Tero Laitila

USER-CENTERED DEVELOPMENT AND MAINTENANCE METHOD FOR SOFTWARE TEAMS

JYVÄSKYLÄN YLIOPISTO
TIETOJENKÄSITTELYTIETEIDEN LAITOS
2017

TIIVISTELMÄ

Laitila, Tero
User-centered development and maintenance method for software teams
Jyväskylä: Jyväskylän yliopisto, 2017, 129s.
Tietojärjestelmätiede, pro gradu -tutkielma
Ohjaaja(t): Tuunanen, Tuure

Tämä tutkimus pyrkii löytämään sopivan metodin jatkuvaan ohjelmistokehitykseen (tuotekehitys ja ylläpito). Se yhdistää ketterän ohjelmistokehitysmenetelmän uusimpiin operatiivisiin metodeihin sekä käyttäjäläheiseen ohjelmistosuunnitteluun. Tutkimus sisältää uuden mallin, joka sisältää metodin arvot, tavoitteet, periaatteet, säännöt, prosessimallin, roolit ja vastuut ohjelmistotiimeille. Tämän metodin avulla ohjelmistotiimit voivat mahdollisesti tehokkaammin tuottaa ja ylläpitää käyttäjäystävällisiä palveluita. Tutkimuksessa on haastateltu IT-alan ammattilaisia isoimmista suomalaisista IT-yrityksistä. Tutkimuksen lopputulos on metodi ohjelmistokehityksen ammattilaisille. Metodi yhdistää käyttäjäläheistä suunnittelua nykyaikaisiin ketterän ohjelmistokehittämisen metodeihin.

Asiasanat: Ketterät menetelmät, Agile, Scrum, DevOps, Käytettävyys, Käyttäjäkokemus, UCD, Jatkuva kehittäminen, Tietojärjestelmien kehittäminen

ABSTRACT

Laitila, Tero
User-centered development and maintenance method for software teams
Jyväskylä: University of Jyväskylä, 2017, 129 p.
Information Systems, Master's Thesis
Supervisor(s): Tuunanen, Tuure

This research aims to find answer for the question: How to continuously develop and maintain software while fulfilling customer and user expectations? It combines agile development methods and DevOps together with user-centered design. Research includes new method which includes values, objectives, principles, rules, process models, roles and responsibilities for a software teams. By using this kind of method software teams can possibly develop and maintain user focused software more efficiently. Research includes interviews from information technology professionals from Finnish companies. The end result of the research is method which covers software development and maintenance artifacts.

Keywords: Agile, Scrum, DevOps, UCD, Continuous development, Method, Software development

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Explanation
UCD	User-centered Design
IT	Information Technology
IP	Intellectual property
B2B	Business to business
SaaS	Software as a Service
CD	Continuous Development
SysAdmin	System Administrator
DBA	Database Administrator
CI	Continuous Integration
TDD	Test-driven Development
UI	User interface
UX	User experience
DoD	Definition of Done
DoR	Definition of Ready
DSRP	Design Science Research Methodology
QA	Quality Assurance
SUS	System Usability Survey
SAFe	Scaled Agile Framework

TABLE OF CONTENTS

1 INTRODUCTION OF RESEARCH

Digitalization is a term which occurs in many circumstances nowadays. Gartner (2013) defines digitalization as an emerging business model that includes extension and support of electronic channels, content and transactions. Various organizations are looking way to improve and enhance their processes and lower cost level. Typically organizations are aiming to find competitive advantages or respond to challenges in their industry. New technologies including software and hardware are changing organizations ways of working. Successful organizations are today typically technologically orientated and they effectively use new technologies to improve their operations. When adapting new technologies and information systems organizations need to understand their own processes. By understanding processes organizations can find new advanced tools and information systems to improve operative work. When adapting new ways of working change management is crucial as these new technologies and tools need to deploy also to daily work.

Behind of new technology there are always information technology teams which are producing tools and systems for businesses. This research focuses on IT teams which are providing new information systems for their customers. Previously software was sold to customers as an installable package. During the years business models have changed and most of the new information systems or software is provided to customer as a service. As a service means that service providers provides to customer needed hardware (servers), software, customer service and maintenance services. With this kind of approach organizations are moving IT related functions to service providers and at the same time they can focus on their core business. In the big picture this kind of approach is beneficial also IT service providers as they can handle all components of their services by themselves. Biggest factor for this kind of business change has been evolvement of telecommunication networks. If previously networks weren't fast or reliable enough today those are. With robust networks service providers can provide their services example from their own server environments. In practice information systems can be located far away from the end-user. This also means that service providers can create hosting environments more efficient way than

before. With new technologies and tools software can be designed to work more effectively in shared environments. If earlier one information system was designated for one customer today same information system can serve even hundreds of customers with same instance. This kind of change in business is beneficial for all parties as service providers can lower their costs with shared services and customers can procure services with lower, cost effective prices.

Behind of information systems and IT related services are IT teams and members of the team. As business around teams has changed during the years so has changed the way of working in teams. New kind of business models and evolvement of technologies requires change to the team working also. As previously mentioned most of the information systems are provided to customer as a service and one service can be used of even hundreds of customer. This kind of environment requires for the IT teams robust processes and ways of working to deliver required high-quality services in a fast changing business environment.

1.1 Research problem

Information system development has evolved during the years from method to another. During these years concentration has been in the information system development phase. Earlier software projects have included one development phase and after moving to maintenance phase the service or product which customer has stopped to evolve without additional orders from service or product provider. Nowadays customers on business side are expecting continuous development from those services what they are buying. This expectation comes from consumer business where software continuously evolves; nowadays companies are expecting same from their service providers. Example operating system providers like Microsoft and Apple are developing their systems all the time and within certain time cycles they release new version of their operating systems. In internet era service providers are developing their services in quicker pace than previously. Current techniques and web-based services are providing environment where releasing and continuous evolving is easier than previously. As internet availability has increased also B2B customers are buying information systems as a service which means that they don't necessary need to host own server environments anymore. Service providers are providing to customers their software with Software as a Service (SaaS) model. For the customer this is also easier way to procure information systems. Continuously evolving software requires releases in fast pace. If development and maintenance has earlier relatively clear line between different lifecycle phases, nowadays this line has been fading out and development happens iterativelys.

When technology has evolved consumers have been able to get familiar with many new services and systems. Many large consumer services like Facebook, Google and Spotify are based on easy and user friendly interface. When consumers use these systems their expectations to B2B software and services are going higher all the time. This kind of evolution puts B2B software providers to

position where they need to focus more and more to user experience and usability of software and services.

Main problem behind the research is lack of method which consist user experience perspective together with modern agile development and maintenance methods. For IT teams there are methods available which consist certain part of software development phases but there is not available method which combines suitable parts from different methods in one method. With this kind of method teams would find development and maintenance method which answer to current users and business needs. Main questions for the research is: *How to continuously develop and maintain software while fulfilling customer and user expectations?*

1.2 Motivation for the research

Motivation for this research comes from my daily work. I am working on IT business and I have business to lead. My objective is to find new ways of working and help team to improve and develop their working as a team. We have used several year agile methods and I have experience about those also on my early positions. I have been involved to ramp-up and maintain large scale web-services which had multiple millions daily users. Personally I have quite solid understanding of software development and maintenance process but during the years used methods have changed as business also. During the years I have noticed that meaning of user experience and software usability have increased. Already in the beginning of sales process when you meet the customer first time it is important to give good impression about the services or software which you are offering to customer. Good feedback from end-users is always asset when closing the deal with customer. Without doubt customer is first paying attention to functionalities and price but on the final decision user experience and usability are playing crucial role. In this research I will observe things mainly from B2B perspective.

I hope that the end result will help also other companies which are operating in the IT sector. My goal is to keep improved method such a general level that it could fit for as many companies as possible. Method will have limitations example regarding team size but it doesn't exclude away any bigger companies. Also in a bigger companies teams are working within a product lines and with certain products and they have quite similar size teams than have in small and medium size companies. My goal is to provide method which would help any software team to provide user friendly software to customers or consumers.

1.3 Objectives for the research

This research aims to *develop a method for continuous development and maintenance of software, which meets customer and user expectations*. On a high level objectives

can be shared on two groups business and technical objectives. In business side this method aims to provide software in a fast pace (aiming for fast market penetration) with lower costs. New method aims to improve also team work and collaboration within the team. On technical side the new method concentrates to provide high quality and better user experience. The main objective of the research is *develop a method for continuous development and maintenance of software, which meets customer and user expectations.*

Previously in the research is spoken about higher release pace but at the same this need is connected to term *time to market*. When developing new products it is important to get these products to markets as soon as possible. Every month that company invests on new product without getting revenue effects on revenue and to cash flow. If company can make sure that new products have short time to market it means that they are getting money in faster than previously and also this might be crucial if observing this from the whole market perspective. This way company can conquer market example six months earlier than competitor and it might gain huge amount of customer within that timeframe. From business point of view it also means that company gets good references and when competitor gets same product on the market it might have too large gap to catch up.

When developing new software, quality is one of the most important things. As reflected previously DevOps and Scrum are part of this study in method perspective. When bringing these methods and UCD together one of the goal is to increase quality from multiple aspects. While considering many aspects in this research it means internal and external quality factors. If looking quality from internal aspect it can example decrease amount of issues in the software. In practice this means more time to productive work when team members doesn't need to spend so much time solving these issues. If looking this from external perspective quality means working software without so many issues than before. At the same time when bringing UCD perspective on the method development it means that customer and end-user needs are taken care more thorough. From business point of view quality with greater user experience and usability bring more satisfied customers and with network effect it brings more new customers and revenue to the company. By bringing DevOps in this method the purpose is to increase automation level and provide essential tools. As the goal is to find suitable model for continuous development it is crucial to automatize as many steps in process as possible. Driving operational and development specialists together in same process one of the objectives is also bringing more control into the process. When everyone in the team knows and understands together agreed processes and ways of working it brings more discipline for the IT teams. Method's process phases need to be verified. With verification this method aims to provide clear approval or decline steps to the process and same time it means that team commits to so far done work or raises possible issues in a structured way. With all previously mentioned things one of the main objectives is also lower total cost of ownership.

Bringing several methods together in more rational way doesn't necessary mean that all of the steps and activities should just put together and create

a method which consist all artifacts from researched methods. It means that during the research it is important to find relevant artifacts, activities, practices and ways of working and use those in reasonable way. Aim for this kind of trimming is to keep method as simply as possible but at the same time as effective as possible. Researching this kind of method development from resource perspective one of the objectives is also to keep utilization level as high as possible. This way method focuses to keep labor cost as low as possible.

1.4 Research method and structure

Selected research method for this research is Design Science. Design Science helps researchers investigate, evaluate, produce and present research results. It also provides a process to produce research in structured way. The Design Science Research Process and Research Method are covered in more detailed level in chapter 3. The structure of document follows research process (FIGURE 1 Design Science Research Process). Problem identification, motivation description and objectives are defined in INTRODUCTION –chapter. Theory behind of the research is introduced in INTRODUCTION OF SOFTWARE DEVELOPMENT AND CHOSEN METHODS-chapter and the design of new method is available in METHOD FOR CONTINUOUS SOFTWARE DEVELOPMENT AND MAINTENANCE -chapter. When planning the research it is recognized that demonstration and evaluation of this kind of research´s is quite laborious and slightly difficult. Background of these challenges is the type of research. When creating new ways of working and renewing processes in teams it requires massive changes to daily working. Therefore artifacts are described in different ways (figures, chars, processes etc.) and the actual results are evaluated by information technology professional with different roles. Evaluation happens with interview which consist the introduction of new method for continuous development and maintenance part where professionals can reply to questions. After interviews new method is refined iteratively. Updated version is introduced in the 6 chapter and survey results in the chapter 7. The purpose of the survey is to re-evaluate method after second iteration round before the final discussion and conclusions are covered.

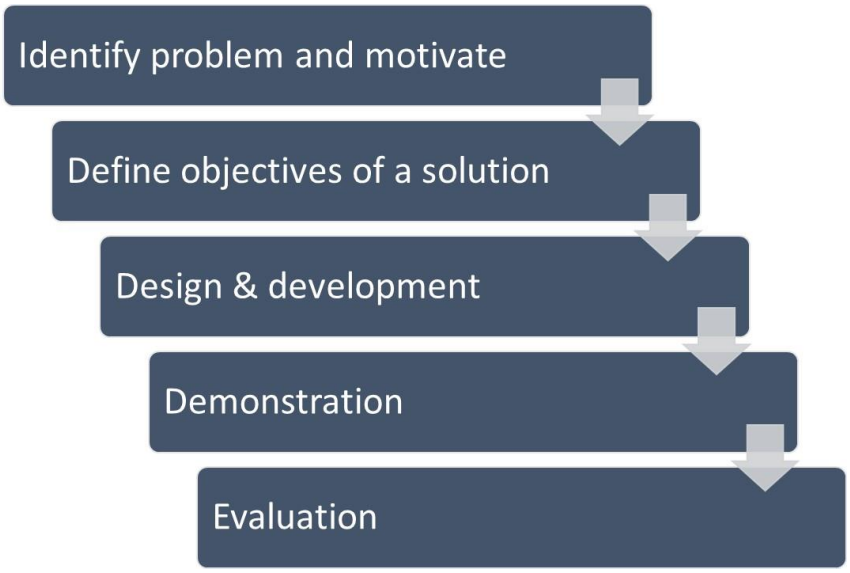


FIGURE 1 Design Science Research Process

1.5 Limitations and constraints

When developing methods and researching possible best sides of different methods it is important to have clear limitations where the end result (method) can be used. Method introduced in this research is suitable for 5-15 IT teams which are developing and maintaining software. Chosen methods in the background and increasing needs of customer to provide SaaS service over the internet, creates limitations to the software and environment type. On this research is focused primarily to web-based software which usually is offered to customer with the SaaS model. This method doesn't have limitations considering lifecycle phases but it is focusing mainly on already implemented products. This research won't take a stance on team scalability but at the same time purpose is not to create limitations which could create obstacles in the sense of scaling.

Using new method requires basic understanding of Agile Methodologies and Software Engineering but at the same time objective is to create a method which would be quite easy to adapt in any team. Adapting this kind of method requires also technical knowledge and familiarization to tools from teams who are willing to use method but this is something what is left for a possible users. When using the method introduced in research it would be useful if team have previous experience of software development. Adapting method can be done in parts and it is not necessary wise to even try to get everything working in one time.

This kind of method development requires mature tools to work with. During the last decade different kind of cloud-platforms have released labor work from operative persons to development. Different cloud-platforms like Microsoft Azure and Amazon Web Services are in crucial role when team wants to do continuous development for the provided services.

2 INTRODUCTION OF SOFTWARE DEVELOPMENT AND CHOSEN METHODS

In following chapter research concentrates to create understanding for the fundamental methods on the research´s background like *Software Engineering*, *Information System Development* and *Software Maintenance*. First part of chapter will help reader to understand what kind of terminology is behind the research and on later part of this chapter research concentrates to understand how selected methods works and what those includes. These selected methods are *Scrum, DevOps* and *User-centered Design (UCD)*. As operative work in IT business needs also governance and business perspectives those are partially covered in research material but not inclusively than in literature of previously mentioned methods.

2.1 Software engineering; system development and maintenance

Software Engineering is "the application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software" (Ieee, 1990). Definition of Software Engineering can be seen as a generic term and it describes well also what kind set of activities and processes are done nowadays in IT teams. These teams provide IT processes and services either to internal or external clients. In the software development process there are four different high level phases: gathering requirements, planning the system, developing the system and delivering it (FIGURE 2 Software Development Process). Jayaratna (1994) defines that the Information System Development is seen often as a systematic process which is composed of actions like identifying, analyzing, designing and implementing. In a literature terms System Development and Information System Development are often used in same context when speaking of Software Engineering. Depending on source terms might slightly change in the process but the main function and objective for each of these phases has remained the same during the years.

