing financial institutions exactly with the same products, using their competitive edge in reacting to customer needs and delivering new solutions faster. Compared to banks, fintech companies are focusing only on one product and they are free to develop the whole software from scratch.

Large financial institutions have found their selves facing the problem, how to reorganize their internal development processes, so that they could keep up the same speed with fintech companies. There are many different causes, why these two types of companies are having so different delivering times, starting from general management principles of the companies and individual motivation of people up to technical tools used in the companies. In today's world, most of the financial products are delivered digitally to customers. Not only fintech companies, but also large financial institutions have started to brand their selves as IT development companies in large extent [3]. Therefore, the main reason in difference in speed of delivering new products is the software development process, starting from business analysis up to technical solutions used for delivering.

Nowadays, agile methods are used in the majority of the software projects. Agility is not used just in the fintech companies anymore, but also in the large financial institutions. The main challenge that needs to be solved in large organizations is how to scale the agile.

1.2. Problem Statement

In the light of the above context, the current thesis is focusing on finding the proposals to improvement and make more efficient agile software development methods for a large financial institutions taken into account the constraints coming from legal requirements, data protection, cybersecurity and regulation these organizations are having. The current thesis is concentrating on finding the proposals for improving the product development, software development and software delivery using agile methods that are scalable in large financial institutions.

1.3. Research Question

Large financial institutions have been actively looking for software process improvement (SPI) in the last few years to keep up the same speed with fintech companies. The research question of this thesis is how agile software development process can be scaled up within the context of financial institutions.

There is no such thing as perfect software development process, nor perfect agile software development process. Agile software development methods include number of various development frameworks and practices, but not all of these are suitable for financial institutions.

1.4. Research Design

To find solution and propose SPI using agile methods for financial institutions, a systematic literature review and comprehensive interviews are carried out. The interviews are done with two leading financial institutions (Swedbank, Bigbank) to get understanding if and how much have larger companies implemented agile methods in their software development processes and with two leading and fast growing fintech companies (TransferWise, Monese) to get information about best practices of agile methods used in these companies. Based on the findings both from literature and interviews, improving agile software development process suitable for large financial institution is proposed based on the LHV Bank case.

The current thesis is divided into four parts. In the first part, an overview of SPI, agile methods with components, scaling agile frameworks and financial institutions is given. In the second part, current software process at LHV Bank is described and improvement areas are brought up. In the third part, a literature review about agile methods in SPI is given, followed by the practices based on the summary of interviews conducted in various financial institutions and fintech companies. This part of the thesis ends with the discussion about the results and findings. The literature survey and conducted interviews contribute to the input of proposing SPI using agile methods for LHV Bank in the fourth part of the thesis. Threats to validity are discussed at the end of that part of the thesis.

2. Background

This part of the thesis gives an overview of the software process improvement, agile methods with components, scaling agile frameworks and financial institutions. In the agile methods section, more detailed description of Scrum, Extreme Programming, Lean Software Development and Kanban is given. In scaling agile frameworks section, more detailed description of Scale Agile Framework and Large-Scale Scrum is given. In financial institutions section, the main differences compared to any other organization are listed.

2.1. Software Process Improvement

Software Process Improvement (SPI) is an initiative to improve the processes of software development. Software process includes tools, methods and practices used in software development. Software process can be improved like any other process. As a first step, the existing software development process should be mapped. After that a vision of the required process should be drawn. To accomplish the improvement of the existing software process, a list of required changes should be formed, a plan to implement the changes prepared, resources committed and plan executed [4].

There are many different SPI models that are used. Two most commonly used models are Capability Maturity Model Integration (CMMI) and ISO/IEC 155047 Software Process Improvement and Capability determination (SPICE). CMMI has 17 core process areas, which include the goals and practices, how to reach the general goal. SPICE consists of 5 process categories, which include 9 process attributes assessed on a four-point rating scale. The other often used SPI models are Quality Improvement paradigm (QIP) [5], Goal-Question-Metric (GQM) [6] and Initiating, Diagnosing, Establishing, Acting, Learning (IDEAL) model [7].

The first SPI models were introduced in the 1990s to improve the traditional waterfall method software development processes. After the agile methods started to become more used, the SPI models were modified to take into account the new methods.

Waterfall-inspired methods for software development employ a linear development process. In the original waterfall model, software development process flows through certain phases like a cascading waterfall in the following order:

- system and software requirements captured in a product requirements document,
- analysis resulting in models, schema and business rules,
- design resulting in the software architecture,
- coding, i.e. the development, proving and integration of software,
- testing, i.e. the systematic discovery and debugging of defects and
- operations, i.e. the installation, migration, support and maintenance of complete systems [8].

Waterfall-inspired methods require a lot of planning, especially in the large financial institutions with legacy infrastructure, which allows these companies to allocate resources ahead of time. The main problem in the waterfall method is that the result of the required development is tested just at the end of the process. The risk is that the developed solutions do not meet the original needs or requirements. That is something that cannot be allowed in today's customer centric and fast changing world [2]. Long planning, forecasting and delivering can be used only in stable and relatively static environment that was a case for large financial institutions before, but not anymore when fintech companies have entered into the competition.

In traditional software development process, the SPI is used to reach universal and repeatable process that increases the predictability of the process. In agile software development, the goal is to improve the process to provide higher customer satisfaction by faster development time

and responsiveness to changes [9]. If SPI models in traditional software development processes concentrate on organizational level and outline what to do, then in agile development processes the models focus more on team level and outline how to do it [10].

2.2. Agile Methods

As the organizations are usually build in hierarchical way inspired from the military, it is often concluded that also waterfall method is emerged from the military world. However, it is important to note that also agile is a military word coming from strategies applied by the forces on the battlefield. The objective of agile is to adapt and exploit the chaos of the battlefield and, in order to succeed, to do it in a faster and better way than the enemy does [11]. Only in agile, the team is organized in a way, where all team members are equal. Agile promotes iterative and adaptive thinking by splitting larger projects into small increments in order to get more flexibility and continuous improvement. Collocated small cross-functional teams, rapid decision making, regular check-points to find solutions for appeared problems and sharing information between all stakeholders are important parts of agile development.

Agile software development methods allow companies to develop new products, evaluate and learn from these much faster [12]. Agile approaches are focusing on the speed in software development and specifically, on the speed at which a minimum viable product (MVP) can be created. This approach allows fintech companies to launch new products to market with less features, but at the same time to get immediate feedback from the customers. That allows fintech companies to focus on the best solutions for the customers and not on the internal development processes [13].

In software development, the changing requirements is one of the main challenges. Agile development methods are accepting the reality of change rather than search for complete specification. In many cases, accepting changes can cost less than preparing perfect requirements and guaranteeing that these will not change [14].

The first versions of agile software development started to evolve in the 1990s, when software developers started to look for alternatives to dominating waterfall methods that were considered as too structured, regulated, planned and micro-managed. The objective of agile approach was to fasten the software development process. The following agile methods were evolved in the 1990s:

- 1991: Rapid Application Development (RAD);
- 1992: Adaptive Software Development (ASD);
- 1994: Unified Process (UP), Dynamic Systems Development Method (DSDM);
- 1995: Scrum;
- 1996: Crystal Clear, Extreme Programming (XP);
- 1997: Feature Driven Development (FDD) [15], Whitewater Interactive System Development with Object modules (WISDOM) [16].

In 2001, 17 experienced software developers met to discuss the agile software development methods including the above mentioned and wrote down the Manifesto for Agile Software Development. Based on their practice, they outlined four values of agile software development:

- individuals and interactions over processes and tools;
- working software over comprehensive documentation;
- customer collaboration over contract negotiation;
- responding to change over following a plan [17].

In the main principles of agile software development, the continuous delivery of software is put into priority to satisfy customer expectations. The working software should be delivered as frequently as possible. The agile project should be built around and cooperated daily between business people and developers, who share the same location. Although the primary measure of the development process is working software, the changes in requirements are welcome even in the last stages of development. This allows the project to maintain a constant pace. In agile development, continuous attention is paid to technical excellence and design, keeping things simple at the same time. From the team's perspective, the agile development team should be self-organizing by reflecting regularly, how to become more efficient [18].

The following paragraphs describe in more details the agile methods that are mainly used since the mid-1990s until today. According to the literature, the two most commonly and widely used agile methods are Scrum and Extreme Programming [19]. Furthermore, two other agile methods, Lean Software Development and Kanban, are briefly introduced in the following paragraphs as the components of these methods are often combined to basic agile methods.

2.2.1. Scrum

Scrum is the leading agile development method used around the world. Scrum is used by 56% of software development companies [19]. The term "scrum" is borrowed from a study published in 1986 in the Harvard Business Review, where high-performing and cross-functional teams are compared to the scrum formation used by Rugby teams. In software development, Scrum is a lightweight agile project management framework for managing the development of a product. Scrum describes an iterative and incremental approach to development process [15].

The Scrum framework defines the general guidelines with rules, roles, artifacts and events, which are all important for a successful usage of the framework. The main components of the Scrum framework are the Scrum Team, the Scrum Events and the Scrum Artifacts [15].

The Scrum Team consist of Product Owner, Development Team and Scrum Master. In Scrum projects, the teams are self-organizing and cross-functional. Self-organizing teams find the best way how to accomplish their work and do not expect directing from outside. Cross-functional teams have all competencies inside the team and do not need functions from outside. Both these values are driving the teams to be more flexible, creative and productive. The objective of the Product Owner is to maximize the value of the product. The Product Owner is responsible for managing the Product Backlog and the entire organization must respect the decisions made by the Product Owner. The Development Team is responsible for delivering a potentially releasable Increment of "Done". The Scrum Master is a servant-leader to the Scrum Team, who is responsible for promoting the Scrum and helping all Team members to understand Scrum theory and practices. The objective of the Scrum Master is to maximize the value created by the Scrum Team to the whole organization [20].