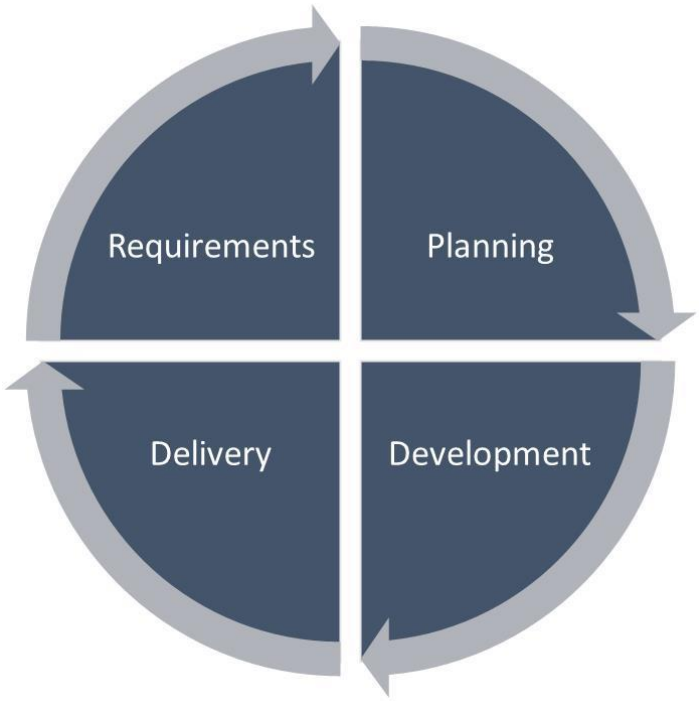


FIGURE 2 Software Development Process

Information system development and processes regarding it covers phases until delivery but after delivery there is also phase called maintenance which covers several activities. "Software maintenance refers to all the actions that are needed to keep software in such a running order that it achieves all its objectives from the beginning until the end of the usage" (Vehvilainen, 2000). ISO/IEC (2006) defines that "Software maintenance in Software Engineering is the modification of a software product after delivery to correct faults, to improve performance or other attributes". Typically system maintenance covers software, hardware and network related support functions.

2.2 Researched methods

As research´s objective *is develop a method for continuous development and maintenance of software, which meets customer and user expectations* it is important to understand what kind of current methods, processes and practices are used in IT operations nowadays and what kind of methods would be suitable for continuous development. Focus in this research is to understand more deeply following methods; *Scrum*, *UCD* and *DevOps* and inspect how those can be combined and refined to one common method for IT teams. At the same time when observing main concepts research will introduce other necessary terms and concepts around these main observation methods.

2.2.1 Agile and Lean

Agile Methodologies have already used for a long time in the software industry but wider spreading has happened quite tardily. Biggest difference in the Agile Methodologies (FIGURE 4 Agile Method from traditional waterfall method (FIGURE 3 Waterfall method) is iterative delivery which means repeating basic software development process (conception, initiation, analysis, design, construction, testing and deployment) in short development cycles. Agile software development praises things like collaboration and self-organizing individuals. There are several methods which follow Agile Manifesto and principles but all those differ from each other slightly. Most well-known agile software development methods/frameworks are: Scrum, Kanban, Scrumban and Extreme Programming (XP) (Poppendieck & Cusumano 2012).

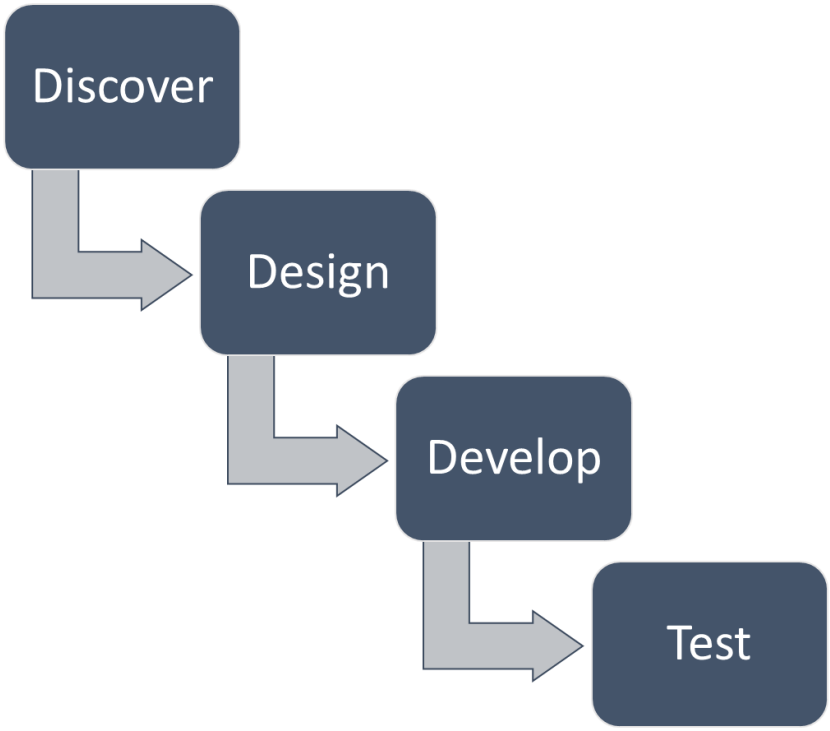


FIGURE 3 Waterfall method

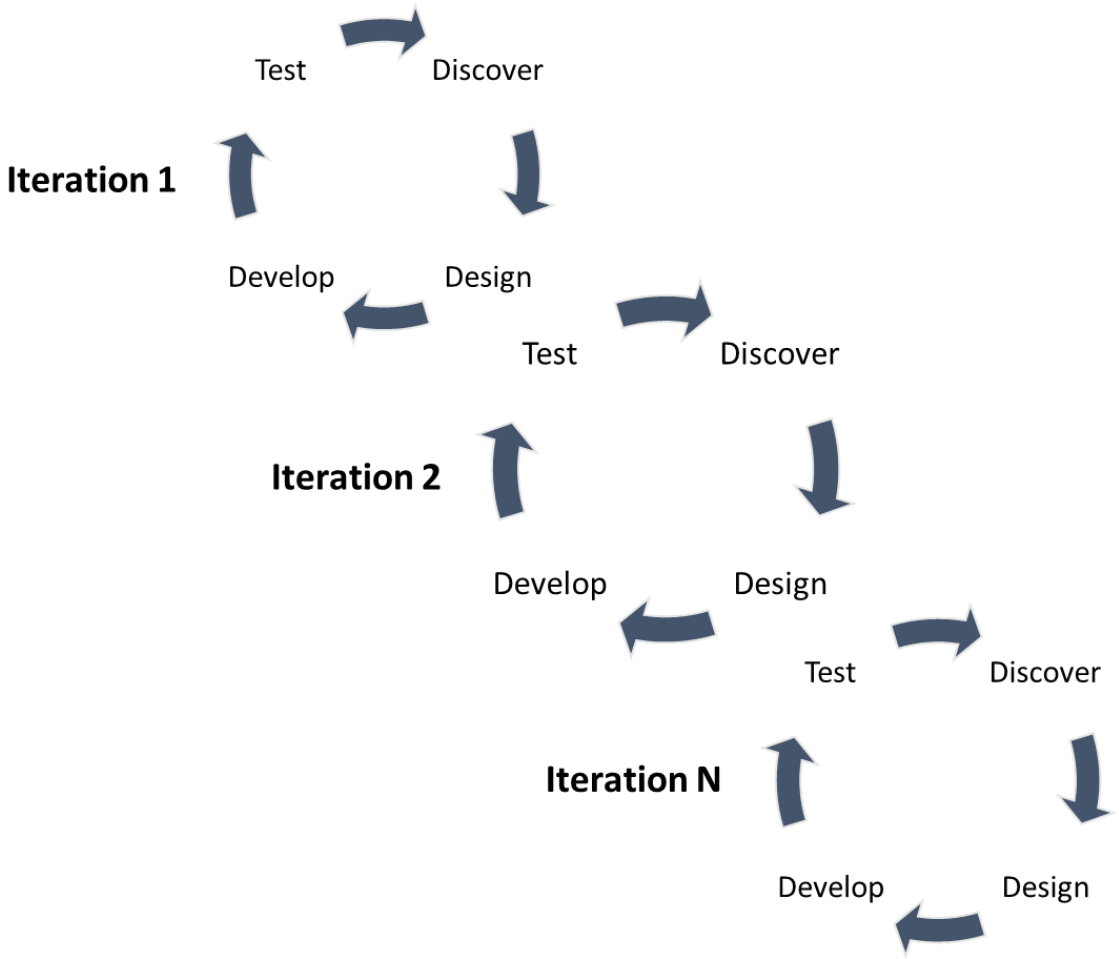


FIGURE 4 Agile Method

Agile Manifesto and Agile Principles (FIGURE 5 Agile Principles (Agile Manifesto, 2001) are probably the most well-known part of the agile methodologies. Agile Manifesto for software development highlights individuals and interactions over processes and tools. Agile Manifesto also praises working software, customer collaboration and responding to change over documentation, contracts and plans (Agile Manifesto 2001).

Principle
Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
Business people and developers must work together daily throughout the project.
Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.
Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
Continuous attention to technical excellence and good design enhances agility.
Simplicity--the art of maximizing the amount of work not done--is essential.
The best architectures, requirements, and designs emerge from self-organizing teams.
At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

FIGURE 5 Agile Principles (Agile Manifesto, 2001)

Lean is customer-orientated model for process management and leadership (Six Sigma, 2015). Lean was first time attached to software development at MIT during the mid-1980. Background of Lean comes from Japanese automotive industry where Toyota wanted to enhance their product development process (Poppendieck & Cusumano, 2012). With Lean Toyota achieved significant improvements to their production. After doing the changes they noticed that they need less direct labor, defects amount decreased substantially and they gained also higher quality for their products (Kennedy, 2010, 13). Lean's main objectives are quality improvement, waste elimination, time and cost reducing. Waste as a term means unnecessary things in production process which doesn't bring value to the process. In practice this means that by eliminating example useless meetings, tasks or documentation, company can increase the performance of their processes. Lean emphasizes word System as in this context it means team which operates as a whole. Science behind Agile and Lean are similar and both of those are concentrating on same core artifacts like people/individuals, fast/continuous delivery, quality and simplicity. (Poppendieck & Cusumano 2012)

2.2.2 Scrum

The Scrum is a framework to develop and maintenance complicated products (Schwaber & Sutherland, 2011). The Scrum is one of the agile methodologies and it follows agile principles and manifesto which are covered in the chapter 2.2.1. It consists different roles, activities, outputs and rules. Scrum is known for a lightness and same time it is easy to understand and learn for the IT teams. At the same time Scrum is difficult to control as it doesn't give comprehensive instructions to users. In other words framework brings loose structure for daily working but at the same time it leaves lot of responsibility and freedom for the users.

The Scrum has been used since early 90's for product development. Schwaber and Sutherland (2011) sees that Scrum is a group of processes (FIGURE 6 Scrum Process) and techniques which are making product management and system development viable for relevant stakeholders. Scrum utilizes iterative and incremental approach which is common for also to other agile software development methods which are covered on chapter 2.2.1. Schwaber and Sutherland (2011) highlight three pillars of Scrum. First of pillar Transparency is aiming

for visibility to all Scrum members. In practice transparency is viable in common language and in together decided definitions. Inspection pillar emphasizes the meaning of adequate monitoring. With adequate monitoring Scrum aims to advance implementation work and also remove obstacles which might disturb developers work. The last of three pillars is Adaptation. In practice this means that "if an inspector determines that one or more aspects of a process deviate outside acceptable limits, and that the resulting product will be unacceptable, the process or the material being processed must be adjusted" (Schwaber & Sutherland 2011; Dingsøyr 2010).

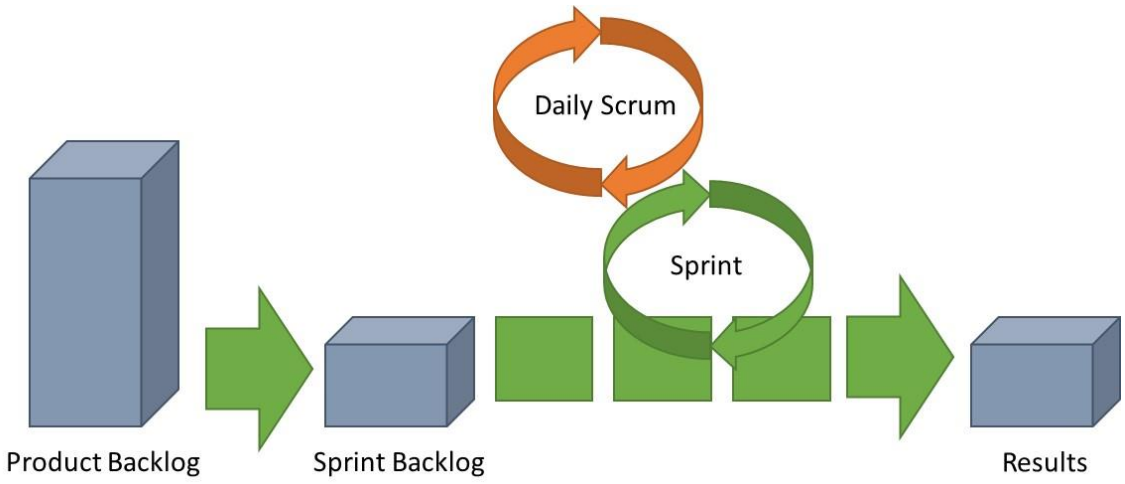


FIGURE 6 Scrum Process

A Scrum Team is the most important part of Scrum and it includes following roles; Product Owner, Development Team and Scrum Master. Schwaber and Sutherland (2011) highlights that the Scrum Teams are self-organizing and cross-functional. Product Owner is responsible for a Product Backlog. The Product Backlog is one of the Scrum artifacts and it can be seen as storage for a possible work. Scrum Alliance (2014) defines that the Product Backlog is "*a list of ideas for the product, in order of priority*" and Schwaber and Sutherland (2011) defines it with following words: "*The Product Backlog is an ordered list of everything that might be needed in the product and is the single source of requirements for any changes to be made to the product*". At the end of the day Product Owner is always accountable for the Product Backlog. The main idea of the Product Backlog is to gather relevant requirements in one place where those can be chosen to next iteration of work. Prioritization and choosing correct requirements belongs on the Product Owner responsibilities (Schwaber & Sutherland, 2011). Goal and mission need to be crystal clear for the Product Owner as selection of implementable items belongs to him or her. Scrum highlights the meaning of visibility and in the Product Owner role this is also one of the main things if thinking the Product Backlog. The Product Backlog and items inside of it need to understandable for everyone in the Scrum Team. The Scrum Master is servant-leader of the Scrum Team (Schwaber & Sutherland, 2011). The Scrum Master takes care of team´s daily

work and makes sure that everyone follows the policies, processes and rules. At the same time the Scrum Master is also protector of the team and on this role it is important to make sure that other Development Team members have industrial peace. The Development Team is the part of Scrum Team which completes most of the work what Scrum Team provides. All team members are professionals with necessary competencies and Scrum highlights the importance of capability for self-organizing. Development Team members might have usually specialized skills for certain areas but accountability is still shared for the whole Development Team. (Guang-Yong 2011; Mundra et al. 2013; Schwaber & Sutherland 2011; Dingsøyr 2010)

Development process of Scrum includes various events. These events are time-boxed as framework creators have wanted to avoid irregularity. All the Scrum events have maximum time limits as also Sprints have fixed timeframe. The Sprint has a timeframe which tells the length of development iteration. Usually the length of Sprint is between one to four weeks (Sutherland & Van Solingen, 2011, 28). During the Sprint development team produces useable and potentially releasable product increment (Schwaber & Sutherland 2011). Every Sprint includes all relevant events that belongs on process like; the Sprint Planning, Daily Scrums, the development work, the Sprint Review and the Sprint Retrospective (Schwaber & Sutherland 2011). The Sprint Planning is an event where team members are planning the content of next Sprint. Sutherland and Van Solingen (2011, 71) emphasizes that "*the essence of planning is to become predictable. Comparing estimates against reality facilitates learning*". The Scrum Master is responsible to arrange this meeting and owner of role takes care that all rules and policies of Scrum are obeyed. Output of Sprint Planning should be together agreed and understandable list of items what team is committed to deliver in upcoming Sprint. This list of selected items is called the Sprint Backlog. The *Sprint Backlog* is one of the artifacts of Scrum and it consist a set of selected functionalities from Product Backlog to the next increment (Schwaber & Sutherland, 2011). When Sprint Planning is done team can start their work (Sutherland & Van Solingen, 2011, 41). Work monitoring happens daily in Daily Scrums. Daily Scrums are short time-boxed meetings where team can identify and remove impediments, get answers shortly from a colleague and in general share knowledge of progress. Agenda for participants is usually following; what did I do yesterday? What I will do today to meet Sprint goals? Do I see any impediments that might harm own or teams goals? In the end of Sprint Product Owner invites the Scrum Team and stakeholders to Sprint Review meeting. In the Sprint Review meeting participants will go through all finished items and also those items which for reason or another are not ready. During the meeting completed items will be presented usually by responsible person. In the meeting there should be discussed also following matters; possible problems and resolution of those; situation of the Product Backlog; preparing of next Sprint; and possible changes on market situation, financials or in the schedule (Schwaber & Sutherland, 2011). Continuous learning is one of the important things in Scrum process. End of every Sprint Scrum Master organizes Sprint Retrospective meeting where whole Development Team gathers together.

During the meeting team inspects how last Sprint went regards to people, relationships and tools (Schwaber & Sutherland, 2011). In the meeting team will gather a list of improvable things and plan how to implement those. Sprint Retrospective process guides team to make improvements and create more enjoyable working environment for the team members. (Guang-Yong 2011; Mundra et al. 2013; Schwaber & Sutherland 2011; Dingsøyr 2010)

2.3 DevOps

On IT industry there has been already couple year's vivid discussion about the DevOps and seems that definitions of DevOps are changing as much as there are answerers. Swartout (2012, 1) defines that DevOps is *"a way of working that encourages the Development and Operations teams to work together in highly collaborative way towards the same goal"*. Longbottom (2014) says: *"DevOps – essentially, bringing the development and operational teams closer together through automation to support the speed of change the business requires"*. Both of these professionals are speaking about operations and development teams and bringing those closer together. DevOps is also seen as a combined part of *OPS*, *DEV* and *QA* functions (FIGURE 7 DevOps).

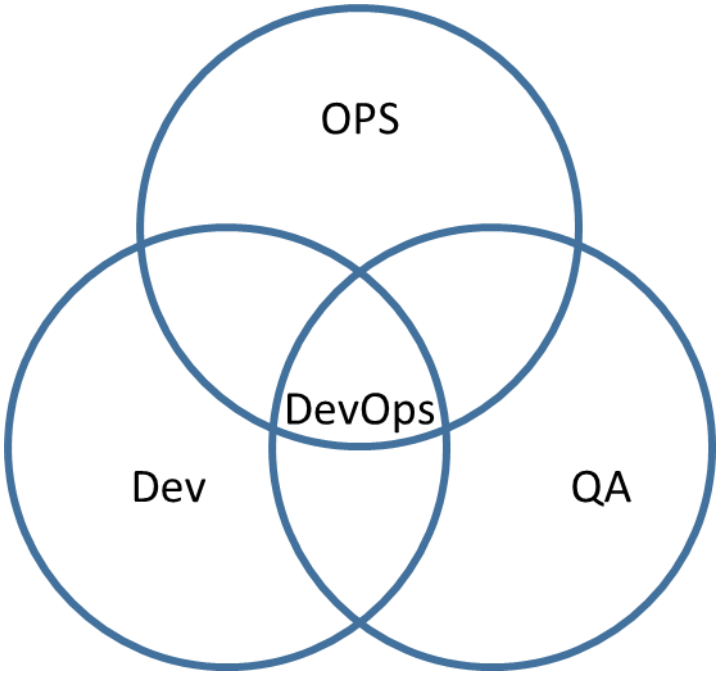


FIGURE 7 DevOps

DevOps is many times connected also for delivering software in quickly pace. Regarding quick pace Humble (2011) says that DevOps is *"a cross-disciplinary community of practice dedicated to the study of building, evolving and operating rapidly-changing resilient systems at scale"*. As can be infered there is no official definition

available for DevOps but the main objective for the DevOps is to bring developers and operative specialists together and the same time break barriers between organizational lines (FIGURE 8 Organizational silos between Development and Operations). With this kind of approach infrastructure management will be part of software development process (Gofore, 2013). On high level DevOps can be shared on two parts: philosophical and practical. In the background of DevOps there are Lean and Agile Software Development. Humble (2011) says that DevOps values are pretty much captured from Agile Manifesto (FIGURE 5 Agile Principles (Agile Manifesto, 2001). Baseline for the DevOps methods comes from Agile Methodologies. In the practice this means that operational functions are integrated to Agile Methodologies. Scrum or Kanban or any other Agile Method is suitable for the DevOps purposes. Continuous Delivery is a term which is often connected to DevOps. Swartout (2012, 1) defines that Continuous Delivery *"is a way of working whereby quality products, normally software assets, can be built, tested, and shipped in quick succession"*. Like some might perceive Continuous Delivery is different thing than DevOps but Swartout sees it as a part of DevOps. (Cois, Yankel, & Connell 2014; Farroha & Farroha 2014; Hussaini 2014; Hüttermann 2012; Spinellis 2012)

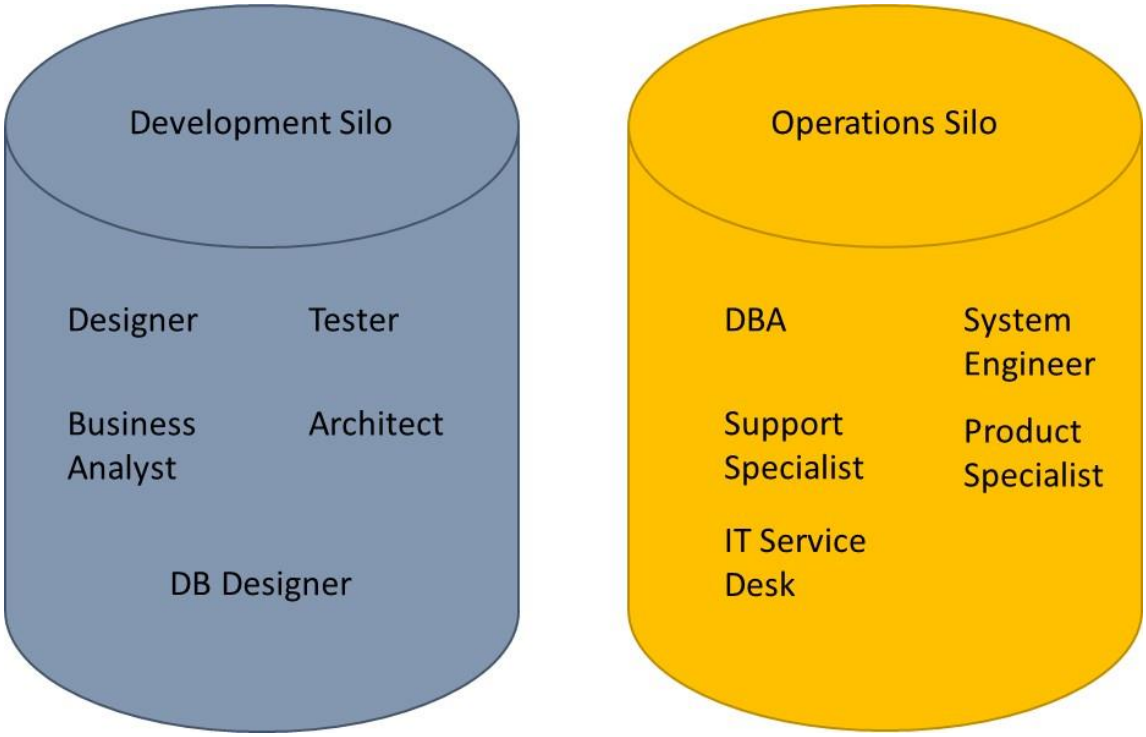


FIGURE 8 Organizational silos between Development and Operations

When going deeper to DevOps can be seen that it includes numerous aspects and activities. In the DevOps these can be classified to following groups: Culture, Automation, Measurement and Sharing (Hüttermann 2012, 4). Like in Agile Methodologies people are acting crucial role in the DevOps. In the Agile Manifesto there is a mention that people should go over processes and tools and same value

applies on DevOps. Therefore Hüttermann (2012, 22) emphasizes meaning of people in DevOps with following words: *"Software is made by and for people"*. Swartout (2012, 71) encourages to open, honest and courageous dialogue. With this he means that everyone involved to the delivery process should be able to comment and openly discuss any ideas, issues, concerns and problems. Hüttermann (2012, 30) is speaking about avoiding so called "blame game". When earlier operations and development teams were apart now this kind of situations shouldn't become as DevOps quides to "one team" approach where all members are aiming to mutual goal. In this kind of "one team" approach programmers, testers, and system administrators are all involved to develop solution together (Hüttermann, 2012, 15). Building trust is also one of the themes what is often spoken in DevOps. All values and practices emphasizes team work and collaboration as a "one team" and by achieving common goals together team starts to build trust between each other's. Also Swartout (2012, 86) emphasizes the meaning of trust with following words: *"Trust is powerful tool and allows for a greater degree of collaboration"*. Hüttermann (2012, 27) defines that respect for one another, commitment to shared goals, collective ownership and shared values are prerequisites for the cooperation and the trust. Swartout (2012, 75) encourages to collaborate but also emphasizes that it needs monitoring as collaboration needs to be continuous way of working. Swartout (2012, 87) describes on his book many layers of CD and DevOps. From description of these layers it is easy to see overview of culture related matters in DevOps (FIGURE 9 Layers of CD and DevOps). (Cois et al. 2014; Farroha & Farroha 2014; Hussaini 2014; Hüttermann 2012; Spinellis 2012)

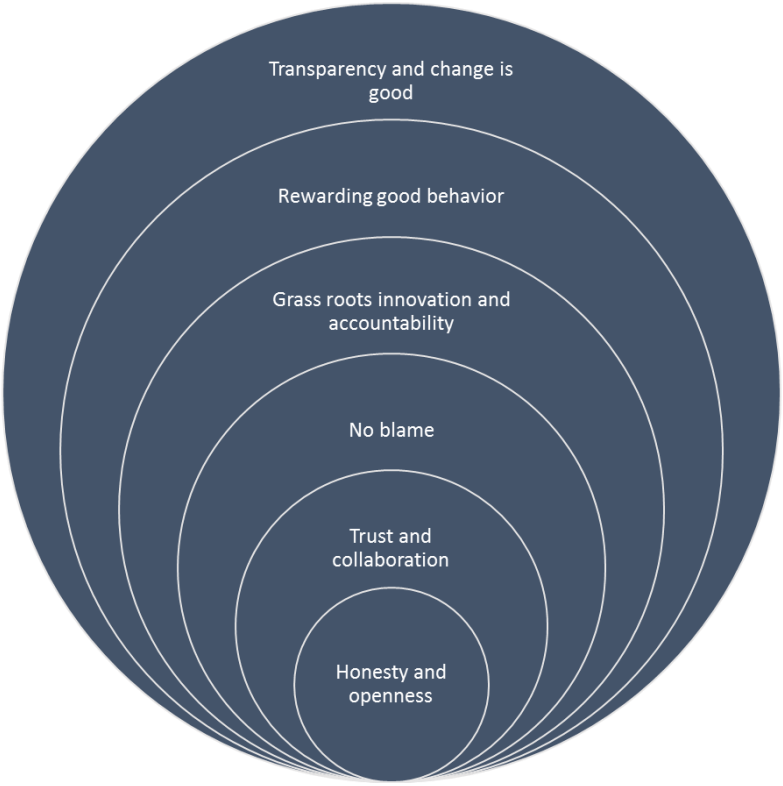


FIGURE 9 Layers of CD and DevOps

As covered earlier in the DevOps there is value: working together as a "one team" and this partially works already when observing to Agile Methodologies where software programmers and testers are seen as a "one team" and they are called developers. So in Agile Methodologies these two group of specialists are already working in a "one team" but DevOps emphasizes that specialist like SysAdmins, DBAs and Network technicians should be a part of this "one team" not separated (Hüttermann, 2012, p. 87). When these functions are isolated from one another it might cause conflicts example during the transition process if development team provides software increment to operations team and there are some nonfunctional requirements which should be fulfilled but those are not considered before increments comes to operations. That's only one example but gives insight why these functions should work as a "one team". This kind of misalignment situation comes from the classic view where development team wants to create change to systems and operations wants to stabilize environments. Both of these functions are working towards their goals so no-one doesn't have made mistake but still there is possibility for a collision. DevOps aims to avoid this kind of possible collision situations by bringing these people together in same team. Bringing people to same team means also changes in culture and way of working. As in previous paragraph was spoken people are in crucial role in the change process. By bringing these two functions together it creates same goals and targets for the whole team which is one of the most important fundamentals of the DevOps. (Cois et al. 2014; Farroha & Farroha 2014; Hussaini 2014; Hüttermann 2012; Spinellis 2012)

Working as a "one team" is a first step for the cooperation but there is still a need for shared processes and tools. The Scrum concentrates on the process which includes several activities. Simplified and effective processes bring visibility and transparency for all involved stakeholders and these same goals are also highlighted in DevOps than in the Scrum. DevOps wants to bring alignment between teams with defined quality attributes (Hüttermann, 2012, 27). The main objective of DevOps still remains; delivering changes to application in fast pace and with quality.

In the Agile Methodologies processes, roles and responsibilities are more important role rather than single tools. In the DevOps tools are also in important role as those helps teams in their daily working with defined processes. The biggest fundamental in the background of tools is high level of automation. DevOps relies on end-to-end automation (Hüttermann, 2012, p. 29). With appropriate tools DevOps wants to apply baselines for the source code; conduct builds; run technical, functional and acceptance tests; packing code packets; deploying to different environments. With this kind of automation and tools DevOps aims to reduce a need for human work and at the same time it decreases amount of possible mistakes in the development process. This kind of approach also creates common practices and it means certain practices are every time done with similar way. Automation can be seen an essential for the high speed delivery.

Swartout (2012, 50) defines that most relevant rules or processes for DevOps point of view are; Always use source control; Commit small code changes frequently; Do not make code overly complex and keep it documented;

If you have automated tests, run them very frequently; If you have a continuous integration (CI) solution controlling your build and test suite, run it very frequently; Use regular code reviews; Do not be afraid of having tests that fail or others finding fault in your code. The idea of source control is to have one common place for all code what team has implemented. Source control gives team an overview of code they have created and at the same time it gives secure environment which includes access for all team members. Usually source control also provides automatic backups for the code and with it team can easily create example different code branches if wanted. Source control can be seen as valuable tool for DevOps adoption and at the same time it creates possibility to deliver small and frequent updates to system (Swartout, 2012, 51). The purpose of delivering small and frequent changes is to reduce complexity and maintain quality (Swartout, 2012, 51). It also helps to reverse-engineer if there are some regressions noticed in code base. As DevOps embraces the high level of automation automated build and testing is one of the most important things in the delivery process. Swartout (2012, 55) recommend Test-Driven Development (TDD) which aims to find possible defects in code as early in the process as possible. In practice it means those tests are written before even the actual code development begins. During the code writing these tests can be run over and over again. During the development process team member can easily notify if something fails in the code. Developers might see TDD slightly cumbersome but meaning of automated testing increases as system grows bigger. Hütterman (2012, 58) spokes about Test Automation Mix which consists UI, Service Components and Unit layers. All these different layers are using automated tools for testing. UI testing layer tests implemented code which consist functionalities in user interface. Service Components layer tests codes in back-end system which usually consist code related on business logic. Units are often classes which can be tested with automation tools. These classes are providing certain functionality which can be used in the one purpose but in a multiple instances. A unit doesn't usually have connection to any database or other systems and those can be tested individually. (Cois et al. 2014; Farroha & Farroha 2014; Hussaini 2014; Hüttermann 2012; Spinellis 2012)

Continuous Integration (CI) is also seen as important in DevOps. CI is software solution that allow team to run automated scripts during the development process which example commits to source control every ten minutes, overnight and so on CI systems provides also usually extensive reporting which means that team members can easily see if something fails in the process and they can trail results, compare to previous and fix possible error in the process. Previously mentioned automated test can also run with CI systems. In practice system can run automated test in binary repository and if something fails it can be traced and fixed in short notice. (Swartout 2012, 56).