The Scrum Events consist of Sprint and Sprint Planning, Daily Scrum, Sprint Review and Sprint Retrospective, which are all part of the Sprint. The Sprint is the heart of Scrum. The Sprint is a time-box up to one month, during which a potentially releasable Increment of "Done" is created. After the end of one Sprint, the next Sprint starts immediately. The Sprint Planning is the event, where the work to be performed in the Sprint is planned by the whole Scrum Team. Sprint Planning is time-boxed up to eight hours for a one month Sprint. The Daily Scrum is a meeting event for the Development Team that is time-boxed to 15-minutes in every day. At that meeting, the Development Team plans its activities for the next 24 hours period. The Sprint Review is the event held at the end of the Sprint to inspect the Increment and adapt the Product Backlog. This meeting is time-boxed up to four hours for a one month Sprint. The

Sprint Retrospective is an event, where the Scrum Team inspects itself and makes plan for improvement in the next Sprint. The Sprint Retrospective is after the Sprint Review and before the next Sprint Planning. The meeting is time-boxed up to three hours for a one month Sprint [20].

The Scrum Artifacts consist of Product Backlog, Sprint Backlog and Increment. The Product Backlog is a list of orders that the product needs. The Product Backlog is never complete. It is the best understanding of product requirements that the Product Owner is having at that moment. Therefore, the list is dynamic and changes constantly to reflect the current understanding of the product. The Product Backlog exists as long as the product itself exists. The Sprint Backlog is the set of items selected from the Product Backlog for the Sprint. The Increment is the sum of all items from the Product Backlog completed during a Sprint [20].

2.2.2. Extreme Programming

Extreme Programming (XP) is another agile development method that is becoming more widely used at many companies of different sizes and industries around the world. If Scrum is focusing more on project management, then XP is focusing more on implementation and delivering of software. The objective of XP is to deliver high quality product which quickly responds to the changes in customer requirements. XP concentrates more on customer satisfaction by delivering the software as fast as possible and responding to customer requirements even at a very late stage of development [21].

In XP team, Customer, Developers and Manager are all equal partners. The role of the Customer is very similar to the role of the Product Owner in Scrum. In XP, it is required that all team members work at the same place as one single team. In XP, the development is done in very short cycles. The XP development process differentiates from all other agile methods as it involves pair programming, where two developers are working at the same computer reviewing the code of the other developer [15].

The only two planning events in XP are Release Planning and Iteration Planning. Similar to Scrum, the Daily Stand Up meetings are used also in XP [21].

The XP practices include Test Driven Development, Customer Testing, Continuous Integration, Small Releases, Pair Programming and Refactoring. It is common that organization using Scrum have also integrated the XP practices like Test Driven Development or Refactoring into their workflows [21].

2.2.3. Lean Software Development

Lean Software Development (LSD) is a software development approach that shares the values of agile development. The goal of LSD is to achieve the maximum value from production by reducing waste from the production process. In LSD, everything and every step in development is scrutinized. The LSD has grown up from lean manufacturing principles and is adapted from the Toyota Production System [22]. The principles of LSD are summarized as following.

- eliminate waste;
- amplify learning;
- decide as late as possible;
- deliver as fast as possible;
- empower the team;
- build integrity in;
- see the whole [23].

To eliminate waste, any unnecessary activity should be eliminated or bypassed in the development process. For that, a value stream mapping technique is used. LSD emphasizes the learning process based on iterations when writing code. The learning process is sped up by complementing the development cycles with refactoring and integration testing. Better results are received, when the decisions are made as late as possible to take into account all facts becoming available by that moment. As fast as possible delivery is important to deliver the working software to the customer. For that, the tools like presenting cards or stories by the customer, estimating the time needed for the implementation of each story and giving overview of the development process in daily stand up meetings are used. To empower the team, in LSD, the "Work-Out technique" is used, where the roles of manager and developers are turned. As a principle, in LSD, the developers should have direct access to the customer and, therefore, the role of the team leader is focusing more on providing support in team communication and keeping up the motivation at difficult moments. In building integrity, LSD supports refactoring, restructuring the existing code without its external behavior. As the software development consists of small iterations, it is important to keep in mind also the whole picture and objective of the software under development [22].

2.2.4. Kanban

Kanban was developed as a subcomponent of the Toyota Production System and was a part of the Lean and Just in Time manufacturing processes. In agile software development, Kanban is considered as a lean approach. It follows many LSD principles like dividing the development process into smaller iterations and letting the teams to be self-organized. Different from any other agile method, the workflow in Kanban is visualized. The work is broken into small items, written on cards and stuck to the board. The basic and most simple division of the board is "waiting", "work in progress" and "completed work", or in even more simple and general way "to do", "doing" and "done" [23]. More complex divisions may include separate divisions for different stages of analysis, development and testing. In addition to breaking the work into smaller items and visualizing their status, another important objective is to strictly limit the works in progress at any one time. That technique is called Limit Work in Progress. In Kanban, every task is tracked and optimized to make the whole process more efficient and predictable. The average time that every task is taking is measured [24].

2.3. Scaling Agile Frameworks

The increasing number of teams need increased transparency, flow efficiency, learning and layered-in practices such as cadence and synchronization, managing work in progress and collaboration in solving the biggest problems. With the increasing number of teams, more synchronization between related information systems and teams is required. Managing work in progress efficiently becomes even more important as the number of teams is increasing. The flow management and work prioritization need to be properly assigned to the right roles. Scaling agile can get stressful for teams when tracking dependencies and trying to understand their role in the big picture. Therefore, it is important to increase the collaboration across the whole organization [25].

The scaling agile frameworks became important as in development each team was efficient in agile, but across the teams agile was not really working. The scaling agile frameworks mainly cover the aspects of organizing the work between multiple teams in collaboration, synchronization of information and managing work in process. There are nearly ten different frameworks that scale agile. The main scaling agile frameworks that have evolved in 2000s are the following:

- 2011: Scaled Agile Framework (SAFe);
- 2012: Disciplined Agile Delivery (DAD);
- 2013: Large-Scale Scrum (LeSS);
- 2015: Nexus [26].

On one hand, these frameworks seem to be similar as all of them are focusing on solving the problems in large projects. On the other hand, the differences between the frameworks are coming from the methods and practices adopted, organizational type, team size, training, certification and technical practices required [26]. Some frameworks like SAFe have a lot of precise material about organizing the processes and other like Nexus very little. The following paragraphs describe in more details the two most commonly referred and used frameworks: Scale Agile Framework and Large-Scale Scrum [25]. Disciplined Agile Delivery and Nexus are not covered in this thesis as these are relatively new frameworks, used less than the first two and lack reliable information from the practice.

2.3.1. Scaled Agile Framework

Scaled Agile Framework (SAFe) was first introduced in 2011. It is a framework that captures the organization's know-how and workflow to scale agile and lean practices. SAFe promotes alignment, collaboration and delivery across large numbers of agile teams.

The main challenges of scaling agile principles and practices that SAFe is trying to solve are coping with longer planning horizons, keeping agile at abstract levels of responsibility, dealing with delegated authority, synchronizing deliverables and allowing time for innovation and planning [26].

SAFe handles the issues on four levels: Agile Team, Program, Portfolio and Value Stream. All SAFe teams are Agile Teams, but the teams may have different functions such as architecture team, development team and systems team. Agile Teams consist of five to nine people, who usually work on two-week sprints using XP methods. The Program is a group of five to ten Agile Teams that form Agile Release Train that works on a larger Program Increment. Program Increment occurs on a rhythm of 3-5 development iterations, followed by one Innovation and Planning Iteration. Teams and Programs can release functionality at any time, including continuous delivery. Portfolio is a set of Value Streams that are budgeted via lean-agile budgeting. Value Stream is a business process that captures the flow with each basic step used to deliver value from the first triggering action to the last action that ends the process [27].

2.3.2. Large-Scale Scrum

Large-Scale Scrum (LeSS) was introduced soon after SAFe, in 2013. It is a product development framework extending Scrum with scaling rules and guidelines. LeSS is quite often used in organizations with large-scale product development, especially in the finance and telecom businesses.

There are two versions to the framework. The thinner framework LeSS is for up to eight teams and the thicker and more complex framework LeSS Huge is designed for the organizations with tens of teams and hundreds of developers. LeSS follows the original purpose of Scrum and tries to use all the Scrum components, but in a more larger and scalable base [28]. The idea of LeSS is to descale the complexity of the organization and solve problems in a more simple way with less organizational structure, less management and less roles.

2.4. Financial Institutions

Financial institutions are companies dealing with monetary transactions like payments, deposits, credits, investments, insurance, etc. Typical financial institutions operating in the financial services sector are banks, credit unions, investment companies, brokerages, asset management companies and insurance companies [29]. The business range of financial institutions is usually broad. Practically everyone has a need for the services offered by the financial institutions.

The most complex financial institutions is a full service universal bank offering a wide range of services to private, corporate and institutional clients. The services of a bank can be divided into four main groups: transactional services, deposits, credits and investments. Transactional services include payments, currency exchange, card issuing, card acquiring and ATMs. Deposits include current account deposits and term deposits. Credits include corporate loans, mortgage, consumer finance and leasing. Investment services include brokerage, analysis, advising and portfolio management [29]. Such banks are typically having also subsidiary companies dealing with asset management and insurance.

The software development in banks is complex. In addition to developing and maintaining the above mentioned products and services, the software development is also needed for client service channels like internet and mobile banks, for support services like finance and risk management and, if the bank is operating in several countries, for multi-currency and multi-language setups. Moreover, many banks carry on the legacy systems, which need to be maintained and updated continuously.

Financial institutions are different from any other organization due of the following aspects:

- Data: Financial institutions are storing the financial data of their customers that has to be extremely precise at any time currently and historically. Financial institutions fall under high reporting standards, which means that they have to have data governance in place and guarantee the correctness of the data.
- Compliance: In financial institutions, increasingly more attention is paid to Know You Customer (KYC) and Anti-Money Laundering (AML) principles. Also, the activities of financial institutions are regulated by hundreds of laws and rules. Financial institutions need to have in place the four-eye principle for looking over the new code that is produced.
- Dependency: Most of the financial institutions depend on other third party service providers such as worldwide networks to send and receive information (such as SWIFT), worldwide networks for card payment services (such as VISA, MasterCard, American Express), regional payment service providers (such as EBA Clearing), regional card payment service providers (such as NETS), other correspondent banks, etc [30]. It means that quite often financial institutions can release new products and features only together with the third party partners and they have to take into account the specialties of the partners.
- Security: Financial institutions are storing the data with extremely high sensitivity and they are obliged to protect the customer's personal data from any other party. Financial institutions are also offering vital services to the community and they are expected to be up and running also in the extreme situations [30].
- Impact: Financial institutions have high number customers, who are using banking services on every day. Therefore, financial institutions have to keep up and running all the infrastructure 24/7, often also real-time like for payments and card transactions [30]. Every change that financial institutions are making in their information systems is impacting high number of customers.