Monitoring and measuring the work with certain metrics is one part of software engineering process. As DevOps and Agile Methodologies wants to create visibility and transparency to the whole delivery process is important to have simple and adequate monitoring metrics. In the Scrum there is a Definition of

Done (DoD) which is the exact definition of when task is completed. In measuring context this is first part where team should have well enough documented acceptance criterias. In the DoD team defines what kind of tests and functionality should be implemented before work can be approved. In the DevOps typical metrics for the monitoring are Cycle Time, Lead Time and Throughput. These are telling how well the process works and how long it will take example from customer contact to shipped fix for the reported issue. Cycle Time includes all the sub processes which are determined to do during the process (Hüttermann, 2012, p. 39). Usually this process includes several steps and Cycle Time measures how long it takes to go through all of those. Lead Time differs slightly from Cycle Time as it usually measures time when issue is raised to system until to completion. Both of these metrics are measuring effectiveness of team. In the DevOps usually also amount of shipped functionalities or fixes are measured. Certain amount of fixes or functionalities can be shipped either in bigger or smaller batches which is decided by the product management together with team. During the development process there can be done several different test which are helping team to understand what kind of defects process creates. (Cois et al. 2014; Farroha & Farroha 2014; Hussaini 2014; Hüttermann 2012; Spinellis 2012)

2.3.1 User-centered design

User Experience (UX) is critical part of information system development. Kraft (2012, 1) defines User Experience as a "feelings" that user gets when he or she is using the product, system or service. User experience is a part of information system process which is controlled by using User-centered Design (UCD) methods or practices. User-centered design (UCD) is a "*design philosophy that puts the user of a product, application, or experience, at the center of the design process. In UCD, a designer strives for a detailed understanding of the needs, wants, and limitations of the people who will use the end product and then makes design choices that incorporate this understanding*" (Pratt, 2012, 12). During the User Experience design process team members need to consider many different factors which are affecting the whole. Pratt (2012, 15) mentions that factors like business goals of client, the limitations of technology, timeline and budget will effect on designing process. User Experience is highly connected to user expectations. If user expectations aren't high expectations are easier to fulfill but this works also the other way around, when expectations are high user can easily get disappointed. When speaking about User Experience and User-centered Design process the output of this process is User Interface in products, systems and service. Understanding the user has seen as an important factor during the development process (Pratt, 2012, 16). In practice this means that developer need to understand what user wants and what they need. Pratt (2012, 16) emphasizes that poor design can be frustrating, preventive, and in extreme cases even deadly. From business perspective User Experience of the new products, system and service is the key battlefield (Kraft 2012, p. 16). The ideal User Experience makes user to feel happy, satisfied, proud or even love most of the time (Kraft, 2012, p. 9). Good User Experience starts from

innovation. Innovation can be seen an idea to market (Kraft, 2012, 12). Innovation is often seen as a "big thing" but it is important to understand that many times even small innovation can be meaningful; the main point is to create value for user. Kraft (2012, 12) introduces the five main elements of successful User Experience innovation (FIGURE 10 The five main elements of successful User Experience innovation).

FIGURE 10 The five main elements of successful User Experience innovation

Relevance indicates the need of something. Without relevancy user doesn't do anything with new innovation. Experience of positive feelings is something what makes user enjoy when using the product, system or service. When speaking about uniqueness it doesn't necessary mean that innovation need to be something which is not discovered before but it needs to be something new for the user which creates a feeling of uniqueness. By visibility Kraft means that user needs to experience the innovation and notify this is something new. From business perspective innovation need to have always market. With this kind of approach and considering these elements during the development process team is closer to successful User Experience. (Kraft 2012, 16)

User-centered Design process contains on high level three main factors; understanding users; interaction definitions; and UI design. Many sources are using more or less similar process model but on this case I have chosen model which is presented by SAP (2009) (FIGURE 11 User-Centered Design). As UCD has seen collaborative process between product, system or service providers and users is important to understand the baseline for the process. On planning phase team needs to identify all stakeholders who would be involved in the process.

This group of stakeholders includes both external and internal people. During the planning process relevant UCD activities will be identified. Typical UCD activities are; building of User Scenarios; describing process flows; defining conception outputs (wireframes, UI-proto, site maps); user testing and possible surveys. This kind of planning gives scope for the coming process. (SAP, 2009)

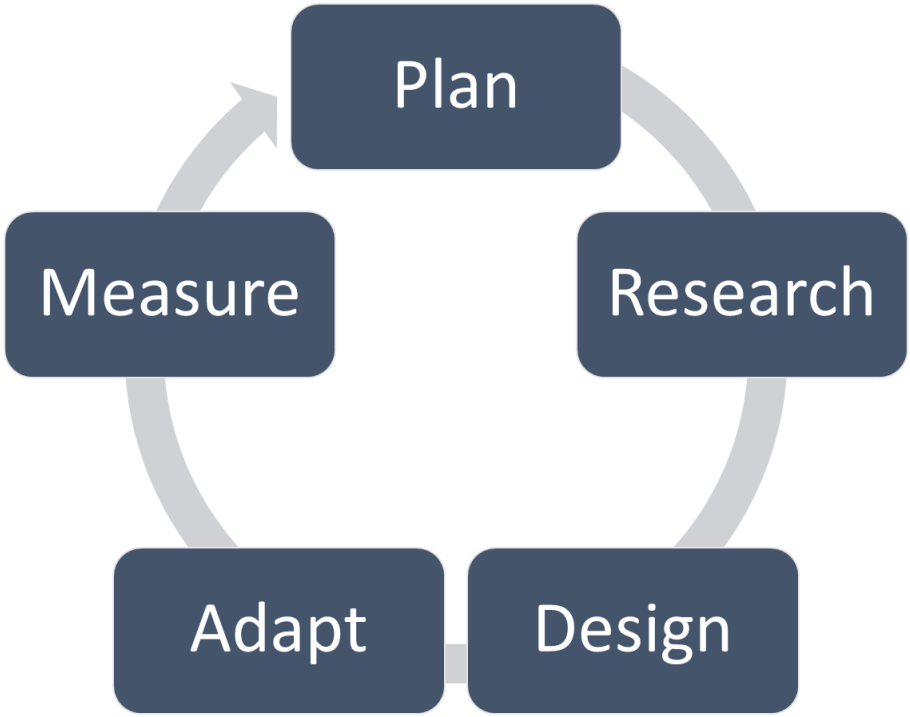


FIGURE 11 User-Centered Design

Research phase concentrates on understanding user needs. On this part of process team defines what the possible user groups are and what kind of different kind of needs these user groups have. User groups and users might differ very much depending on product, system or service market. Example there might be users from elder customers who might have different needs than younger users or in companies different level of organizations might have different information needs. Kraft (2012, 26) sees that "*defining target users is essential step, since it will not only help you focus on innovations, your approach, and your marketing, but it will help you to avoid creating product that tries (and fails) to do everything for everyone*". Typical activities on this phase are; field research (what users want really do); run focus groups or lead user workshops (Kraft 2012, 39)(what users think and how they are responding to new concept); conduct interview; and conduct formative usability test to evaluate design. (SAP 2009)

During the design phase team uses all the information what is gathered in research phase. In system development this phase includes activities like user case creation and user object modeling; UI design sketching (wireframes); and validation of design proposals (SAP 2009). Pratt (2012, 124) highlights that during the design phase it is important to maintain continuous collaboration with users.

In the beginning all sketches and wireframes are blueprints but during the iterative dialogue and development process understanding increases and product, system or service evolve towards customer needs. The design phase can start from sketches, evolve to wireframes and prototypes.

When the design phase end and adaption phase starts it important to understand that evolvement continues. During the adaptation when team members are implementing the product, system or service there might become issues which must be handled proper way. During the adaptation phase dialogue with need to remain and it is important that at the end all implemented things are validated by customer. After designing phase when the actual product, system or service is ready starts measuring phase. On this phase team measures effectiveness, efficiency and satisfaction. All these metric are planned in first phase and results of measurement will be reflected on those. (SAP, 2009)

2.4 Earlier research

Each of main methods and models DevOps, Agile (Scrum) and UCD are topics which have earlier researches available. These researches typically focus on one of these and they are observing method or model in different perspective than I am doing. Earlier researches might focus example for agile testing, challenges of scrum, how to deploy agile methods, how to implement usability testing with agile methods, comparison of maturity models in agile or how to move from waterfall to scrum. Researches about UCD usually focus directly on some part of the UCD model. Example researchers have focused on doing mockups, culture of UCD, interactivity of the process or knowledge based approach for UCD. Uniqueness of this research comes from combination of these three methods which together provide whole method for developing software continuously and more effectively. With UCD this method brings user closer to the development and maintenance work. (Pesonen 2012; Tikkanen 2014; Kamppi 2013; Maukonen 2015; Säde 2004; Iivari 2006; Kalermo 2014; Koskela 2014)

3 RESEARCH METHODS

At the beginning of the research there were two different research methods which would be possible to use in this kind of research. These two different research methods were Method Engineering and Design Science. Both of these methods have good sides but eventually Design Science was more suitable for this case. In the Method Engineering process gathers all requirements and during the process researcher will develop and prototype solution. The evaluation of Method Engineering is done against the original requirements which have conducted at the beginning of research. In the Design Science process researcher constructs a hypothesis as is done in this research. Based on the hypothesis evaluation can be done with different ways. In this research evaluation is done by professionals who are working in the IT industry and they have relevant competencies and experience to evaluate constructed hypothesis.

3.1 Design Science

The background of the Design Science extends to 1960 when first time was talked about systematic form of designing. First opinion leaders during that time were Fuller, Simon and Compton. Design science can be seen as a body of knowledge for designing. Design science provides systematic and formalized design methodologies to research various segments. Design science research aims to produce valid knowledge for designing. The biggest difference between Design science and natural science is their nature. Natural science aims to understand reality and design science attempts to create things that serve humans. Design science can be seen technology orientated and usually with it researchers aims to gain either value or utility. The products of design science include four different types to describe results: constructs, models, methods and implementations. With constructs design science characterizes different phenomenon. Models are used in design science to describe tasks, situations or artifacts. Design science is also used to describe goal-orientated activities. Previously mentioned goal-orientated activities are handled in practice with implementations. Design science can be seen as a research method which helps to create innovative constructs, models, methods and implementations which create value way or another for certain group of people. Design science consist two basic activities, build and evaluate. Building phase constructs artifact for a specific purpose. The goal of evaluation process is to understand how well artifact actually performs in that context where it is meant. (March & Smith 1995)

Peffers et al. (2006) have built in their research a process model for the design science (FIGURE 12 Design Science Research Methodology (DSRM) Process Model). DSRP model includes six activities in nominal sequence. First, problem

identification phase defines the problem behind the research. In this phase researcher defines the problem and divides it to small enough pieces. When problem is described comes motivation part. In motivation part researcher describes why it is important to solve this particular problem and why it is worth of researching. On a second phase researcher defines objectives of solution. This part of process describes what desirable solution is and how it solves the problem behind the research. In this phase researcher answer also to question: why solution is better example than some previous solutions. This phase gives goals and objectives for the research. On a third phase researcher creates artificial solution. As previously mentioned design science consist four different types to describe results: constructs models, methods and implementations. These are used to describe the artificial solution. Fourth phase demonstration is meant for proving the efficiency of the solution. This phase creates understanding how well the created solution actually works with research problem. For this kind of demonstration can be used example simulation, case studies or any other suitable activities. Second last phase evaluation observes and measures how well solution solves problem. Evaluation is done against the original problem definition. Depending on research researcher need to choose relevant tools and metrics to evaluate how well the solution suits for problem solving. Researcher can use either quantitative or qualitative metrics for providing relevant information for the research. Typical implementations are surveys, feedback from relevant people or simulations. Last phase in this process is communication. When research is done and described with common structure of research papers it will be shared to relevant parties like practicing professionals. (Peffers et al. 2006)

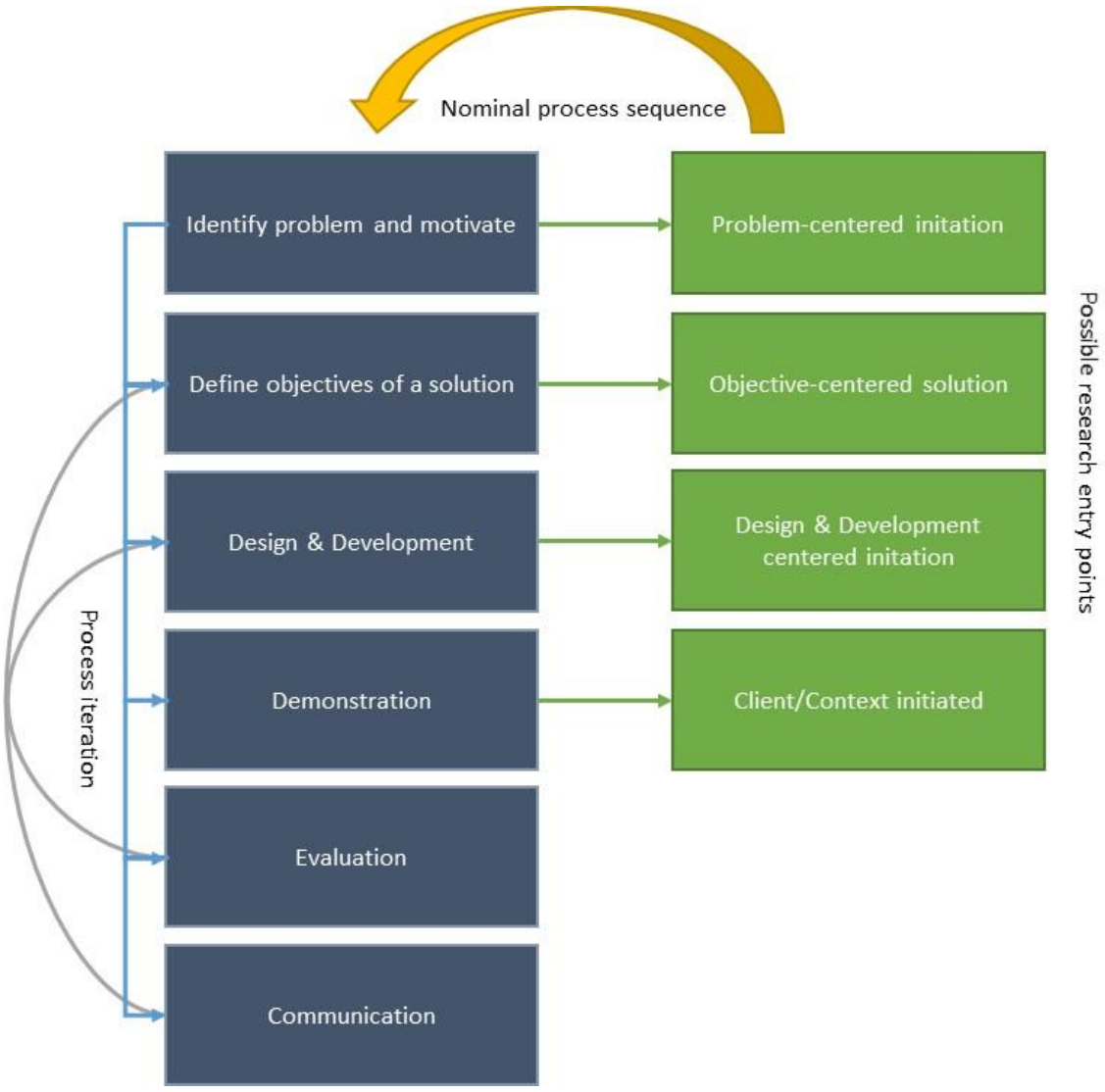


FIGURE 12 Design Science Research Methodology (DSRM) Process Model

3.2 Approach for qualitative research

Qualitative research will include approximately five interviews. On these interviews researcher introduces the created method material (Attachment 1: Interview Material) and each of participants will answer to open questions which are evaluating how method would work in their working environment. Each of participants in these interviews are professional from the information technology industry. In the interviews there will be 2-3 persons at the same time and answering will be done in a group. Purpose for these interviews are to gather information for method values, principles, rules, processes, roles, responsibilities and tools (Attachment 1: Interview Material) and in later phase breed and updated method based on the interview results.

3.3 Design Science use in the research

This research will follow the Design Science process. At the chapter 1.1 it is described the problem for the research. Motivation for the research is described in the chapter 1.2 and objectives are available in the chapter 1.3. Designed theoretical method is introduced at chapter 4. Theoretical version of method will be demonstrated to IT professionals who will also evaluate the hypothesis of method. After first evaluation round method will be iteratively developed as described in the Design Science process. First evaluation round will be based on face-to-face interviews and after iterative development another group of IT-professionals will re-evaluate the method with survey. Results of first evaluation round are available at chapter 5. After first evaluation and analyze method will be updated and next version of method will be evaluated with the survey. Survey will include numerical evaluation of method. In this survey answerers will different persons than in the first evaluation phase. Next version of method and survey results are available at the chapter 6 and 7**Error! Reference source not found.** (FIGURE 13 Process phases in the research)

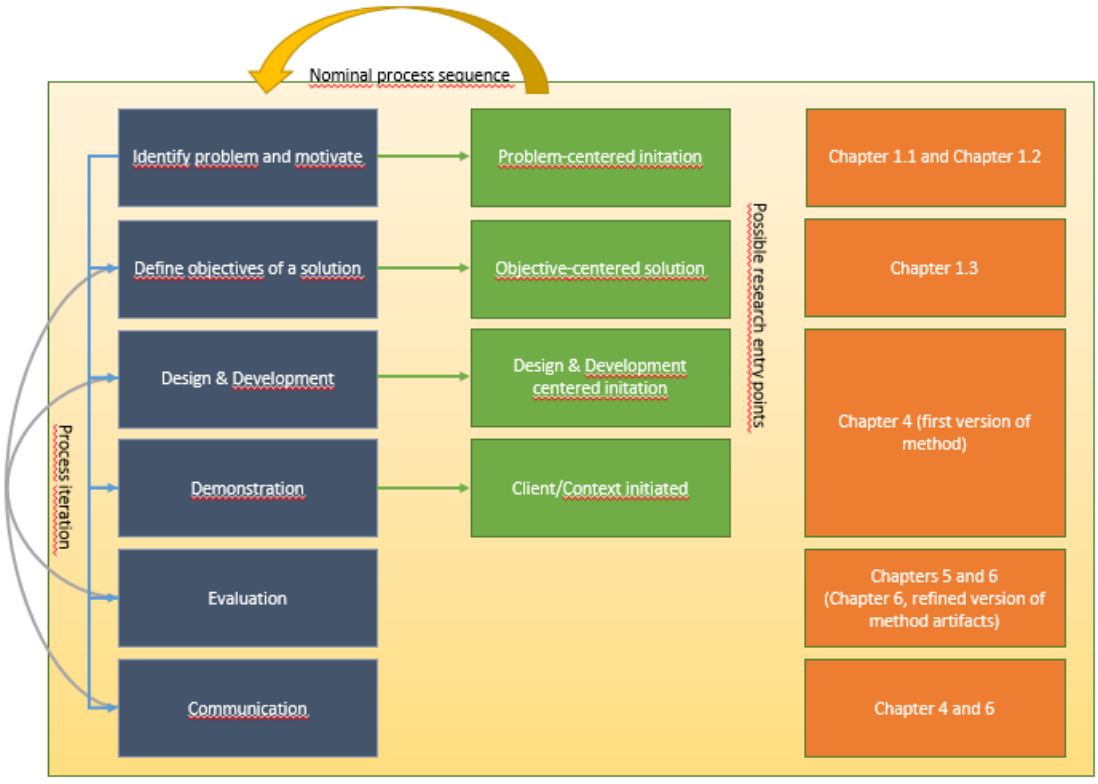


FIGURE 13 Process phases in the research

3.4 Background of methods (software development)

Methods usually include guidelines, processes, roles, responsibilities and other relevant information to organized way of working. Jayaratna (1994, 35) defines that method is *"an explicit way of structuring one's thinking and actions. Methodologies contain model(s) and reflect particular perspectives of 'reality', based on a set of philosophical paradigms. A methodology should tell you 'what' steps to take and 'how' to perform those steps but most importantly the reasons 'why' those steps should be taken, in a particular order."* In many sources term "methodology" is synonym for the "method". Brinkkemper (1996, 275-276) defines method as "*an approach to perform systems development project, based on a specific way of thinking, consisting of direction and rules, structured in a systematic way in development activities which corresponding development products*". Methods in operative work can be seen as backbone for the IT operative work. Methods have always certain structure of information. This structure describes relevant information components and helps us to understand what kind of information is needed when creating method for the IT operations. Leppänen (2005) defines on his research method context with following model (FIGURE 14 Context of Method). On this model method consists following components; Historical view, Application view, Generic view, Contents view, Presentation view, Structural view and Physical view.

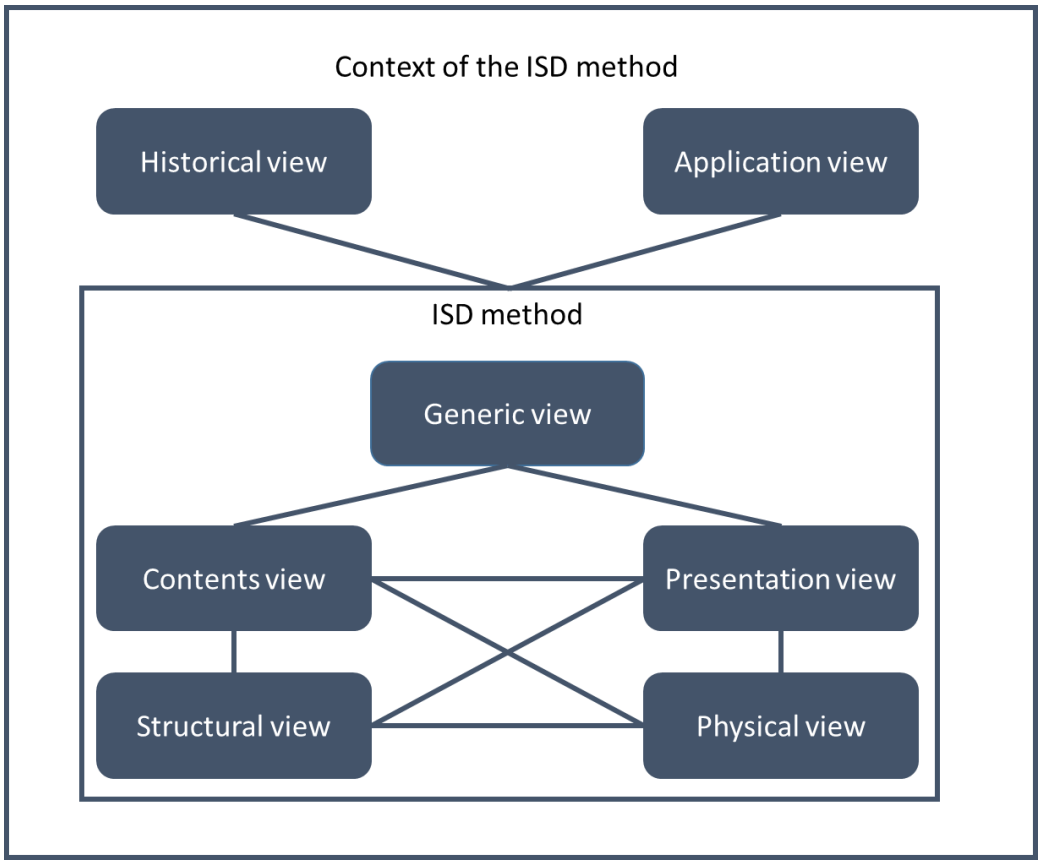


FIGURE 14 Context of Method

All of these different views (FIGURE 15 Descriptions of different views) have their own characteristics and each of those complements one another. For the research most important views are application, generic, contents, presentation and structural views.

Name of view	**Description of view**
Historical view	Describes the background of method
Application view	Describes also where and how the method is applied.
Generic view	Nature of method, fundamentals on background like values and assumptions.
Contents view	Conceptual contents of the method
Presentation view	Describes how method is expressed to understandable mode and how it is presented
Structural view	Structure of method, including principles, models, techniques, rules, guidelines etc.
Physical view	Describes how method appearance and how it is available for the reader

FIGURE 15 Descriptions of different views

4 METHOD FOR CONTINUOUS SOFTWARE DEVELOPMENT AND MAINTENANCE

4.1 Background of method

This method gathers, combines and tailors values, principles, rules, roles, processes, practices and techniques from the Scrum, DevOps and UCD. All selected methods fulfill each other in this method. On a high level Scrum framework creates a baseline for development work and DevOps brings maintenance activities and practices together with agile methodologies. User-centered design gives end-user and usability aspect to the method. As a whole method for continuous development focuses to *develop a method for continuous development and maintenance of software, which meets customer and user expectations*.

4.2 Application of method

This method is meant for teams which are providing software development and maintenance for consumers or/and customers. Agile background of method provides a way react more quickly and respond more accurately to the inevitable change that comes from consumers and customers. Especially method works with complex software where targets aren't always clear during the software lifecycle. Method includes practices to work in fast pace environment and it is designed for developing software continuously with high automation level. Method also takes into consideration user experience and usability of provided software. Method consists following structure in this document:

- Common values and objectives
- Principles, rules (and practices)
- Roles and responsibilities
- Processes
    - Events/meetings/activities
- Tools

4.3 General

4.3.1 Values

Agile Manifesto provides values for Scrum framework but there are no common known values available for DevOps. As DevOps relies on Agile Methodologies

values remains quite same but it would be good to have also operative perspective for the values. The method combines UCD and Agile Manifesto with relevant parts. (Agile Manifesto 2001).

On this method there are divided values for three main categories (FIGURE 16 Values). These categories are **collaboration**, **quality** and **delivery**. Both Scrum and DevOps are highlighting the meaning of collaboration within the team. UCD has also same kind of values but it highlights the meaning of end-user/customer. In the values one team means same time working as a team. If previously hardware admins and software developers were separated on this method situation is changed and those specialist are brought together in one team. Communication is crucial part of software development and that reason it is one of the sub-values. During the software development things are changing all the time and without researching/planning together and sharing information software development will become more difficult. Therefore it is very important to share information and communicate with each other. It is also important to make information visible for all team members. Commitment is one of the sub values. Purpose for this value is to make sure that everyone commits on their work as every piece and every part of software matters in the end-result. Committing to deliverables strengthen the team as everyone can see that whole team is working towards common goal. (Dingsøyr 2010, 203-233; Swartout 2012, 22-26, 75; Kraft 2012)

One of the main values is a quality. Under the quality there are four different sub-values: provide excellence, embrace working software, user comes first and measure and evolve. With these values method aims to create best possible software which works all the time and same time it is user-friendly. As one of the main ideas of method is continuous improve it is important to measure how well software is working. By gathering feedback from end-users team can understand their needs. By using the acquired information team can improve the software. (Agile Manifesto 2001).

Third main value is delivery. Delivery section includes also four sub-values which highlight continuous development and delivery. It also considers automation as automation creates framework to deliver software in fast pace. When talking about measuring quality same thing applies on delivery side. It is vital for the team measure how well they are performing and how well software is performing. Therefore measuring and growing knowledge that way helps team to improve developed software and their practices during the software´s lifecycle. (Swartout 2012; Agile Manifesto 2001)

Method value chart:

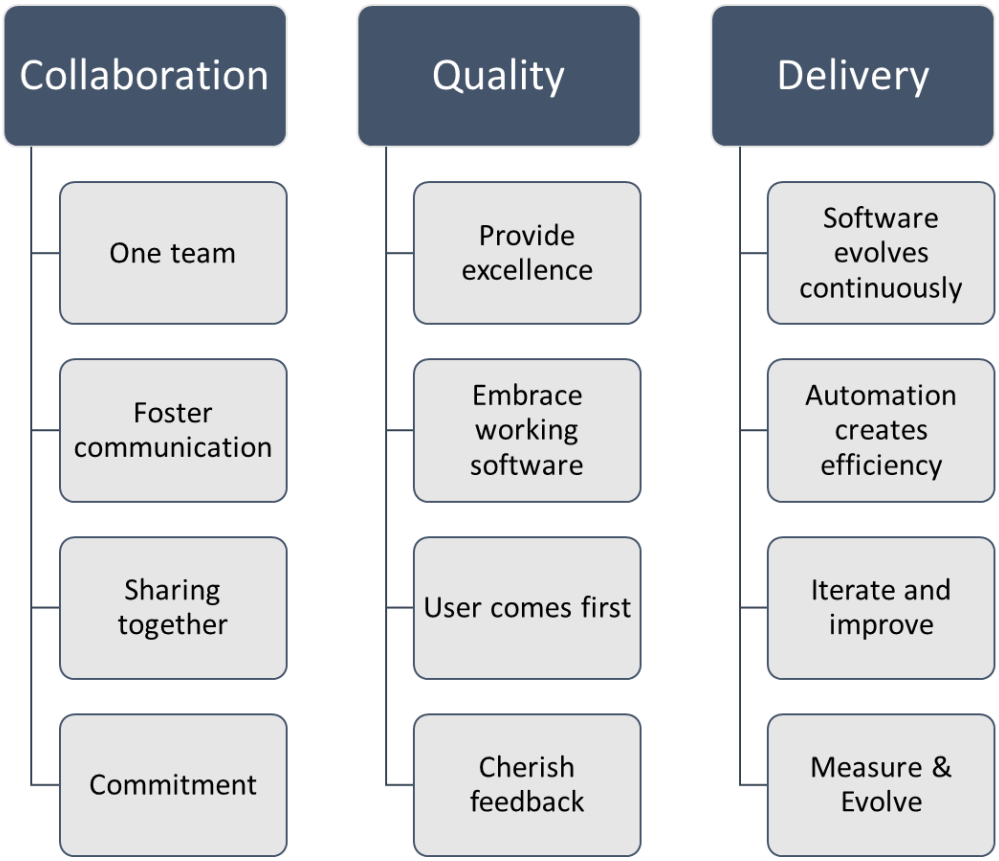


FIGURE 16 Values

4.3.2 Objectives

All researched methods have their own objectives. In the created method objective is to combine best parts of these common methods in one model which would get benefits from all of them (FIGURE 17 Objectives). Scrum and agile methods are aiming for following things; managing risk and change, faster time to market, improved quality, stakeholder satisfaction and higher productivity with lower costs. Slicing the scope for smaller parts helps software teams to understand the each part in development better than previously. Process forces teams to communicate with particular way which helps everyone in the project to stay up-to-date. One of the main things in Scrum is faster time-to-market. Everyone who is working with the business understands that every month what company is not in the market with new product they are losing potential money. Scrum aims to produce minimum sellable product which can get market faster than with the older software development methods (e.g. waterfall). When the new product is on the market cash flow starts and it helps company financially to invest on further development. Scrum aims also to improve quality and mainly this happens by formalizing processes. As a process of scrum is easy to use it creates the effectiveness for the IT teams. Same time it forces stakeholders to participate in the process whereas helps information sharing and collaboration between stakeholdes. Scrum as a process can be compared to train line; it always

follows same track, delivers new shippable increments and when it is at the end point it starts again from the beginning. This kind of approach makes it also effective and improves productivity and lower costs. (Swartout 2012; Hüttermann 2012)

DevOps and Scrum completes each other as they have many same goals. DevOps also aims to faster time to market with high automation. These methods gives a tools and practices to provide faster software and also update software in faster pace than previously. If Scrum aims to reduce risk with certain process DevOps aims to reduce risk mainly with tools and with certain practices and cultural factors. When as many as possible parts in the process are automatized it helps whole team to develop new version and ship those to production faster than before. Shipping software faster than previously lower costs also. If certain parts of process took previously two or three days with DevOps this part of process can be handled with correct tools and practices even ten times faster. Additionally DevOps aims to effect on culture. Previous goals and objectives have concentrate on tools and business matters but it also aims to bring people together in team. (Swartout 2012 72-74; Hüttermann 2012; Schwaber & Sutherland 2011)

In previous chapters covered goals have been related on following questions; what kind of process need to have when creating software; how development can be done effectively; how all these can be done in a fast pace? UCD goals are more near to user who actually uses the delivered software. UCD aims to design software for the users and for their duties. UCD aims to consistency which means that provided software is easy and logical to use in every situation. By using UCD in the method gathering requirements phase ease to team members. From business aspect UCD aims to improved customer satisfaction and eventually increase revenue. Summing up goals for this method includes six main goals; Improved collaboration, better customer and end-user satisfaction, lower risks, faster time to market, improved quality and higher productivity with lower costs (FIGURE 17 Objectives). (Kraft 2012)