- Complexity: Compared to fintech companies, financial institutions are having much broader choice of products [31]. That increases the complexity of managing and developing products and information systems used for different products.
- Vulnerability: Financial institutions are vulnerable to the challenges that are raised by fintech companies. Financial institutions are becoming more digital service companies as their customers prefer to bank on-line and not so much in the branches anymore. Financial institutions are spending increasingly more of their annual budget to software development and IT in general. At the same time, financial institutions are having their legacy systems that do not allow to launch new products or product features as fast as fintech companies are able to do. Therefore, fintech companies have better position in disrupting one or another specific service that was so far offered only by financial institutions [31].

In today's customer centric and digital world, new financial products created by fintech companies are challenging large financial institutions at high rate. As the fintech companies are less regulated due to the reason that they are not taking deposits from the clients, they are better positioned in not following all constraints that large financial institutions have to follow. With less regulation, there is less compliance risk. With smaller business, there is less to loose with reputation risk. If there is a loss of data or breach of security in a fintech company, it will affect only a small part of the financial market, but if the same thing is happening to the financial institution, they might lose the trust at all and disappear from the market with huge impact to the financial market.

3. Software Process. LHV Bank Case

This part of the thesis describes the current software process at LHV Bank. It also brings out the main process issues or bottlenecks in the current software process at LHV Bank. The current software process is the basis for changes proposed in the further parts of this thesis.

3.1. Current Software Process

The organizational structure of LHV Bank consists of four business divisions, three supporting divisions, two organization-wide divisions and one daughter company. The three business divisions are Retail Banking, Private Banking, Corporate Banking and Financial Institutions. The three supporting divisions are Information Technology, Finance and Risk Management. The two organization-wide divisions are Human Resources and Marketing and Communication. The daughter company LHV Finance is for Consumer Finance. In total, 300 people are employed by LHV Bank.

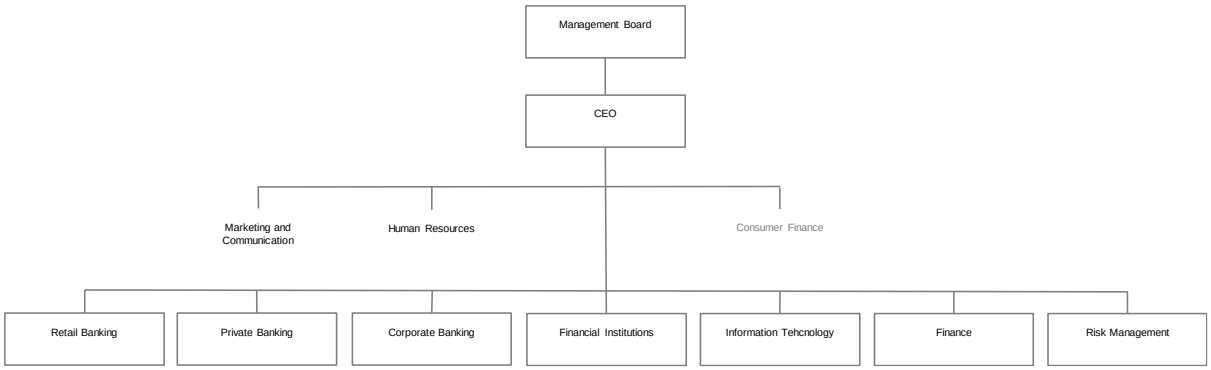

Chart 1: Organizational structure at LHV Bank.

In addition to Sales and Customer Support, the Retail Banking is the main division for product development. There are five product units in the Retail Banking: Transactions, Investments, Credit, Leasing and Digital Channels.

Some products are split into sub-products, called accounts. The following table describes the division of products and accounts between the divisions and departments at LHV Bank.

Division	Department	Product	Account
Retail Banking	Sales and Customer Support	Client Service	Client Service
			User Rights
		Gold Client	-
	Transactions	Cards Issuing	Cards Issuing
			Cards Credit Limit
			Cobranded Cards
		Cards Acquiring ATM Banklink Connect	-
	Investments	Brokerage	-
	Credit	Credit	Mortgage
			Student Credit
			Micro Credit
			SME Credit
	Leasing	Leasing	Leasing

			Leasing Core
	Digital Channels	Digital Channels	Digital Channels
			Financial Portal
Private Banking	Portfolio Management	Portfolio Management	-
Corporate Banking	Corporate Credit	Corporate Credit	Credit
			Overdraft
			Guarantee
			Collateral
			Trade Finance
Financial Institutions	Payments	Payments	Payments
			E-Invoices
Finance	Treasury	Deposit Treasury	-
	Accounting	Accounting	-
	Data Warehouse	Data Warehouse	-
	Back Office	Back Office	Back Office
			Bouncer
Risk	Risk Control	Risk Control	-
	Compliance	Compliance	-
	Anti-Money Laundering	Anti-Money Laundering	-
	Debt Management	Debt Management	-
LHV Finance	Consumer Finance	Consumer Finance	Consumer Finance
			Consumer Finance Core

Table 1: Division of products between the divisions and departments at LHV Bank.

In total, there are 24 products at LHV Bank. Every product has its Product Owner (PO). Some POs have more than one product to manage. The role of PO is similar to the role that PO is having in Scrum Team. PO is the owner of product vision, product roadmap, product backlog and product metrics. PO knows the customer, market and competition of the product. PO manages the stakeholders of the product and the risks related to the product. PO is responsible for the sales and profit/loss of the product. PO has to understand technology space, both current and future, related to the product.

The Information Technology division of LHV Bank is divided into two departments: development and operations. The development department is managed by Head of IT Development. There and seven development teams and one R&D initiative team. The development team is led by Software Development Unit Manager (SDUM). Usually one SDUM is leading two teams. The development teams consist of 8-12 people. The teams usually consist of one Analyst, one Lead Software Engineer, 3-6 Software Engineers and 1-3 Quality Assurance Engineers. The operations department is managed by Head of IT Operations. In total, over 70 people are employed in the Information Technology division. In addition to that, the Digital Channels team is working under Retail Banking division, the Data Warehouse team under Finance division, the Information Security team under Risk Management division and the HR specialists dedicated to IT under Human Resources division.

As there are 24 products and only 7 development teams, the products are divided between the teams. Some products are divided between several teams as these products belong to supportive functions of the bank and are covering many principal products of the bank. The following table describes the division of products between the development teams at LHV Bank.

Team	Own Products	Shared Products
Team 1	Deposits Payments Cards Issuing Banklink Treasury	Accounting Data Warehouse Back Office Risk Control Compliance Anti-Money Laundering
Team 2		Accounting Data Warehouse Back Office Risk Control Compliance Anti-Money Laundering
Team 3	Cards Acquiring ATM Connect	
Team 4	Credit Corporate Credit Debt Management	
Team 5	Leasing Consumer Finance	
Team 6	Client Services Gold Client Digital Channels	Data Warehouse Back Office Compliance Anti-Money Laundering
Team 7	Brokerage Portfolio Management	Data Warehouse Back Office Compliance

Table 2: Division of products between the development teams at LHV Bank.

The product development is starting from the product vision created by PO and getting the approval to that from the Management Board. As PO is responsible of planning the annual product Roadmap in co-operation with SDUM, the large works called Epics are included to the Roadmap. The Roadmap prioritization takes place on monthly basis for the next three months at the Epics level on Monthly Planning Meetings. As the development teams are having more than one product, the POs of different products have to agree on the priorities of Epics in the Roadmap on mutual understanding and agreement. The products and Epics with better KPIs are usually getting higher priority. The products supporting the strategic goals on the company's level are getting higher priorities. All Epics with strict deadlines coming from law or from any other unseen reason will get the highest priority.

Every Epic in Roadmap is getting a rough size estimation for resource planning and prioritization. Later on, in the development process, the time used for different tasks is precisely monitored and reported. On Monthly Planning Meetings, the ways to improve the processes are evaluated. Analysis is done on Epic level.

When the Epic is described, the SDUM and Analyst will take it over and divide the Epic into smaller tasks. PO is connected to the development team through SDUM, although in a relatively small organization, all other connections are also allowed and happening between all team members. As the teams are having daily standups, PO has always the possibility to participate in the standups. The development is done by the team within designated resource window in the Roadmap.

In addition to Epics, the teams are having Weekly Backlogs for smaller tasks and bug fixes to avoid the situation, where the whole team is dealing with one large Epic for months and nothing else is developed. The tasks in the Weekly Backlog are also described and prioritized by PO.

In the development process, everything starts from proper analysis of the task performed by Analyst. After Analyst has described the task in very detailed manner, Engineers are taking the tasks into development. When the development is done, the Quality Assurance Engineers are performing both manual and regression tests for the task. If there are bugs found, the task will go back to the Engineers. If there are no bugs, the code will go to the release process. In the release process, all changes to the code are gathered and released once a week. In the release process, both Lead Engineer and System Administrator from the IT Operations are involved.

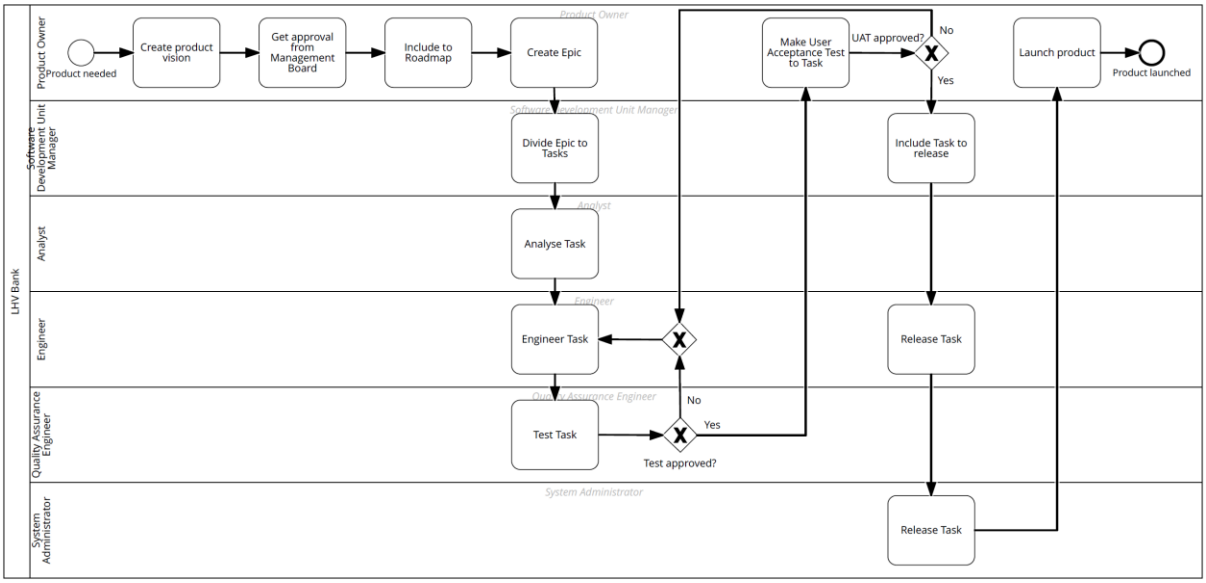

Chart 2: General product development process at LHV Bank.