FIGURE 17 Objectives

4.3.3 Principles, guidelines and rules

The fundamental principle of developing software with agile methods starts from iterative development. Iterative development creates a baseline for all development work in this method. When doing software development all iterations and activities need to have a time-boxes. In practice this mean example that team has to have together agreed amount of days/weeks how long iteration will last before next one starts or sprint planning takes two hours. Typically iteration length is between 2-4 weeks. With prioritization product owner chooses those backlog items to development which creates most value for users and for the business. Collaboration is one of the main things in the whole method. It is highlighted in many places already in the method structure. Collaboration can be seen as a backbone for the whole method. It needs to be considered in many places and there need to be culture of which encourages for open and honest communication within the team and with stakeholders. If earlier developing teams have leaded strictly by each person nowadays that situation has changed. This method emphasizes the meaning of each individual self-organizing. In practice it means that each member will proactively take tasks and aims to do best in every situation. Strong results will follow when every person in the team will do their best, everyone follows the rules (FIGURE 19 Rules) and whole team is aiming for common target. One part of collaboration is feedback. Getting feedback from stakeholders and inside the team creates baseline for growing up. Whole method is

structured to follow certain process. It means that without following process things doesn't work as planned. Therefore it is important to follow the process. This method doesn't go details in that sense what actually happen inside particular activity but it creates baseline for the continuous development. As one of the main ideas of this method is also to focus end-users principle chart includes also principles regarding end-user experience and usability. Developing starting point need to be in users and their feedback. When team understands the end-user needs and they are designing and developing software for them team has good starting point. When designing usable software everyone need to follow consistency on designing. As a part of this consistence team is good to have guidelines for doing user interfaces which gives framework for designing. In many cases less is more and it also works in software. Therefore one of the principles in this principle chart (FIGURE 18 Principles) is "keep it simple". In practice it can mean that example when developing new feature developer together with UI/UX specialist will think what is the easiest way to implement this new feature? How amount of clicks during the process can be decreased? When developing user-friendly software if possible take end-users part of the development work. As earlier covered the collaboration and meaning of it but in this context it is important to find ways to get feedback from users. Getting feedback creates baseline for continuous improvement. Improving continuously needs solid structure behind of development. It means that system architecture and whole ecosystem of several products need to be created to support these functions. When fundamentals like architecture, common techniques and environment structures are well planned it is easier to automatize certain parts in the process. Automation improves the pace and quality, lowers costs and it also makes team life easier as manual work in the process has decreased. (Swartout 2012; Hüttermann 2012; Schwaber & Sutherland 2011; Poppendieck & Cusumano 2012)

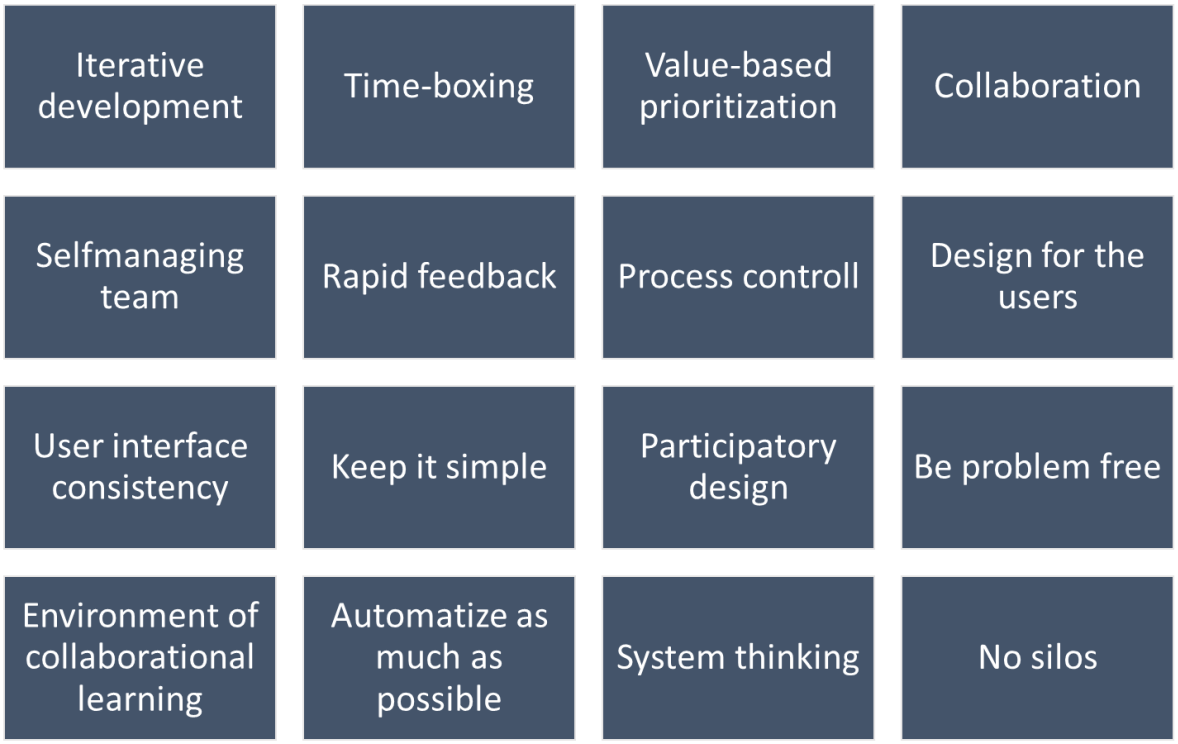


FIGURE 18 Principles

FIGURE 19 Rules

4.4 Structure of the method

4.4.1 Roles and responsibilities

On this method baseline for the roles comes from the Scrum. Primary target was combine several methods in one so therefore in method there will be some new roles which doesn't are part of Scrum framework. As one of the main goals has been bringing development and maintenance specialist to working in same team that is something what has been taken into account. Target was also raise up meaning of user experience and usability this method includes roles for UI/UX specialists. (Hüttermann 2012; Mundra et al. 2013; Schwaber & Sutherland 2011)

Responsibilities of each role are listed on a high level on roles chart (FIGURE 20 Roles) and more detailed level in responsibility matrix (FIGURE 21 Responsibility matrix). Depending on size and broadness of business one person might have several roles. Example one person might be developer and architect/tech lead at the same time. Also team structure can include several members with same role. Example team can have 5-7 developers. Team roles and responsibilities might change during the time and depending on business but more important is to keep up-to-date list where these are documented.

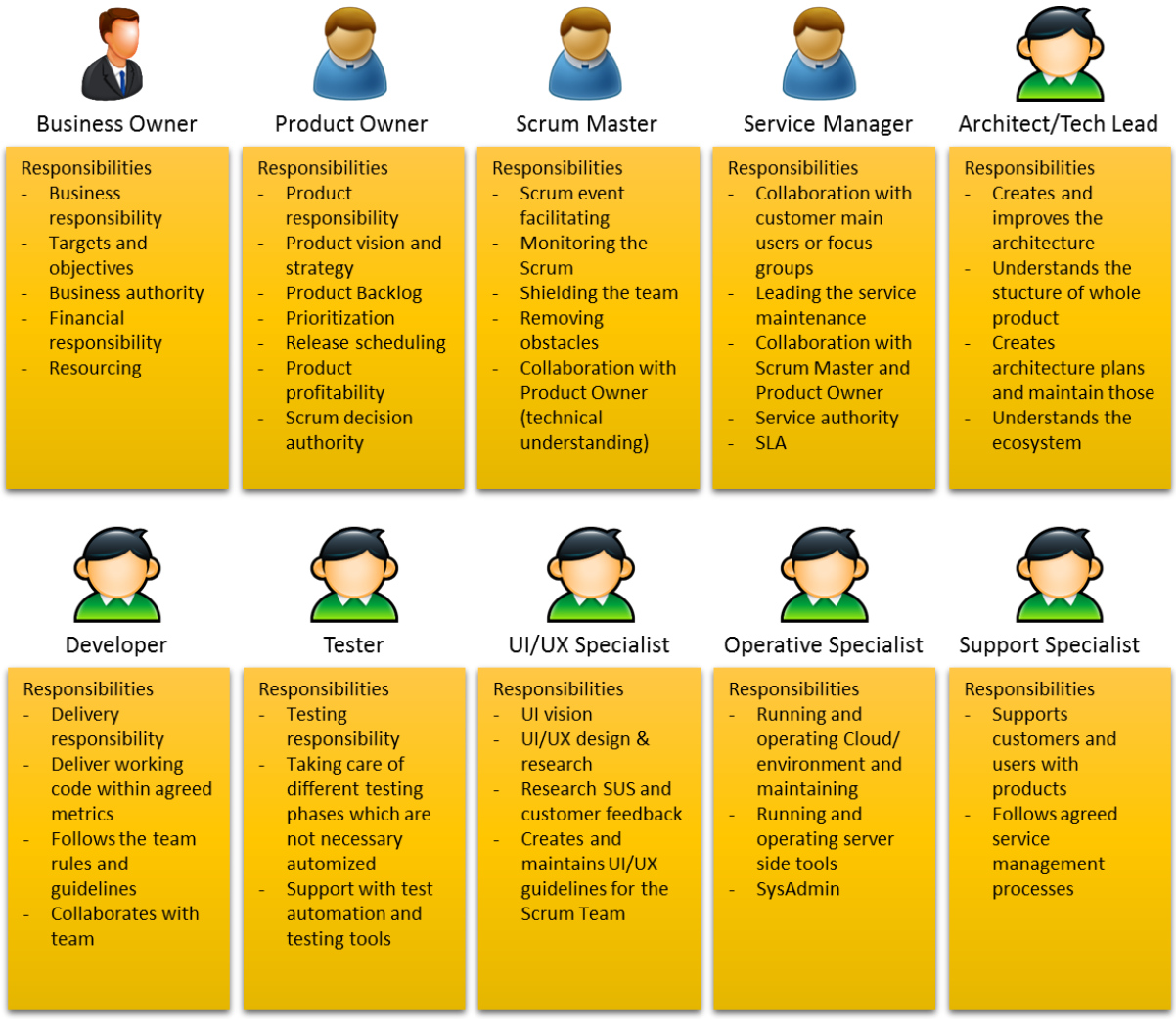


FIGURE 20 Roles

Responsibility matrix	BO	PO	SM	Serv. Mgr	Arch./Tech Lead	Dev.	Tester	UI/UX Spec.	Op. Spec.	Sup. Spec.
Provide strategy and objectives	R/A	C	I	I	I	I	I	I	I	I
Business authority	R/A	C	I	I	I	I	I	I	I	I
Contracts	R/A	C	I	I	I	I	I	I	I	I
Ensure consistency of development practices	A	C	R	I	I	C	C	I	I	I
Provide vision, goals and context for the product	I	R/A	I	C	C	I	I	I	I	I
Provide resources with right skills and mindset	R/A	I	I	I	C	C	I	I	I	I
Gather business requirements	C	R/A	I	C	I	I	I	I	I	I
Gather functional requirements	I	R/A	I	C	I	I	I	I	I	I
Architecture design	I	I	I	I	R/A	C	I	C	C	I
Prioritize and manage the backlog	I	R/A	C	C	C	C	I	I	I	I
Creates design guide	I	C	I	I	C	C	I	R/A	I	I
Plans and design feature wireframes/mockups	I	I	I	I	I	C	I	R/A	I	I
Remove impediments and obstacles	R/A	C	R	I	R	R	I	I	I	I
Make sure development practices are used	C	C	R/A	I	C	R	I	I	I	I
Create, apply and improve Definition of Done	C	C	R	I	C	R	C	I	C	I
Create, apply and improve Definition of Ready	C	C	R	I	C	R	C	I	C	I
Define acceptance criteria	C	R/A	R	I	C	C	C	R	C	I
Write/Perform component, system, unit and security tests	I	A	R	I	C	C	R	C	I	I
Write/Perform UI/UX tests	I	A	R	I	C	C	C	R	I	I
Write/Perform acceptance tests	I	A	R	I	C	C	R	C	I	I
Ensure quality of the product	C	A	R	C	C	R	R	C	I	I
Manage risks	I	R/A	R	I	C	C	I	I	I	I
Code review	I	I	I	I	C	R/A	I	I	I	I
Releasing and building	I	I	I	I	C	C	I	I	R/A	I
Approve user stories	C	R/A	R	I	C	C	I	I	I	I
Define coding patterns	C	I	I	I	R/A	R	I	I	I	I
Define component and module interactions	C	I	I	I	R/A	R	I	I	I	I
Define problem handling	C	C	R	C	R/A	R	I	I	I	I
Creating user stories	I	R/A	R	C	C	C	I	I	I	I
Sprint planning and task breakdown	I	C	R	I	C	R/A	I	I	I	I
Daily scrum meeting	I	I	R	I	I	R/A	I	I	I	I
Sprint demo	I	A	R	I	I	R	I	I	I	I
Sprint retrospective	I	R	R	I	I	R/A	I	I	I	I
Training plan	I	C	I	R/A	I	I	I	I	C	I
Communication plan	R/A	C	I	C	I	I	I	I	I	I
Hosting and deployment	I	C	C	I	C	I	I	I	R/A	I
System, network and monitoring tools	I	I	I	I	C	I	I	I	R/A	C
Issues management	C	C	I	R/A	I	I	I	I	I	C
Issues gathering and support	I	I	I	C	I	C	I	I	I	R/A
System monitoring	I	C	I	C	C	C	I	I	R/A	I
User surveys	C	C	I	R/A	I	I	I	I	I	I
Arrange focus groups	I	C	I	R/A	I	I	I	I	I	I

FIGURE 21 Responsibility matrix

4.4.2 Process

As discussed in chapter 2.1 there are usually pretty similar phases while developing software. Always when designing new software there need to be a planning phase where team members can design incoming features or bug fixes etc. The most important thing in this phase is that all team members have common understanding of the scope and everyone understands the content of work packages which are included to the next iteration. When planning phase ends and team have common goal for the next iteration the development phase starts. Development phase includes practical code implementation but also many other relevant activities which team needs to take care about. When the iteration ends team has been able to provide working increment to the software they were developing. This particular increment is always a part of some production release but production release might happen in different cycles than iterations. Example one production release might contain working software increments from three different development iterations.

On a high level the process is divided on a three different main phases which are plan, develop and maintenance. Inside of these phases have own sub-phases. In the planning phase there are two sub-phases: vision and planning. Development phase contains development, quality assurance and end of iteration sub phases. In the running phase there are possible release and operating sub phases. All sub phases includes one or more activities which have own certain purpose in the process (FIGURE 22 Process Model). Activities can consists example event or some practices but in this method those are called activities. In the next paragraphs there are introduced activities and example tools which can be used in the process. It is important to understand that chosen tools will be applied in whole team which covers all different phases in software lifecycle.

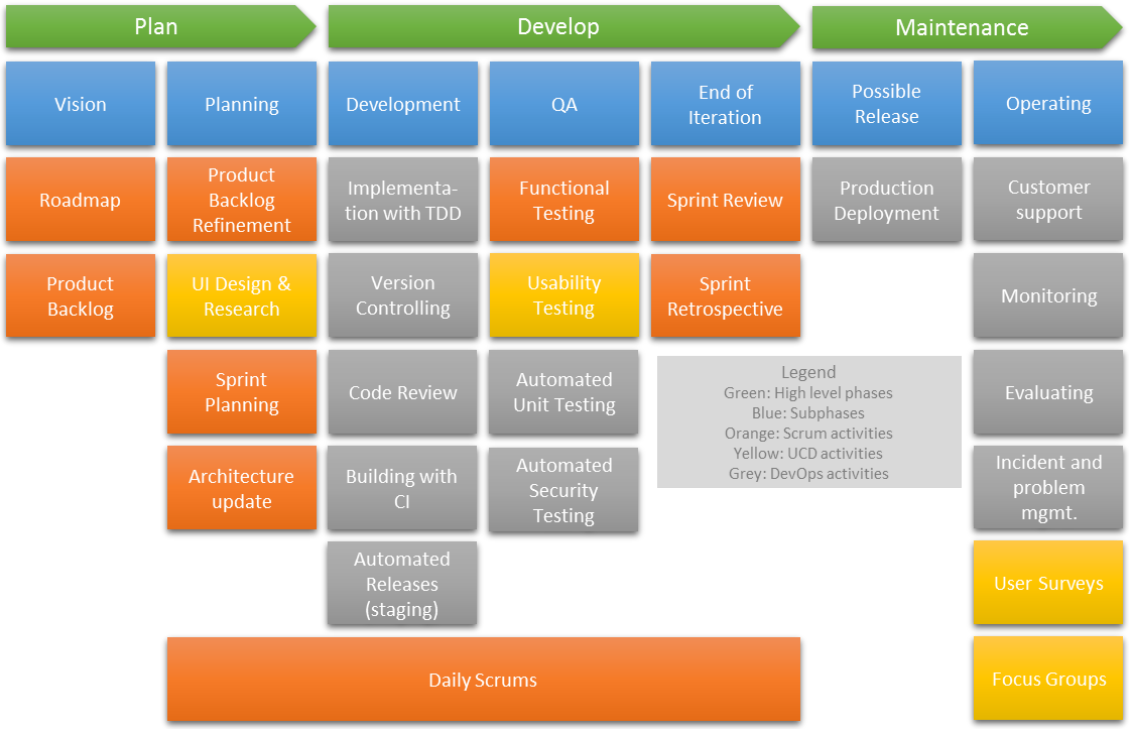


FIGURE 22 Process Model

4.4.3 Planning phase

Vision sub-phase includes a roadmap activity. The roadmap is a document which describes high level features which are coming to certain product. Roadmap can change during the time and prioritization of items in the roadmap changes depending on business needs. Items in the roadmap are described on a high level and that does not typically cover all technical details. The roadmap gives insight to the team members and also to customer and both of them can see what is coming in the future. The roadmap can be seen as a crucial document for product management which gives a structure for future development. Product Owner is responsible for the roadmap. Inputs to the roadmap come from the business and

customers and it provides outputs to the product backlog. (Mundra et al., 2013; Schwaber & Sutherland, 2011)

The product backlog is one of the scrum artifacts. Product backlog includes all features and items which will be developed in the future. Product backlog is the main tool for the Product Owner. Product backlog includes user stories which describe functional requirements for the new features. Roadmap is one of the main sources of items in product backlog but product backlog handles these items more accurately and more detailed level. Items from product backlog are refined in product backlog refinement. This event is organized by Scrum Master which prepares items from the product backlog for the coming iterations or sprints. During the refinement team defines product backlog items in level which fulfills the definition of ready. Definition of ready means that item in the product backlog is described at least in a level which covers all requirements in DoR. There might be requirements like: User story must be available, test cases must be defined, UI wireframe or mockup must be available, estimations must be available etc. At the same time when product backlog refinement is done UI/UX specialist will concentrate on UI design and research. This process will provide UI/UX related requirements and specifications to the team. As an output from UI design and research activity there will be process flows and wireframes or mockups available for the development team. There might be also UI-prototype which shows how certain functionalities should work in the product. UI-prototypes are usually done in rough level and it evolves in the process. UI-prototypes work also well when product management have close cooperation with stakeholders which are using the system. This way Product Owner and team can get valuable information from stakeholders and relevant people. (Guang-Yong, 2011; Mundra et al., 2013; Schwaber & Sutherland, 2011)

When team has refined product backlog available and all prioritized backlog items fulfills the DoR team can start to plan next sprint. Sprint planning is an event where team checks how much they have resources in use in next sprint and how much each team member has allocated time for development. Each team member has certain capacity to use for a development purposes. When Scrum Master has understanding of resources team plans together with Product Owner which items they will choose from product backlog to the next sprint. Team picks-up as many items from the product backlog as they can do during the sprint. During the sprint planning team commits to the work they have chosen for the next sprint and the same time they have responsibility to deliver agreed items for the next software increment. As an output from the sprint planning event team has sprint backlog. Sprint backlog includes all items which team will implement in agreed timeframe. Usually timeframe of one sprint might change from one week to four weeks depending on team. During the sprint team and Product Owner monitor progress against sprint backlog. Typical chart for the monitoring is a Burndown Chart which shows graphically how much work is left versus time. Horizontal axis describes the timeframe and vertical describes the amount of estimated work in the sprint. All sprint work is eventually split to

tasks for each person will handle tasks one by one during the sprint. (Guang-Yong, 2011; Mundra et al., 2013; Schwaber & Sutherland, 2011)

4.4.4 Development phase

Development phase starts from developing work which got inputs from the planning phase. Software implementation happens in decided development environment. Team can use example together agreed IDE (integrated development environment) while they are completing their tasks from the sprint backlog. During the programming process team complies with together agreed coding conventions. With coding conventions team ensure that programming principles are followed. This kind of approach increases the quality of code and drives down complexity. In every phase of software lifecycle it is important to follow together agreed coding conventions as there will be phases where developers need to refactor and renew or change source code which might be produced even years ago. Without having common rules in programming handling change request and implementing new features will be more laborious than if team has complied with coding conventions. During the programming work developers will follow rules of TDD (test-driven development). In practice this means that developers will write test cases before writing the actual part of code. This kind of approach increases quality and creates a baseline for automated testing. Benefits of TDD are coming more visible when system grows. TDD with automated tests provides teams ability to launch new releases in quick pace when testing phase doesn't take so long time. In software development process teams need to use version controlling systems. Version control systems provide control over changes to source code. With version control team can manage different source code branches and merge branches to code baseline. Version control must be integrated to the used IDE and continuous integration tool. With integration continuous integration tool can automatically deploy newest committed changes to chosen environments. This provides a practice where example several persons can work in same software environment and commit new changes to version control and continuous integration tool will automatically deploy done changes to testing, pre-production environments which allows to process continue smoothly to the next steps. In the process there is also step called code review. This activity aims to find and remove vulnerabilities and possible mistakes in code. In code review activity it is recommended to use automated code review tools which can be integrated example to IDE. Code review and rules can be written directly to the decided tools. This activity provides higher code quality and it also heads to solid source code. (Cois et al., 2014; Farroha & Farroha, 2014; Guang-Yong, 2011; Hüttermann, 2012; Mundra et al., 2013; Schwaber & Sutherland, 2011)

Part of development work method includes activities which effects on quality like code review and TDD. Reason why previously mentioned activities are under development phase is their tight bond with developing work as activities under QA are timely positioned to later phase. In QA sub phase team will

run different kind of test combinations. Functional testing test covers the actual functionality of new features. During the product backlog refinement process and part of DoR team defines test cases for the functional testing. When development team has done their development work testing specialists will test delivered functionality against these specifications. Usability testing could be also part of functional testing but in this method it is highlighted to be own separated activity in the process. As usability and user experience is highlighted in the method the process provides possibility to end users use system before official release. Usability testing covers internal usability testing where own personnel tests usability and user experience against UI/UX specifications which were done in planning phase. In addition automated releasing process with CI tools will provide newest releases to dedicated environment for focus groups that can test new features before the official version release. During this process they have also possibility to provide feedback to the development team about possible improvements. Automated testing includes test sets example for unit, regression and sanity testing. Automated testing can cover testing for different components like units, UI, security and interfaces etc. (Cois et al., 2014; Farroha & Farroha, 2014; Guang-Yong, 2011; Hüttermann, 2012; Mundra et al., 2013; Schwaber & Sutherland, 2011)

4.4.5 Maintenance phase

At the end of development phase there is new software increment available. Depending on release cycle team can release new version or not (requires approval). Software increments might have dependencies to previous or next increments and therefore is not always reasonable to release newest software increment to the production environments. Software increment might include also parts which might belong to bigger epic in product backlog. In practice this means that product management has decided to create larger package of new features which compliments each other. This kind of situation means that there is no sense necessary to provide partial feature set to customers and product management has decided to launch whole package in one bigger release which contains several software increments. During the process continuous integration tool is maintaining automatically different environments but production updates need to be handled with manual trigger. New releases to production environment include larger testing set before the new release can be launched. Team will provide release candidate version example to the pre-production or staging environment where testing specialists will complete their testing set. Typically this kind of testing set includes end-to-end testing where example all interfaces all tested.

In the operating phase there are several activities which need to be handled. Team will monitor all environments which are needed in various activities. Typically there are software environments for development, testing and production purposes. All environments need to be up-to-date in all circumstances. In practice this means that all necessary operating system, security and other required software need to be up-to-date. Monitoring activity includes also moni-

toring for networks and connections. Evaluating activity includes close cooperation with developers in team. In this activity team evaluates whether there are some requirements from software perspective which might effect on hardware. During the development process there have to be continuous discussion with the developers and operative specialists whether there is a need for hardware updates or not. During the operative phase there might occur issues which might be related on software or hardware. These issues are monitored in incident and problem management activity. Service manager together with support specialist will triage possible issues and address those to correct persons in team. It's important that team have common tools to gather issues. In optimal circumstances all tickets are gathered in one tool. In practice this means that team has one tool where they have development and maintenance related tickets in same system. One tool helps also teams to identify if there is some regression related on previous software increments. It also helps team to identify and monitor what kinds of issues are raised up from customer. Issues might have also dependencies to some previous development tasks and during the process support specialists might point out that this certain issue is related on certain development ticket. Baseline to incident and problem management comes from ITIL V3 processes. ITIL V3 processes include also change management processes. In this method change management is handled through the product management and development process. Development iterations include both new features and also possible change requirements management activities. (Farroha & Farroha, 2014; Guo & Wang, 2009; Hüttermann, 2012)

Operating phase includes activities which are handled by service manager and product owner. Every time when new release is available service manager and product owner will arrange system usability surveys. This survey gathers input from end-users where they can give valuable input to the team. This survey includes system usability surveys (SUS) and general evaluation of new features. As a part of operating phase product owner will also arrange focus group meetings where chosen main users are involved. The purpose of this meeting is to gather feedback from customer related on previous release but also gather input to roadmap and product backlog regarding their requirements and needs. This process is important as main users have best insight of their business needs. (Schwaber & Sutherland, 2011; Veneziano, Mahmud, Khatun, & Peng, 2014)

4.4.6 Relations of activities in the process

In the process model there are mentioned several activities. Each of these activities has connections with previous activities and the same time each activity aims to create value and relevant input for next activities in the process. In following table I have listed all activities and what kind of input and outputs each activity contains (FIGURE 23 Inputs and outputs of activities).

Input(s)	Activity	Output(s)

Market trends, Strategy, Vision, User Surveys and Focus Groups	Roadmap	Documented Roadmap with preliminary schedules and high level epics
Roadmap, change management, User Surveys and Focus Groups	Product Backlog	Documented Product Backlog with user stories
Documented Product Backlog with user stories	Product Backlog Refinement	Refined backlog with technical details which fulfills DoR.
User Surveys, Focus Groups, Visual Guide	UI Design & Research	Mockups/wireframes to complement backlog items
Refined backlog with UI mockups/wireframes	Sprint Planning	Sprint backlog
Roadmap, Product Backlog	Architecture update	Updated architecture
Sprint backlog	Tasks for development	Allocated tasks (tickets) to team members
Sprint backlog, allocated tasks and user stories	Implementation with TDD	Releasable functionalities and test cases for automatized testing
Version controlling system, agreed branching rules, programming guidelines	Version Controlling	Releasable functionalities in correct source code branches
Automatized code review system, manual code review (if used pair coding), implemented backlog items	Code Review	Reviewed code, source code ready for testing
Committed code and automatized CI scripts	Building with CI	Compiled code
Implemented, tested, builded and compiled code	Automated Releases (staging)	Newest software version available in all decided environments
User stories with test cases and testing documentation	Functional Testing	Tested functionality and updated testing documentation
User stories with usability guidelines and visual guide	Usability Testing	Usability tested user stories
Coded test cases in same time with development	Automated Testing	Automatized tests and test reports

Security requirements and security test cases. OWASP top 10, CWE/SANS TOP 25 Most Dangerous Software Errors	Security Testing	Security/thread report and possible improvement requirements
Sprint backlog and reports	Sprint Review	Communicated features and changes to the system
Feedback from previous sprint	Sprint Retrospective	Action plan for improvements
Software increment, release plan	Production Deployment	New software version to production environment
Customer feedback	Customer Support	Support and possible requests for the team
Monitoring tools (OS, Hardware, network, automatized reporting)	Monitoring	Automatized reports from environments
Environment reports and Roadmap	Evaluating	Hardware update plan
System monitoring, automatic reporting, user feedback, focus groups	Incident and problem mgmt.	Triaged incident and problem tickets
New available software version in correct environments	User Surveys	SUS report
New available software version in correct environments	Focus Groups	Changes to vision, roadmap items and user stories

FIGURE 23 Inputs and outputs of activities

4.4.7 Tools

Teams have different kind of tool use during the process (FIGURE 24 Tools). On below it is listed possible tools which are sorted for different categories. This chapter won't go through all of these tools but it is good to have a list where to start and tailor toolset to fit for team´s and business´s purposes. Product Owner can maintain product roadmap example with basic MS tools but there are also tailored tools for road mapping (e.g. Aha!). When developing new software it is good to have work documented in same place and for that purpose there are some good ticketing tools available (e.g. JIRA). These kind of ticketing tools are very effective to handle all things in one place and in one system. Example with JIRA team can gather all work to tickets in same system no matter was the task related on development or to maintenance or support. These tools have also extensive plugin storage where each team can tailor suitable set for them. Many

tools have also plugins which are suitable for certain methods and these plugins will help team to example work with scrum processes with certain tool. Tools usually include editors where team can edit workflows within the tool which makes those flexible also for possible process changes. Many of ticketing tools have also different schemas for ticket information structure. Example if team follows in maintenance ITIL way of working they can attach ITIL processes and ticket structure to the system. When team is designing new layout or developing mockups (e.g. Sketch 3) or wireframes (e.g. Pencil) to software they have own tools for that. With these tools they can "draw" easily wireframe which shows on high level where each buttons and each functions are located in layout. With this kind of tools team can also describe the navigation structure better that everyone in the team understands how the coming software or feature works. If team decides to design layouts with wireframes it's good to have visual guide which tells things like typography, colors and themes and that kind of things as wireframes doesn't usually cover those. It is also possible to create more accurate mockups where previously mentioned things are covered but this is something what each team to think and decide by themselves.

Earlier in the research is covered what DevOps will change on high level. It will bring individuals to work together and it will do the culture shift for the daily work. In addition it will bring also lot of tools which can help teams in their daily work it will bring the effectiveness of working. The most important tools on this area are continuous integration (e.g. Jenkins) and configuration management tools. Continuous integration tools will help team to automate and deploy software easier than before. These tools help also to share information and keep it on one place. Configuration management (e.g. Chef) tools help team to automate and orchestra infrastructure and components inside of it. It is one of the main matters when speaking about infrastructure management in this method. There are also many other tools which will help team standardize builds and monitor systems working (e.g. Nagios).

Type:	**Example tools:**
Communication tools	Slack, Skype, MSOC, IRC
Roadmap tools	Aha!
Product Backlog tools	JIRA + JIRA plugins (Agile)
Ticketing tools	JIRA, Redmine, ITSM
UI wireframe & mockup tools	Balsamiq, Pencil, FluidUI, Pidoco, Sketch 3, Keynote, Invision
Operating systems	Linux (variants), Windows
Infrastructure as a Service	Amazon, Azure
Virtualization platforms	VMware, VirtualBox, SaltStack
Configuration management	Puppet, RANCID, Chef
Containerization tools	Docker
Application servers	JBoss, Tomcat, Jetty
Monitoring	New Relic, Nagios, Logstash

Databases	MySQL, PostgreSQL, MongoDB
Queues and caches	Memcache, RabbitMQ
Logging	PaperTrail, Loggly
Version control	Git, SVN
Continuous integration	Jenkins, Bamboo, Hudson
Test and build systems	Maven
Test automation	JUnit, Selenium, SoapUI
Security	Snorpy Threat Stack, Tripwire
Security testing	Owasp Zap, Burb intruder, BDD-security
Functional Testing	Selenium, WebDriver
System tools	Atlas, ScriptRock GuardRail

FIGURE 24 Tools

5 INTERVIEW RESULTS

5.1 Chosen group of people in the research

All interviewed people were IT-professionals with two to ten year experience (FIGURE 27 Work experience (in years)) on IT-sector and age between 26 to 34 (FIGURE 25 Age distribution). These persons were working in following roles: Scrum Master, Technical Architect and Software Developer (FIGURE 26 Roles of interviewed people). They all had previous experience about Agile, UCD and DevOps methodologies. Scrum was the most well-known methodology for all interviewed persons. During the research there was eight participants who were part of the interview process. In each interview there were two persons and both of them had possibility to answer for all questions. At the beginning of interviews researcher introduced the main things of interview and chosen methods (DevOps, Scrum and UCD) which were already familiar already for most of the people. After introducing researched methods persons answered to open questions regarding the methods' values, objectives, principles, rules, roles and tools. In open questions -part researcher introduced the new method. During the method introduction all interviewed persons had a possibility to comment and evaluate the method. When evaluating the method´s different parts we also discussed about previous open answers.