The setup of the teams and work processes follows Scrum method in large extent, but it has borrowed the elements from XP and Lean Software Development methods as well. The bank has relatively strong visualization culture, where all teams are keeping track on the tasks using the visualization principles from Kanban.

The information system in LHV Bank consists of nearly 40 components that can be grouped into 13 larger clusters. The first core system of bank accounts was developed in the beginning of 2000-s. This core system has remained as the basis for most of the main banking services. Although the new modules of the system have already been built separately, the core infrastructure is playing a large and important role in the banks infrastructure especially for accounts, deposits and payments. Other larger components are built later for cards issuing, cards acquiring, portfolio management, credit decision, credit, leasing, consumer finance, debt management, etc. The core system of the bank is written using Adobe ColdFusion and new components using Java.

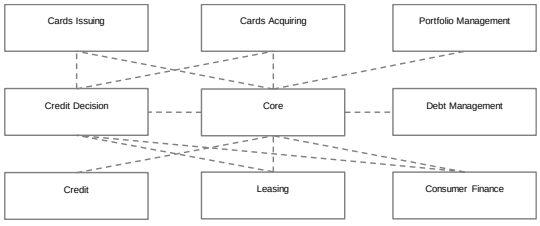

Chart 3: Extract of the main components of the information system and relationships between these at LHV Bank.

3.2. Process Issues

There are several process issues in the current software process. From product development perspective, one of the main bottlenecks is the situation, where the bank is having as much as 24 very different products and only 7 development teams. As there are situations, where the products belonging to the same development team have different POs, a lot of time is spent on discussing and agreeing on development priorities between POs. Often it means that the priorities in the Roadmap have to be agreed in the Steering Committee or between the members of the Management Board, who are the managers of POs. In such situations, as the PO is having only the vision of the product or new feature, it is difficult to argue between the parties about the necessity of one or another product to make the decision. To overcome that bottleneck, the number of products has to be decreased or the products have to be consolidated under fewer POs. The option of increasing the number of development teams is theoretically also an option, but in real life the budget is currently restricting to increase the headcount significantly.

From the development perspective, the development is usually done in quite an agile manner, although very clearly separated roles between SDUM, Analyst, Engineer and Quality Assurance Engineer is causing the situation, where the development process is split into small steps, where everyone has their own concrete role that can be done just after the pervious step has been finished.

From the delivery perspective, the development teams are not able to deliver the product or feature from the beginning until the end of the process. The whole delivery process is very time consuming as the releases are done by other people and both development and operations are involved in delivery process. The release process in general is old fashioned and a true bottleneck causing unsmooth end in the development process. Also, in case of bugs, the same process is went through and the development teams do not have the possibility to fix the bug by their selves.

From the system architecture perspective, some older components of the bank’s information system are relatively large and not enough modular to enable different teams to work on the same component at the same time.

4. Analysis and Findings

This part of the thesis analyses the agile methods used in SPI through the literature analysis and the agile practices used in SPI through the conducted interviews. Based on these analyses, the findings are brought out and discussed at the end of the chapter.

In this thesis, both literature review and interviews with companies practicing agile software development were carried out. The interviews were carried out in addition to the literature review due to two main reasons. Firstly, there is not much literature that covers the SPI using agile methods in large financial institutions nor in fintech companies, and secondly, the interviews give the latest information about the trends in using agile methods and components in the leading large financial institutions and fintech companies.

4.1. Agile Methods in Software Process Improvement

The literature covering agile methods for the SPI can be divided into three parts. The first part of the literature covers general issues of applying agile methods in software development in different organizations with the focus on combining various components from different agile methods. The second part of the literature covers scaling agile in different organizations. The third part of the literature covers the analyses and case studies of agile methods applied specifically in the financial institutions. However, there are only few studies that are covering the usage of agile methods in large financial institutions and even these is usually done as a part of some larger study.

Finding 1: Agile is about the mind-set of the whole organization.

In general, the idea of agile is to break things into smaller increments to guarantee that the value is delivered each time [32]. Before moving to the detailed components used in the agile software development process, it is important to understand that the agile development principle has to cover the whole team including the product owner, i.e. the customer.

The first general principle in moving from waterfall to agile is to bring the customer very close to the rest of the parties involved in the development process. In this process the interaction between different parties involved is improved significantly. The focus is shifting from checking the tasks and boxes more to serving the customer and getting the right things done for the customer [33]. The second general principle is to focus the whole team more on people interactions instead of processes. People with different skills are working together to deliver product features. It is crucial to guarantee that the team members are actually collaborating with each other, they enjoy what they are doing and feel like they are accomplishing their mission [34]. The third general principle is welcoming change. It is very important to react to the changes and even failures, when something went wrong. The team has to learn from each sprint and integrate that learning into the next iteration. It is natural that in quick sprints the focus is on delivering something and the customer is obliged to give feedback on what was delivered [35]. Dedicated agile teams have a clear objective and a single purpose with the customer, whose task it to guide the team along.

The basic organizational unit in agile development is a small team, which typically consists of seven, plus or minus two people. The team includes the functions such as analysis, development and quality assurance [36]. From time to time, it is necessary to move people between the teams to encourage cross-fertilization of ideas and to guarantee that the teams do not become isolated and diverged from the other teams. At the same time, from the people management perspective, it is also important not to move people too frequently to avoid that the teams fail to develop into highly productive units. It is important to understand that physical location of teams is much more important with agile methods than with waterfall methods [36]. The size of the

teams has to be kept more or less stable as adding people to a team increases the total effort necessary in three ways: the work and disruption of repartitioning itself, training the new people and added communication [37].

In 2015, a research carried out among the financial institutions in Kosovo concluded that the success of implementing agile depends on the structure of the organization and culture. Agile is about the mind-set of the whole organization. The financial institutions that promote collaboration and cultivation culture are in a better standing and accept more easily the transformation from waterfall to agile. In addition to that, two other attributes from culture, control and competence, are needed for successful transformation [38].

Finding 2: Different agile methods and components should be combined.

Early thinking in agile software development focused very much on detailed activities of code creation and deployment, complemented by small and self-contained teams. Over time, it has become clear that the large proportion of software development problems are caused by the poor process management in general. Software development processes can be described in terms of queues and control loops, and managed accordingly [34]. The agile software development is not anymore only about the work division inside a small team, but rather managing the whole development process. The full benefits of agile can be achieved only with engaging management and business people. The business people have to have the appetite for working in a new agile way [36].

The question becomes more important, when there are many teams and their work needs to be organized at the same time. On one hand, large organizations need to have more development resources and teams to fulfil their needs. A single pipeline could carry only one pipeline's worth of capacity. On the other hand, more teams are needed also to guarantee the delivery of the product in larger projects. If all work would move through a single pipeline, then a stall in that pipeline could disrupt everything. The penalty from a stall is reduced with more than one pipeline [34]. The more teams and pipelines the organization is having the more complex the whole agile development becomes. It is widely discussed and agreed that Scrum has helped agile software development teams organize and become more efficient. In addition to that, lean methods like Kanban are extending these benefits. Kanban is a powerful tool to identify process improvement opportunities [34]. It is even argued that it is difficult to manage Scrum without Kanban. Many Scrum implementations suffer from all the same problems that traditional project management or waterfall software development methods are having. Therefore, adopting Kanban is an appropriate way to help Scrum deployment evolve [32]. Therefore, in the related literature, the term Scrumban appears more often in the recent years. Scrumban applies Kanban systems within Scrum context, enabling team members, teams and organizations in general to better understand, manage and coordinate their work [32].

In 2016, the Scrumban concept was efficiently implemented also in a large bank in USA. Although the bank was in agile transformation already for years, it still struggled on slow and unreliable delivery of work. The reason for that was that the bank focused on improving the individual components of the delivery process, but not the entire system [32].

Finding 3: Methodology training enables organizations to develop know-how and prepare better for the implementation of the agile methods.

There are several factors that impact the success of implementing agile software development in the organization. Training on the methodology, access to external resources, active management support and involvement and company size all significantly impact the implementation of agile methods [16].

Organizations that provide methodology training to the teams are having more successful implementation of the methodology than organizations that do not provide that. Methodology training enables organizations to develop know-how and prepare better for the implementation of the methodology. Access to external resources such as books, journals, consultants and attendance at conferences also increase the likelihood that the project of taking agile into use will be successful. As in any other project, the support and involvement from the management level improves the success of every business project. The most important factor is the size of the organization. There larger it is the more complex it becomes to implement and later manage the agile software development [16]. Therefore, in the related literature, the concept of Scaling Agile is covered from many angles.

However, the agile development framework has to be organization-specific to take into account the current stage and work culture of the organization. In practice, a successful implementation of agile methods needs to be carried out by dedicated professionals. The cross-functional teams have to be established in order to support a common understanding across organizational boundaries [39]. Becoming agile too fast may turn the whole organization into a chaotic one [40].

Improving organizational structure at the same time when trying to follow the guidelines of agile development, is in several cases contradicting. Therefore, finding the right balance between these is the key for successful agile development. Four organizational factors are the most important, when scaling the agile projects: organizational structure, decision making, collaboration and coordination and agile culture [41].

In 2014, the large Scandinavian bank Nordea decided to renew their digital banking platform and use agile development methodology to achieve that. As the organization and project were large enough, Nordea decided to use SAFe framework for that purpose. In total, 80 people were trained and the Agile Release Train formed of them consisting of five Agile Teams of development. The bank itself has concluded that proper training was vitally important to start with the project in the right way. The Program Increment, a 10-week cycle, consisted of a release planning session, four development iterations, an innovation iteration and a planning iteration. In Program Increment sessions, they identified dependencies, planned the overcomes and broke down the features. The bank stated that the delivery system improved significantly. The main improvements came from identifying the dependencies between teams during the Program Increment sessions [42].

Finding 4: Following agile principles in full brings the success in SPI.

During 2004-2007, a large financial institution KeyCorp made the transition from waterfall to agile software development. One of the main changes among many was that so far existed project managers were turning into Scrum Masters. Although the roles were similar, the difference was in perception of the role of that function. The development team did not see the project manager as a person, who delivers heavy documentation and process requirements, but rather as a team member of their own [43].

In 2009, the large Australian financial institution Suncorp implemented a major system replacement using agile iterative method. Until that time, Suncorp was mainly using waterfall method and agile method was used predominantly in small niche developments or those that involved external software companies. One of the main lessons learned in that project was that although unstable and changing requirements are one of the key benefits of using agile methods, agile cannot be seen as a way to increase the scope of the project without impacting the time, quality and budget. All changes in the requirements should be still managed through

the backlog and the impact should be continuously discussed with the team and all stakeholders, whose main objective is to prioritize the tasks [44].

In 2010, a Danish mid-sized bank Jyske Bank experienced a problem in their distributed and co-located agile projects that was connected to the mantra that responding to the changes is important than to follow the plan. It appeared that in development, the planning is difficult and conflicting with the agile manifesto. The developers of the bank felt that preparing planning was like falling back into waterfall approach and they were tagged as old-fashioned. Although the agile development should remain agile and take into account possible change requests in the development process, all larger projects still need a high-level plan to manage the expectations of the stakeholders. Otherwise, the steering committees, external parties or other stakeholders cannot get the needed information about when and what will be delivered. Therefore, it is important to have a high-level planning in addition to the iterative planning in order to keep track on the development [45].

Agile methods do not pay enough attention to agile testing. But the reality is that testing is a crucial part in the whole software development process. It is sometimes said that the rule of thumb is that 1/3 of the team is scheduled for design, 1/6 for coding, 1/4 for component testing and 1/4 for system testing [37]. From here it is seen that testing could take even 50% of the development time. Therefore, it is important to start testing as early as possible and the testing should be run very frequently, even before every source code integration and definitely before every release. In agile development, automation is essential in testing [40]. Automated tests with relevant tools make more sense as these use less resources and time that is critical in the agile development. High skilled testers must test for hours or days and still their test results are less reliable than the automated tests. To avoid the problems that weak communication between developers and testers may rise, they both should work in the same open space area. Full integration of developers and testers is a productive choice. In agile development, the test plan cannot be fixed. It is important to modify the test plan adequately to the changes in requirements and problems appearing in development [40].

In summary, from the organizational point of view, the literature review brought up the findings that agile is about the mind-set of the whole organization, different agile methods and components should be combined organization-specifically and methodology training enables organizations to develop know-how and prepare better for the implementation of the agile methods. From the development perspective, the literature review brought up that following agile principles in full brings the success in SPI.

4.2. Agile Practices in Software Process Improvement

This part of the thesis is conducted and reported as a case study research. Case study is a suitable research methodology for software development as it studies modern phenomena in the natural context which is hard to study in isolation. The analytical literature review is not sufficient for investigating complex issues like the interaction between humans and technology. Although case study does not generate the same results as controlled experiments do, they still give better and deeper understanding about the phenomena under study [46].

Case study is different from controlled empirical and analytical studies. Therefore, the case study can be biased by the researcher, have less value and is difficult to generalize. However, the value received from the case study is adding great value to the analytical literature review [46].

Case studies can be divided into four categories taken into account the purpose of the research: exploratory, descriptive, explanatory and improving case studies. Exploratory case study finds out, what is happening in practice, seeks for new insights and generates ideas or hypotheses for

the research. Descriptive case study describes the current situation. Explanatory case study seeks for the explanation to the phenomena. Improving case study tries to solve or improve some aspects of the phenomena [46].

In this thesis, the exploratory case study method is used. The exploratory case study is usually used to investigate certain phenomena due to the lack of preliminary research and to clarify the research context. It is also used to explore a relatively new field of scientific investigation, where the research questions are not yet clearly identified or formulated [47].

In addition to the literature review, the exploratory case study interviews with relevant companies were carried out as part of the analysis. The aim of the interviews was to amend and validate the findings from the related literature and bring into scope the most up to date agile practices used in fast growing financial institutions and fintech companies.

The companies for the interviews were chosen from Estonia with the aim to cover the companies that offer financial with different size and in different development stage. The main criteria for choosing the companies were the following:

- the company is large enough, well-known and with strong brand;
- the company attracts talented people for the whole development of the company and for the needed software development more specifically;
- the company must have its own in-house software development organization;
- the software used in the company should be developed in-house in most extent;
- the decisions taken in different stages of the development processes must be done by the company itself;
- the company is or has been lately in the fast growth stage;
- the company has entered other markets in the Baltics, UK, Europe or World in general.

The interviews were done with the following companies: Swedbank, Bigbank, TransferWise and Monese.

Swedbank is the leading full-service bank in Estonia offering all financial products to retail, corporate and institutional customers. Swedbank as a company is mature, being in the Estonian market already over 25 years. The company employs around 2 300 people of whom around 500 are working in IT. Historically, the software developed in Estonia has been implemented also in their other sister companies the Baltics. Today, the development is more divided over the group.

Bigbank is one of the leading consumer finance banks in Estonia. The company was founded already 25 years ago, but as a bank started to operate and grow faster just 12 years ago. Bigbank has expanded into other markets in the Baltics and Europe such as Finland, Sweden and Spain. The company employs around 450 people of whom around 100 are working in IT.

TransferWise is the leading fintech company offering low-cost cross-border money transfers, which has expanded to the UK and the worldwide. The company was founded 7 years ago and is currently one of the most popular money transfer apps in the World. The company employs around 1200 people of whom around 240 are working in IT.

Monese is the leading fintech company offering simple bank account and card services to unbanked people, which has expanded to the UK and Europe. Although the company was founded 5 years ago, it launched its first services just 3 years ago. The company is in the early stage of development, but has attracted already high number of customers. The company employs around 100 people of whom around 30 are working in IT.

Profile of the companies interviewed is summarized in the following table.

Company	Main Markets	Current Activity	Number of People	Number of People at IT	Stage of Development
Swedbank	Baltics	27 years	2 300	500	Matured
Bigbank	Baltics, Europe	12 years	450	100	Growth
TransferWise	UK, Worldwide	7 years	1 200	240	Fast Growth
Monese	UK, Europe	3 years	100	30	Early Stage

Table 3: Profile of the companies interviewed.

The people interviewed in the companies were the CTO's, VP's of Engineering and Engineering Leads, who have worked in the companies since the beginning or at least over 10 years.

The companies that participated in the interviews allowed the author of the thesis to summarize the findings, but not present the processes and practices in the way that it might be understandable on the individual company level. The findings brought out are common to all companies interviewed. The findings that were relevant only for one company are usually not included, although in a few places these are mentioned as they seem to be generalizable as well.

It can be concluded that the companies that have started in the last five years, are using agile practices since the beginning, but the companies with longer history, have started to take agile practices into use more systemically just in the last 3-5 years. The main findings from the interviews can be summarized as following.

Finding 5: Modular system architecture and microservices are the prerequisites for applying the agile practices.

The system architecture was pointed out by each company as the main success factor that is determining, whether the agile practices can be used in the software development or not. If the system architecture is monolith and a large part of the system is built as a single system, it is very difficult to apply the agile practices. Therefore, the companies are paying very much attention to have a system architecture, where the system composes of smaller modules and where the microservices are used as the main software development technique.

In microservices, the application programs are structured as a collection of services, where the coupling between software modules is loose. Decomposing the application programs into smaller services improves the modularity and understanding of the system. It also makes easier and faster the development and testing. Microservices are used to increase deployability (deployment independency, shorter deployment time, simpler deployment procedures, zero downtime), modifiability (shorter cycle time, incremental quality attributes changes, easier technology selection changes, easier language and library upgrades) and resilience to architecture erosion [48]. In case of microservices, the data is kept in smaller parts and the services are more scalable.

Microservices are often used to ensure Continuous Delivery (CD). CD is a software development approach, where the software is produced in short cycles to ensure that the software can be released at any time. A straightforward and repeatable process in CD helps to release software with higher speed and frequency [49].

While all of the companies said that their new systems are built up using microservices, three of them admitted that they are still having also the monolith legacy platform in the company and two of them are in the process of completely changing or splitting it into smaller parts.

Finding 6: Common objectives of team members are important for managing and measuring the results of the teams.

More clearly than the literature review, the interviews showed that common objectives of team members are utmost important. The teams should be fully responsible for their product, both from business and development perspective. In a typical setup of the fintech company, the products or features are divided into teams that share common customer support and operations divisions. Such setup allows the teams to set their own business objectives, including not only the development of the product, but also the sales targets.

For setting the objectives and measuring the outcomes, Objectives and Key Results (OKR) management tool was used in one company. The OKR framework was first introduced by Intel and then brought to Google, from where it has reached to LinkedIn, Twitter, Uber and many other companies. OKR's are usually set at the company, team and personal levels with the aim to increase the visibility of goals inside the organization [50].

Using one or another management tool to set the objectives of the teams, it serves as a perfect tool to measure the results of the teams. The question is not in the speed of the development by developers anymore, but the question is, whether the team is producing the right product and achieves its business goals. In fintech companies, all team members are measured by reaching the same business objectives. That is a crucial factor to merge individual team members into a single team that is working in the name of the same objective.

Finding 7: Cross-functional teams are having all skills to cover the full development cycle of a product.

From agile development methods, all companies said that they are using Scrum, but it has been modified according to their real needs. The teams are usually consisting of Product Owner, 4-8 Developers/Quality Analysts and 1-3 other positions necessary exactly in the development of a specific product. Other positions may include Designer, Partner Relationship Manager, Data Scientist, Customer Support Manager, etc. In addition to these positions, larger and more mature companies are having Scrum Master position (sometimes named as Project Manager or Product Engineering Manager), but it is important to note that in many smaller companies, the teams are not having a separate Scrum Master position, but these tasks are given over to the Product Owner or Lead Developer. Scrum Events like Sprint Planning, Daily Scrum, Sprint Review and Spring Retrospective are used according to the theory in most companies. Also, Scrum Artifacts like Product Backlog, Sprint Backlog and Increment are used according to the theory in most companies.

If more mature teams are using Scrum, then less experienced teams are sometimes allowed to use just Kanban. However, most of the companies admitted, that on company level or in Scrum teams, they are also using the elements from Kanban to track and visualize the current tasks in process and to limit the work in progress. From XP, the main element that all companies are using is Daily Stand Ups.

The main idea behind the cross-functional teams is to guarantee that the team can deliver the product from the very beginning until the launching. They have to take into account the input from customer support, compliance, security, etc., but the product itself is delivered by the team and nobody else. Cross-functional team means that the team as a whole is having all skills to develop a product [24].

Finding 8: Responsiveness to changes culture is the key element in agile development.

In the interviews, all companies repeatedly pointed out that the agile development should take into account the changes happening in the process of developing a product. When launching a new product, the companies are using the minimum viable product (MVP) concept.

A MVP is a product that has only the most important features that solves the problem for the customer. A MVP gives early feedback about the product directly from the customer and is a valuable input to developing the product further. A MVP helps to begin the process of learning as quickly as possible. If the MVP is satisfying the customer, it can be developed further, but if not, the product can be changed quickly or in some cases even pivoted or killed [51]. From most of the interviews, it came out that as financial services are regulated more than any other service, the MVP in the first stage (Alpha) is usually given into use internally to the limited or whole staff of the company. Just after that, the MVP in the second stage (Beta) is given into use to the limited or whole clientele. If the MVP has survived the Beta stage, it can be launched as a new product for customers.

It is important to build up the culture of being responsible for changes. All companies interviewed brought out that the key factor here is constant exchange of information. The companies are often having quarterly or monthly meeting, where the management is giving the business overview and directions, and where the teams are giving the overview of their activities or products and product features under development. The culture of closing the products or cleaning the product backlogs is strongly supported in the companies.

All of the companies are using the Atlassian tools like Jira Software for planning, tracking and supporting, Bamboo for coding, building and shipping, and Confluence for collaborating and chatting. The most commonly used solutions for daily communication are Slack and Zoom.

Finding 9: Automated testing fastens the process in agile development.

The interviewed companies are paying much attention, how to reduce the time and resources spent on quality assurance of the code. In case of fintech companies, it came out that these companies have even merged the developer and quality assurance positions and they rely on the testing that is done by the developers their selves. On the other hand, the financial institutions brought up the reason for having a separate quality assurance position to fulfil the regulatory requirement of reviewing the code on four eyes principle.

In any case, all companies stressed the importance of having automated testing process in place. The regression tests were mentioned as the most common software testing type. In regression testing, the existing tests are re-run to ensure that previously developed and tested software is still performing in the same way after it is changed with other software.

In one company, also the Test Driven Development as a practice from XP is used. In Test Driven Development, the system requirements are described in very specific test cases and the software is developed to meet these test cases, only.

Finding 10: Autonomous release process gives the independence to teams.

All of the companies raised the issue of having an autonomous release process that gives the final independence to teams to launch the new products or features. At the same time, it came out that most of the companies have reached to that solution just recently. The previous central release process became the bottle neck for teams and the companies have just recently taken into use independent release processes to fasten the launch of development done inside the teams. For that, the companies have built their own solutions or bought some readymade orchestration software from the market.

Orchestration software helps to arrange, coordinate and manage the computer systems, middleware and services automatically. The objective of the orchestration is to streamline and optimize the frequent and repeatable processes. The orchestration can be used to optimize the process, it the process is repeatable and its tasks can be automated [52].

The idea of the autonomous release process is to give the team possibility and full authority to develop and launch the new product or features by their selves. The whole process becomes more faster, the team does not use and depend on other resources, and in case of bugs, the team can fix these faster and independently. Through that process, the teams can decide, whether the new feature is made available to all or part of the customers. Some of the companies are having the custom of launching the new features only in one country or region to guarantee that there are no bugs and the customers are taking the new feature into use as it was intended.

One company said that they are knowingly following the Canary release technique. In Canary release technique, the new software version is rolled out to a limited customer base with the aim to reduce the risk that the new version could bring to the entire infrastructure or all customers. In Canary release technique, when the new version is set up, some of the customers are routed to the new version. The routing can be done on random basis or selecting the customers by certain criteria based on their profile and demographics. If the new version is working for the selected customers, the number of customers routed to the new version can be increased until the whole customer base is using the new version. After that, the old version can be closed. Canary release is sometimes also referred as a phased rollout or an incremental rollout [53].

From the interviews, it can be concluded, that the trend in releasing is towards smaller and more frequent releases done at any time with no downtime by the teams their selves.

In summary, from the technical perspective, the interviews brought up the finding that modular system architecture and microservices are the prerequisites for applying the agile practices. From the organizational point of view, the interviews brought up that common objectives of team members are important for managing and measuring the results of the teams, cross-functional teams should have all skills to cover the full development cycle of a product and responsiveness to changes culture is the key element in agile development. From the development perspective, the interviews brought up that automated testing fastens the process in agile development and autonomous release process gives the independence to teams.

4.3. Discussion

The literature on the discussed topic is relatively recent. Most of the publications are by companies, consultants or practitioners, who have practiced agile development in a certain organization. Large part of the literature covers and analyses the cultural change and empowerment of the organization, when moving from waterfall to agile software development in the organizations in general. Not enough literature is covering specifically the problems tackled in the software development and software delivery processes when using agile methods.

The findings from literature review are general, but very important to set the right tone in the whole organization, when implementing agile. Agile is about the mind-set of the whole organization, not only the development specifically. It is also important to understand, that although the literature about agile methods proposes very concrete set-ups for the organization and processes, the best result is achieved by combining different agile methods and components. Each organization should find its own and best suitable set-up. When taking agile into use first time, it is important to share the information and offer trainings about the

methodology that will be implemented. Compared to waterfall, agile is a totally different approach to development and every change needs to be managed carefully.

Although some of the issues were similar to the findings from literature review, the interviews with the companies brought up many other issues, how to use agile practices in SPI. The best way to learn for large financial institutions is from fintech companies. The fintech companies have had the opportunity to build the whole agile development process from scratch and design everything with the attitude "if we had to do it all over again". Learning from other large financial institutions is also valuable, but such organizations have just went through the process of moving from waterfall to agile by their own. Their changes are done based on their exact needs and they might still have the legacy in their processes or they have not jet done so radical changes.

The findings from interviews revealed that agile development cannot implemented in larger organizations, if their existing system is monolith. In such case, it is not easy to divide the work between teams and give them independence in development. For applying the agile practices, the systems have to be modular and designed according to microservices principle. Another important aspects raised in the interviews were connected to having common objectives of team members. That is helping to manage and measure the results of the teams. The cross-functional team setup and responsiveness to changes culture were other two aspects from the organizational perspective. As agile means faster and more flexible software development, all other processes like testing and releasing should be organized in the most automated way not to waste time there. If the literature review brought up the findings in general level, the interviews were very valuable in bringing up the findings in deeper level.

In the interviews, it came out that the companies do not have any good place or form to share their experience in setting up and developing the agile practices in different companies. The main source for their knowledge is coming from the literature and case studies from foreign companies. For example, in interviews, people referred to Spotify engineering culture and brought out so called Spotify Model for scaling agile. Henrik Kniberg, who has introduced the Spotify Model to the public, is an agile and lean coach at Crisp in Stockholm, working mostly with Spotify and Lego. He is the author of three books ("Scrum and XP from the Trenches: How We Do Scrum", "Kanban and Scrum: Making the Most of Both", "Lean from the Trenches: Managing Large Scale Projects with Kanban") that have had over half a million of readers in 12 languages and are used as a primary tool for agile and lean software development in hundreds of companies worldwide at the moment of writing the current thesis.

Spotify's approach to agile scaling is considered as the culture, not the process or structure. Spotify Model is unique as it lies on having aligned autonomy, and very few mandatory practices and hard rules. With aligned autonomy, Spotify keeps autonomy of teams, but stays collaborative and working for the same high-level objectives. Spotify Model is promoting short intervals between releases to react faster to the customer needs and investing into failure recovery. In Spotify, the development teams are called Squad and a group of Squads is called Tribe. The Squads in the same Tribe work in a related field. In addition to that, Chapter is formed of people who have similar skill-set and who are working in the same Tribe in similar capacity. People that share same interests, knowledge, tools and code is called Guild [54, 55].

In conclusion, it is important to note that although people interviewed have worked mainly in their own companies for longer period, they are still using many similar agile practices. It shows that the same findings brought up in all companies are worth to consider seriously.

5. Software Process Improvement. LHV Bank Case

This part of the thesis describes the software process improvement at LHV Bank. After the analysis and findings, the proposals are made to improve the current software process at LHV Bank that was described earlier.

5.1. Required Software Process

As the current software process is in place and working well in many aspects, the proposals can be considered as incremental changes to redesign the process. The following proposals are numbered according to the priority or relevance that can be made immediately, not as a part of the longer development.

Proposal 1: Introduce agile management culture organization-wide.

Based on the Finding 1 from the literature review and interviews in general, it can be concluded that the agile is not just about the IT development, but it is about the mind-set of the whole organization. In some smaller organization it really covers everyone from the management to the customer support. In larger financial institutions, it is about the management, product management and IT. It can be even said that agile methods are about the new management style and culture of the new generation financial institutions.

It was all starting from the technology companies, but in practice it can be applied also to other types of companies, including financial institutions, where the technology is playing larger and larger role in every year. In the beginning of 2000-s, around 20% of the financial institutions' annual budget were IT expenses, in the beginning of 2010-s it was already 25% and currently already 33% is considered as reasonable level. It all shows that it is time to change the management principles in financial institutions. Instead of rigid silos between business and IT, the financial institutions consisting of teams can be considered.

In LHV Bank, the agile methods and management culture should be introduced to the management of the bank in broader level. It might be a larger cultural change than can be estimated at the moment, but it is worthwhile to try more loose and less controlling management style in general, in product development and in IT development.

In the financial institutions, the software is usually developed in-house and not outsourced as in many other businesses. Financial institutions are becoming more like technology companies with high focus on product and software development. Therefore, it is important that financial institutions in general have good understanding about the agile methods and practices.

Proposal 2: Organize relevant trainings.

Based on the Finding 3 from the literature review, it is important to have proper methodology training for the whole organization to increase the know-how about implementing the agile methods in financial institution.

In LHV Bank, when introducing the agile management culture, the relevant trainings should be organized and building up the knowledge base should be taken seriously. Learning from other practitioners from the financial sector would add extra value in the learning cycle. The training should take into account the differences of financial institutions as much as available.

Proposal 3: Assemble concrete teams for products.

Based on the Finding 7 from the interviews, the agile organization has to be divided into cross-functional teams in the financial institution. The teams as smaller units are responsible for their own product and process from the very beginning until the very end. Only that can increase the speed in the development.

In LHV Bank, there whole organization structure should be overviewed with the aim of building more close teams with POs and development teams. To avoid the conflict, where each development team is having in average 3.5 products to develop which brings the conflict on POs level about the priorities, the number of products should be reduced by consolidating some of the products. It is even important to review the whole organizational structure of the bank and change and reduce the number of business divisions.

As a proposal, on the division level, Retail Banking and Private Banking should be merged. On the department level, Investments and Portfolio Management should be merged. On the product level, Client Services and Gold Clients should be merged and Cards Issuing, Cards Acquiring, ATM, Banklink and Connect should be merged. The main gain from merging these units is to have one PO for relatively similar products.

Proposal 4: Set common business objectives for team members.

Based on the Finding 6 from the interviews, common objectives of team members in financial institutions are helping to manage the teams, communicate the objectives of the product to each team member and measure the results of the teams later on.

In LHV Bank, after assembling more concrete teams for products, the common objectives can be set for the whole team. In financial institution, it is even easier to do as the whole organization is already having a high level of financial literacy. Usually, financial institutions are project oriented in developing their business. Such culture helps to set and follow the objectives of the teams more easily.

The objectives have to include quantifiable business objectives and qualitative development objectives, but it is important that at least basic business objectives are set for each team member. For example, if the bank is planning to release a new credit product Student Loan for students, the team is having the objective to launch the new product by the beginning of September, but in addition to that, the team is having the objective that 1000 Student Loans are sold in September with the help from Marketing and Customer Support. Such objective will make the whole team be more interested in the actual viability of the new product and react quickly with the changes, if something needs to be improved for the customer.

Proposal 5: Give more autonomy to the teams.

Based on the Finding 8 from the interviews, responsiveness to changes culture is the key element in agile development in financial institutions. In addition to having the team objectives, the teams should also have the power and autonomy to work towards achieving these objectives.

In LHV Bank, more autonomy should be given to teams by increasing their decision power over their products. People working in financial institutions are highly educated and evaluate the trust and decision power given to them. Although the financial budget of the team is limited as the number of team members is more or less fixed, the power should be given over the decisions, what kind of products or product features to develop, in which order and at what time. The income earned from products could be improved, if people close to the business and with development resources can have more decision power.

It is not only about the PO, but also about the SDUM to feel that they are here for achieving the common objectives and they have the power and resources to do that. It will encourage people to start a new product as a MVP and test it on the real customers, rather than developing anything new as a large project without knowing, how customers will react to that. Giving more autonomy to the teams will ensure that the whole development is moving to the direction of continuous delivery.

Proposal 6: Review the agile development method used with its components.

Based on the Finding 2 and Finding 4 from the literature review, each financial institution should find its own way for the agile development. Following only one or another method might not suit to a specific organization. Combining components from different agile methods will give the best result for the financial institution. In this process the organization has to be open minded. Based on the Finding 7 from the interviews, it is important to have as automated testing as possible to reduce the time spent on manual testing for each change.

In LHV Bank, although the components from Scrum, XP, Lean Software Development and Kanban are used, the whole development process should be reviewed by learning from the latest best practices of the other financial institutions and experience the whole organization has collected during the last five years when implementing agile practices.

In software development, using refactoring is necessary, and in software testing, using regression tests is essential to decrease the time on testing and to increase the reliability of the systems. At the same time, financial institutions have to guarantee that the code is reviewed based on the four-eye principle, but that can be solved between two engineers as well.

When giving more autonomy to teams, making full reviews to new code is not necessary as it decreases the motivation of people. Also, in the agile software development, counting the time used for each task is not adding value.

In LHV Bank, the communication tools should be reviewed. Continuing to use Slack is reasonable, but in addition to that, Zoom should be taken into use for teleconferences between the team members locating in different places.

Proposal 7: Automate the release process.

Based on the Finding 10 from the interviews, autonomous release process gives the independence to the teams in financial institutions. It is one the factors of speeding up the development by making the releasing phase shorter. In the easier release process, the team is able to react to the bugs faster.

In LHV Bank, the releasing process definitely needs to be improved as it is long, requires much manual work and involves people from different departments. The whole release process should be redesigned and relevant information systems taken into use for that.

Although the priority of automating the release process is low in this list, it can actually be done as one of the first thing in parallel with the previous proposals.

Proposal 8: Use more modular system architecture and microservices.

Based on the Finding 5 from the interviews, the agile methods can be exploited in the best way, if the financial institution is having a modular system architecture. The trend in modern system architecture is moving towards microservices.

In LHV Bank, the first core system of the bank accounts was developed already 20 years ago and it has remained the basis for most of the main banking services. Although the new modules of the system have already been built separately, these are still relatively large and do not enable different development teams to work on the same component at the same time. Therefore, it is important that in the future, the bank's infrastructure will move towards smaller components and microservices. That will enable to increase the speed in the development of the components.

It is important to follow the way, where the core system is separated to smaller components step by step using Java and not ColdFusion anymore.

5.2. Threats to Validity

There are several threats to validity of this thesis. The main threats to validity are discussed in this part of the thesis.

5.2.1. Construct Validity

The analysis and findings were based on both literature review of agile methods and interviews of agile practices. Although some of the findings were similar, the interviews gave many different and additional findings compared to the literature. Therefore, it might be possible that literature review did not fully cover the agile methods and there were not enough findings on the topic.

Another threat to validity is arising from the interviews itself as the questions discussed in the interviews might have differently understood by the people interviewed. Also, the received answers might have interpreted in different way by the author than by the persons interviewed.

5.2.2. Internal Validity

The thesis might give a subjective overview of the product development, IT development and delivery processes in the organization. The bank itself is a highly regulated institution and for security reasons does not allow to describe all the information systems it is having in a very detailed level. Therefore, the full picture of the current software process might not be absolutely correct and objective.

5.2.3. External Validity

The findings from interviews were based only on four interviews. Although the companies interviewed were in different sizes and in different development stage, they were all having Estonian roots and their main development is done in Estonia. It might be possible that the companies in other countries or regions worldwide are having different agile practices and the thesis does not cover the best practices used worldwide.

Also, although some companies chosen for the interviews are having relatively large size by number of people in total and in development specifically, it might still happen that they are small in the World context, where the financial institutions are having tens or hundreds of thousands employees. Due to this, the results might have limited potential to be generalizable.

5.2.4. Reliability

The research carried out in the thesis cannot be reproduced as it is based on the literature review and conducted interviews. The software processes in companies participated in the interviews and in LHV Bank are in constant change and the results of the research can be very different after some time compared to the current moment.

The proposals made to improve the software process in LHV Bank might not be correct and fully covering the problematic areas. The proposals made are in general level and might need to be elaborated further, when these are actually implemented. The proposals are not implemented in the real life as it will take time. Changing the culture in the whole organization or even among around 100 people, who are involved in development, takes much time and it cannot be made in the frame of the thesis. In the implementation phase, it might happen that some of the proposals are not relevant at all or some are missing that could improve the process further. Therefore, the proposals made in the thesis might only be a part of the solution for implementing agile practices.

6. Conclusions

The thesis is about software process improvement using agile methods in financial institutions, based on the LHV Bank case. For that, the current software process at LHV Bank was described and the main bottlenecks or improvement areas were pointed out. For making the proposals to improve the software process in LHV Bank, the agile methods used in SPI through the literature analysis and the agile practices used in SPI through the conducted interviews were analyzed. Based on these analyses, the findings were brought out and discussed. After the analysis and findings, the proposals were made to improve the current software process at LHV Bank.

During the course of writing the thesis the author of the thesis learned different agile methods and scaling agile frameworks. He also learned the current software process at LHV Bank and made the proposals to improve it. Although the agile methods and practices have been used in the development at LHV Bank already for couple of years, the author learned that to make a successful breakthrough in that the agile practices have to be take into used organization wide and in more powerful way. Agile is not relevant only in managing the development, but managing the whole organization. The organization should first accept looser and not so controlling management culture. In the beginning of writing the thesis, the objective of the author was to make very concrete proposals, how to fasten the software development process, but he soon learned that improving anything in that part of the process does not give much additional value as it forms only a small part of the whole development process in the organization and without changing the overall culture of the organization, not much will improve. If the agile is discussed only among engineers, but the management or other divisions of the organization are not involved, not much will change.

The list of proposed changes is formed and the next step is to implement the proposed changes at LHV Bank. The implementation plan of proposed changes will be drawn, the resources committed and the execution done. Following the implementation, the effects can be evaluated to examine, if improvements are useful for other financial institutions.

7. References

[1] Romanova I., Kudinska M., Banking and Fintech: A Challenge or Opportunity?, *Contemporary Studies in Economic and Financial Analysis*, 2016, vol. 98, pp. 21-35.

[2] Kotarba M., New Factors Inducing Changes in the Retail Banking Customer Relationship Management (CRM) and Their Exploration by the Fintech Industry, *Foundations of Management*, 2016, vol. 8.1, pp. 69-78.

[3] Vasiljeva T., Lukanova K., Commercial Banks and Fintech Companies in the Digital Transformation: Challenges for the Future, *Journal of Business Management*, 2016, vol. 11, pp. 25-33.

[4] Humphrey, W. S., Managing the Software Process. Addison Wesley. 1989.

[5] Basili V., Caldiera G., Improve software quality by using knowledge and experience, *Sloan Management Review*, 1995, pp. 55-64.

[6] Basili V., Caldiera G., Rombach H., Goal question metric paradigm. *Encyclopedia of Software Engineering*, New York: John Wiley & Sons, 1994, p. 528-532.

[7] McFeeley B., IDEAL(SM): a user's guide for software process improvement. Washington: Carnegie Mellon University. 1996.

[8] Ashmore S., Runyan K., Introduction to Agile Methods. United States of America: Pearson Education, Inc. 2015.

[9] Salo O., Abrahamsson P., An iterative improvement process for agile software development, *Software Process: Improvement and Practice*, 2007, vol. 12 pp. 81-100.

[10] Garzás J., Paulk M. C., A case study of software process improvement with cmmi-dev and scrum in spanish companies. *Journal of Software: Evolution and Process*, 2013, vol. 25, no. 12, pp. 1325-1333.

[11] Rawsthorne, D., Shimp D., Agility, *A White Paper from 3Back, LCC*, 2016.

[12] Bassens D., Hendrikse R., van Meeteren M., The Appleization of Finance. Vrije Universiteit Brussel, 2017.

[13] Dapp T., Slomka L., Fintech reloaded – Traditional banks as digital ecosystems, *Deutsche Bank Research, Current Issues, Digital economy and structural change*, 2015.

[14] Cadle J., Yeates D., Project Management for Information Systems. London: Pearson Education. 2008.

[15] Boehm B., A View of 20th and 21st Century Software Engineering, *Proceedings of the 28th international conference on Software engineering*, 2006, pp. 12-29.

[16] Livermore J., Factors that Significantly Impact the Implementation of an Agile Software Development Methodology, *Journal of Software*, 2008, vol. 3, no. 4, pp. 31-36.

[17] Agile manifesto. http://agilemanifesto.org/ (06.01.2018)

[18] Phalnikar R., Deshpand V. S., Joshi S. D., Applying Agile Principles for Distributed Software Development, *2009 International Conference on Advanced Computer Control*, 2009, pp. 535-539.

[19] VersionOne. State of Agile Development Survey. 2017. https://www.versionone.com/pdf/state-of-agile-development-survey-ninth.pdf (06.01.2018)

[20] Sims C., Johnson H. L., The Elements of Scrum. Foster City: Dymaxicon. 2011.

[21] Beck K., Andres C., Extreme Programming Explained. Massachusetts: Pearson Education, Inc. 2005.

[22] Stellman A., Greene J., Learning Agile. Sebastopol: O'Really Media, Inc., 2015.

[23] Kupiainen E., Mäntylä M.V., Itkonen J., Using metrics in Agile and Lean Software Development – A systematic literature review of industrial studies, *Information and Software Technology*, 2015, vol. 62, pp. 143-163.

[24] Kniberg, H., Skarin, M., Kanban and Scrum: Making the Most of Both. United States of America: C4Media. 2010.

[25] Dikert K., Paasivaara M., Lassenius C. Challenges and success factors for large-scale agile transformations: A systematic literature review, *Journal of Systems and Software*, 2016, vol. 119, pp. 87-108.

[26] Alqudah M., Razali R., A Review of Scaling Agile Methods in Large Software Development, *International Journal on Advanced Science Engineering Information Technology*, 2016, vol. 6, pp. 828-837.

[27] Leffingwell D., Yakyma A., Knaster R., Jemilo D., Oren I. SAFe Reference Guide. Scaled Agile Framework for Lean Software and Systems Engineering. Crawfordsville: Scaled Agile, Inc. 2016.

[28] Larman C., Vodde B., Large-Scale Scrum. More with LeSS. Crawfordsville: Pearson Education, Inc. 2016.

[29] Madura J., Financial Markets and Institutions. Boston: Cengage Learning. 2015.

[30] Tinnilä M., Impact of Future Trends on Banking Services, *Journal of Internet Banking and Commerce*, 2012, vol. 17, no. 2.

[31] Marous J., Top 10 Retail Banking Trends and Predictions for 2014, *The Financial Brand*, 2014.

[32] Reddy A., The Scrumban [R]evolution. Getting the Most Out of Agile, Scrum, and Lean Kanban. United States of America: Pearson Education, Inc. 2016.

[33] Livermore J. A., Factors that impact implementing an agile software development methodology, *SoutheastCon, 2007. Proceedings*. IEEE, 2007.

[34] Ladas K., Scrumban And Other Essays on Kanban Systems for Lean Software Development. Seattle: Modus Cooperandi Press. 2008.

[35] Jha M. M., Vilardell R. M. F., Narayan J., Scaling Agile Scrum Software Development: Providing Agility and Quality to Platform Development by Reducing Time to Market, *Global Software Engineering (ICGSE), 2016 IEEE 11th International Conference*, 2016, pp. 84-88.

[36] Kyte A., Norton D., Wilson N., Ten Things the CIO Needs to Know About Agile Development. United States of America: Gartner. 2014.

[37] Brooks F., The Mythical Man-Month. Crawfordsville: Addison Wesley Lomgman, Inc. 1995.

[38] Harjrizi E., Bytyci F., Agile Software Development Process at Financial Institution in Kosovo, *IFAC-PapersOnLine*, 2015, vol. 48-24, pp. 153-156.

[39] Weiss S. K., Brune P., Crossing the Boundaries – Agile Methods in Large-Scale, Plan-Driven Organizations: A Case Study from the Financial Services Industry, *Advanced Information Systems Engineering*, 2017, pp. 380-393.

[40] Jureczko M., The Level of Agility in Testing Process in a Large Scale Financial Software Project, *Software engineering techniques in progress, Oficyna Wydawnicza Politechniki Wroclawskiej*, 2008, pp. 139-152.

[41] Papadopoulos G., Moving from Traditional to Agile Software Development Methodologies Also on Large, Distributed Projects, Procedia – Social and Behavioral Sciences, 2015, vol. 175, pp. 455-463.

[42] Nordea, A Uniform Heartbeat with Help from Scaled Agile Framework and IJI, *Ivar Jacobson International SA*. 2015.

[43] Seffernick T. R., Enabling Agile in a Large Organization Our Journey Down the Yellow Brick Road, *Agile Conference (AGILE), 2007*, IEEE, 2007.

[44] Couzens J., Implementing an Enterprise System at Suncorp Using Agile Development, *20th Australian Software Engineering Conference: ASWEC 2009*, 2009, pp. 35-39.

[45] Svejvig P., Nielsen A.-D., The Dilemma of High Level Planning in Distributed Agile Software Projects: An Action Research Study in a Danish Bank, *Agility Across Time and Space*, 2010, pp. 171-182.

[46] Runeson P., Höst M., Guidelines for conducting and reporting case study research in software engineering, *Empir Software Eng*, 2009. https://link.springer.com/article/10.1007/s10664-008-9102-8 (11.05.2018)

[47] Sterb K. C., Exploratory Case Study. In: Encyclopedia of Case Study Research. Thousand Oaks: SAGE Publications, Inc. 2012.

[48] Chen L., Microservices: Architecture for Continuous Delivery and DevOps, *IEEE International Conference on Software Architecture*, 2018.

[49] Chen L., Continuous Delivery: Huge Benefits, but Challenges Too, *IEEE Software*, 2015, Vol. 32, Issue 2, pp. 50-54.

[50] Niven P. R., Lamorte B., Objectives and Key Results: Driving Focus, Alignment, and Engagement with OKRs. Hoboken: John Wily and Sons, Inc. 2016.

[51] Ries E., The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. New York: Crown Business. 2011.

[52] Bennett, S., Computer Orchestration: Tips and Tricks. Norfolk: PC Publishing. 2009.

[53] Humble J., Farley D., Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Boston: Pearson Education, Inc. 2011.

[54] Kniberg, H., Spotify engineering culture (part 1). 2014. https://labs.spotify.com/2014/03/27/spotify-engineering-culture-part-1/ (30.04.2018)

[55] Kniberg, H., Spotify engineering culture (part 2). 2014. https://labs.spotify.com/2014/09/20/spotify-engineering-culture-part-2/ (30.04.2018)

Acknowledgements

The author of the thesis is very thankful for the people kindly participated at the interviews. The interviews done in different companies gave a very valuable and useful information from the practitioners and helped to detect the findings for proposals. A special thank to PhD Fredrik Payman Milani for supervising and reviewing of an earlier draft of the thesis.

Appendices

I Interview Questions

1. Background of the company

- Name of the company
- Field of activity of the company
    - Active since
    - Products offered
    - Size
    - Growth in the last 5 years
- Please describe the organizational structure of the company
    - Size and division of the personnel
    - Size and division of the development personnel
    - Size and division of the IT development personnel

2. Product development process in the company

- Please describe the product development process in your organization
    - What kind of analysis and analysis tools are used?
    - Are you using user stories or any other techniques?
    - What makes it successful in your opinion?
    - What could be improved?

3. Software development process in the company

- Please describe the software development process in your organization
    - What kind of agile method is used?
    - Which components are you using from Scrum?
    - Which components are you using from Extreme Programming?
    - Which components are you using from Lean Software Development?
    - Which components are you using from Kanban?
    - What makes it successful in your opinion?
    - What could be improved?
- How is the testing process organized?

4. Software delivery process in the company

- Please describe the architecture of your system
- Please describe the software delivery process in your organization
    - How is organized release planning?
    - How are organized releases?
    - Are you using continuous delivery process?
    - What makes it successful in your opinion?
    - What could be improved?

5. Scaling agile and growth related issues in the company

- How many tasks are you having in the product backlog and how many in development?
- If and how are you measuring the speed of development?

- Has the speed of development changed in the last years?
- How have the development processes changed with the growth of the company?
- Are you dealing with Software Process Improvement issues regularly?
- Are you using frameworks that scale agile?
    - What kind of framework is used?
    - What makes it successful in your opinion?
    - What could be improved?
- What are the current issues and proposals discussed to improve the software development process?

II License

Non-exclusive license to reproduce thesis and make thesis public

I, **Erki Kilu**,

1. herewith grant the University of Tartu a free permit (non-exclusive license) to:

1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

Software Process Improvement Using Agile Methods in Financial Institutions. LHV Bank Case,

supervised by Fredrik Payman Milani, PhD.

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive license does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, **21.05.2018**