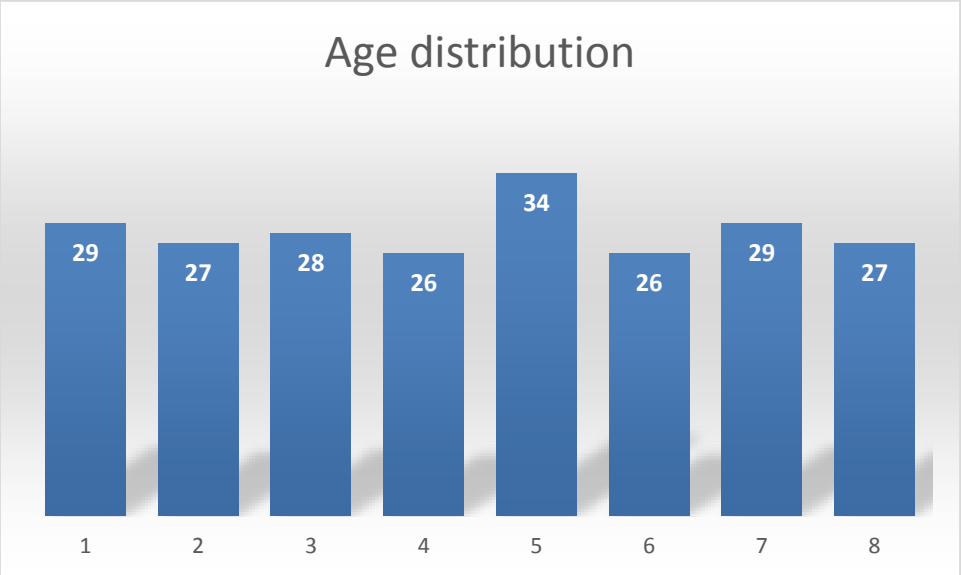


FIGURE 25 Age distribution

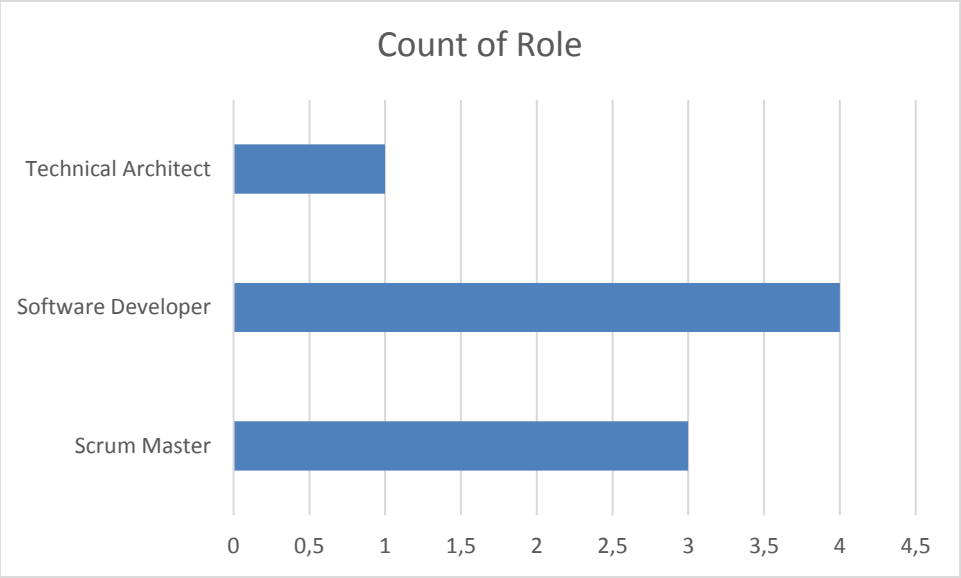


FIGURE 26 Roles of interviewed people

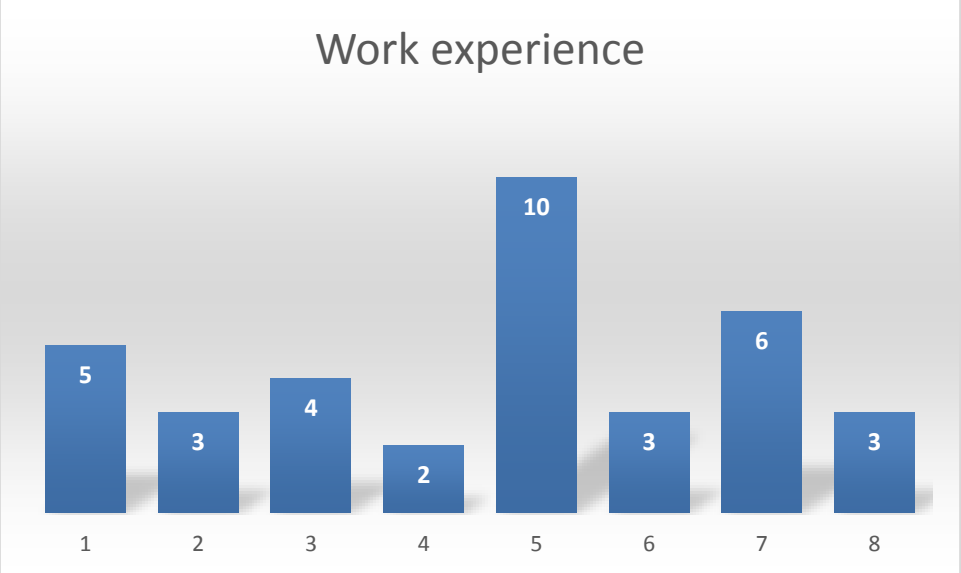


FIGURE 27 Work experience (in years)

5.2 Interview results

Following chapters will include more details about the interviews. At the beginning there is section which covers open questions and on later phase there are details about artifacts evaluation.

5.2.1 Values

Communication and collaboration were the most highlighted values in the interviews. Interviewed persons emphasized meaning of internal and external collaboration. The aim of external collaboration was to understand more deeply customer needs and requirements and also to build up relationship with customer. With internal communication interviewed people were aiming to openness and transparency. These persons also highlighted that within the team there must be an eminent confidence between team members as team's results are the sum of each individual work. They emphasized that all team members have to commit together agreed content of work. Some of the people also mentioned about being proud of own work and they spoke about the great team work which is required for creating good results. About being proud was attached to independency of decisions and having responsibility. Almost every one of interviewed people also mentioned that quality is one of the most important values which should be on one of the fundamentals when doing development work. When discussing about quality interviewed persons commented that it's the ultimate value but it is a sum of many well done actions during the software development. (FIGURE 28 Interview comments regarding values)

Date	Comments from interview
2.6.2016	"Well of course one thing which comes on my mind is bringing development so close to customer that possible."
	"Customer close to development and visibility between different stakeholders."
9.5.2016	"Everyone need to have common target and common ways to work."
3.5.2016	"Trustworthy. When you promise something for the team it must get done. It is impossible to do team work if someone says that it will be tomorrow ready and eventually same situation remains after day and day."
19.5.2016	"Quality is important"

FIGURE 28 Interview comments regarding values

5.2.2 Objectives

Interviewed persons divided goals or objectives for those which are related on internal work and the customer. When speaking about objectives interviewed persons bring forth about value creation to the customer and they also spoke about speed of value creation with iterative methods. Creating value to the customer born from high quality software and easily adaptable user interface. For customer perspective they emphasized that using these methods also increases visibility and it's easier to customer do changes during the software development. (FIGURE 29 Interview comments regarding objectives)

While speaking about internal objectives interviewed persons mentioned that during the software development it is crucial to aim for creating transparency for whole team´s work. Creating transparency arises from smaller software increments, removing organizational silos, common ways of working and clear responsibilities. They also highlighted that product management perspective is important. While doing the software it's important that method aims to focusing prioritized tasks and entities. They mentioned also that this kind of methods should focus on whole product lifecycle. It is important that team concentrates on maintainability already from the beginning. In practice it means that written code must be easy to understand and commented well enough as it helps to maintain the software in future. (FIGURE 29 Interview comments regarding objectives)

Interviewed persons also wanted that method should have individual goals or objectives. They emphasized that bringing operative, UI/UX specialists and development persons in the seam team one aim is also spreading competencies within the team. Interviewed persons emphasized also that the method should aim to self-managing persons who are striving for common target. (FIGURE 29 Interview comments regarding objectives)

Date	Comments from interview
2.6.2016	"Creating value for the customer is most important objective."
	"Doing actions which provides more value to customer in the future."
	"Being closer to customer helps reaction more rapidly to changes and it brings more visibility. Release cycles are bringing visibility to customer in smaller increments. This way we can provide more value to customer in a faster way."
9.5.2016	"Employee satisfaction increases when everyone can show their talent."
	"Expanding own competencies. Example with user interface."
3.5.2016	"I believe that development need to be faster, agile and cheaper. Teams should be self-driven and they should have clear responsibilities"
	"Tools need to support processes and daily work."
	"Main objective is to provide end results what customer is looking for."
19.5.2016	"Processes need to developed iteratively"
	"Quality need to remain high, we need to provide high quality products and customer need to be happy."
	"Of course we cannot just provide functionalities, we need to provide software which is nice to use and it also looks good."

FIGURE 29 Interview comments regarding objectives

5.2.3 Principles

Interviewed persons accentuated that principles are something what team creates together and there might be various bunch of good principles. They emphasized that whatever the principles are in the team whole team must follow those together agreed principles. Focusing on meaningful task/entities and reducing waste was mentioned multiple times. In practice it means that team wants to focus only on important things and they want get rid of everything else. During the software development they wanted to see clear definitions and clear documentation which is output of well-defined DoR. They highlighted that all development should start lightly and lightly doesn't mean low quality and forgetting rules and principles. It means that it is not wise to plan something too long-term and then develop product as things changes during the time. They told that it is easier to create products example with prototyping as then process creates environment for collaboration and discussion which bring better results at the end of the day. Principles and rules were seen quite similar things and therefore next chapter includes complementary matters for principles. (FIGURE 30 Interview comments regarding principles)

Date	Comments from interview
2.6.2016	"Avoid waste."
	"Don't concentrate on things that you can automate."
	"Improving all the time and continuously trying to find things to improve"
9.5.2016	"Clear responsibilities and allocations"
	"Common principles for all"
3.5.2016	"Commitment for rules and this means everyone. We need to hold on together agreed things.
	"Bringing problems for the team. It is important to discuss together if there is some problems. This way we can get best results and we don't need afterwards fix those."
19.5.2016	"Team must plan and prototype the UI at the first phase more thoroughly."
	"Lean thinking on this is quite important. We don't have afford to do anything unnecessary. We need to focus on important matters."

FIGURE 30 Interview comments regarding principles

5.2.4 Rules

When speaking about rules one word comes up many times and it is the team. Interviewed persons told that no matter what kind of rules you have you need to follow those as a team. All members need to commit those rules and they need

to help each other's to achieve common targets. They highlighted that all information must be kept open and visible for everyone and the same time everyone need to know who is doing, what is doing and when is doing. Bringing up possible problems was also emphasized as those obstacles might harm whole team's work. There was listed also common rules which should be followed during the development like code conventions, how to handle tasks in ticketing systems, how to write documentation, how to follow share information to stakeholders etc. (FIGURE 31 Interview comments regarding rules)

Date	Comments from interview
2.6.2016	"If we are speaking about team then we need to commit on together agreed work or tasks. If someone has challenges then someone will help."
	"Decisions need to be made by those persons who understand the problem best. No unnecessary steps."
3.5.2016	"When coding we need to conventions, naming process at general when we have different branches. Also general together agreed rules example with meetings."
19.5.2016	"Planning sprints and freezing those."
	"Start with minimum content and don't try to get everything in one time."
	"Common rules starting by coding conventions and how we work and how we use different tools. As we have good guides and rules for these daily working is quite smoothly."

FIGURE 31 Interview comments regarding rules

5.2.5 Roles

Interviewed persons identified that on this kind of method there must be client facing person who handles all communication between the team and customer. They bring forth that there must be own UI/UX specialist in the team as competences between software developer and UI/UX specialist differs each other so much. Persons highlighted that in the team there must be business owner and someone who owns the products. They told that someone in the team must have the vision of product and it doesn't mean that vision is something what the product looks it is more thoroughly vision which includes business intelligence and market knowledge. Interviewed persons brought up that operative person should become more near to development team as it helps team work together and share information more efficiently. (FIGURE 32 Interview comments regarding roles)

Date	Comments from interview
2.6.2016	"Roles shouldn't be so strict and defined as everyone in the team should understand at least basic things in all relevant areas."

	"All relevant competencies should be available in the team."
9.5.2016	"No matter which role you have everyone need to have same objectives"
	"Someone need to take care of things on a high level and business".
	"We need management roles for business and projects."
	"We need to have some visionary and someone who is taking care of that pixels are in right place."
	"Some of the roles need to focus on usability not only for layout."
3.5.2016	"There are still gap between OPS and DEV and it needs more communication."
19.5.2016	"Of course traditional roles like Scrum Master, Product Owner and those. I am not sure whether OPS members be part of the team but in any case they should be near."

FIGURE 32 Interview comments regarding roles

5.2.6 Tools

Interviewed persons brought up that fundamental tools in this kind of work are working development tools like laptops and robust network. When going deeper to team's processes they highlighted the need of collaboration tools which includes text-based communication and software should also include video conference and screen sharing features (e.g Skype or Slack). Interviewed people mentioned that team should have ticketing tools to track and monitor developing progress (e.g. JIRA). When building a new products and new features for the current products, there should be wire-framing and prototyping tools to plan user interface (e.g. Adobe XD). During the development team needs tools for coding, version controlling and continuous integration tool to build and release software (e.g. GIT and Jenkins). They also mentioned that tools for virtualization and building containers are needed. When doing testing in development phase they see that team should have tools for automated testing. For testing purposes persons listed tools like Mocha and JUnit. (FIGURE 33 Interview comments regarding tools)

Date	Comments from interview
2.6.2016	"CI tools."
	"Project tools."
	"Tools for containerization and virtualization. It makes life easier example when ramping up development environments."
	"Project management, different wireframes, testing automation, virtualization, continuous integration and version controlling."
	"Every interface should be tested and monitored."
9.5.2016	"Tools for managing Scrum, ticketing tool."

	"Automated releases and testing with CI tools."
3.5.2016	"Well the basic tools like version controlling. For continuous development we need CI-tools and it doesn't matter which CI it is. Tools for managing backlog."
	"Some prototyping tools."
19.5.2016	"Place for documentation, guides etc. example wiki or something like that."
	"Working network, tools and laptops."

FIGURE 33 Interview comments regarding tools

5.3 How interviewed people evaluated the method

In the first phase of interviewed persons didn't know anything about created method, they answered for open questions and answer for those questions are analyzed on previous chapters. The structure on the interview gave people possibility to give answers for open questions in defined framework (Fundamental methods (DevOps, Scrum and UCD) were opened). On the last part of the interview persons evaluated the new method.

5.3.1 Values

When interviewed persons got possibility to review the method they mostly brought up same values which was already in the method. They see that collaboration is the most important value together with commitment to work and daily tasks. As a part of collaboration they see that feedback from all stakeholders is important source to continuous improvement. When asking from interviewed persons is there anything missing from the method they emphasized that all values are good and most of the things they have thought regarding this topic are way or another included in those values which are already in the method. No matter most of the values were same than interviewed persons bring forth there was still missing things like self-actualization and rising up work satisfaction. Interviewed persons think that one of the values could be related on passion to work and creation of great software products. When people in the team have passion to their work, very likely they provide better results. Good results will lead to better feedback and better feedback creates flow state to work even harder to perceive better results. Previous comments are related mainly to individual not so much to the method but it was interesting to find out and see how this kind of individual things might effect on whole team. (FIGURE 34 Method evaluation comments regarding values)

Date	Comments from interview

2.6.2016	"Pretty well we had same things highlighted like example collaboration."
	"All values seems really valid."
9.5.2016	"Values seems valid and I wouldn't take anything away."
3.5.2016	"Lot of same things what we said like 'one-team' and communication."
	"Feedback is something what should come from every direction."
	"Most of the comments what we said are pretty near if not exactly same."
19.5.2016	"I think we could find more values but like already said collaboration is still the most important one. We also need to deliver right things and quality, delivery must work."
	"All listed values are good."

FIGURE 34 Method evaluation comments regarding values

5.3.2 Objectives

If interviewed persons agreed method's values same situation was with goals or objectives. They think that listed goals in the method were good but they also find couple things to discuss during the interview. Interviewed persons doubt the balance between faster time and improved quality. They see that you can always improve quality but it affects always on timetable and schedule. During the interviews there was discussion about satisfactory quality which means that during the software develop team aims to together agreed quality level which contains usability, user experience, code, functional and non-functional perspectives. Interviewed persons mentioned about lower risks and one of the persons emphasized that all changes to software are all always a risk but without changes it is impossible work with the software business. No matter of this kind of opinion interviewed persons thought that with relevant processes and ways of working team can mitigate and remove most of the risks. Interviewed persons also highlighted that when team has robust software development processes it will decrease bugs from the software and same time clear processes will motivate people as there is no unnecessary tasks to do. During the interviews was spoken also about launch time heroic. In practice this means that when team has clear processes and goals it's easier to launch new products and releases on time and team can avoid working around the clock couple days before the version releases. Based on this improved predictability could be one of the goals. (FIGURE 35 Method evaluation comments regarding objectives)

Date	Comments from interview
2.6.2016	"Hard to find something to add."

	"When we have focused on collaboration we have notified that also bugs have decreased. One of the things was also the satisfaction which increased when everyone know better where the project was going. These kind of methods will improve own behavior and it these also motivates people."
	"I see that one objective is decreasing risks. Smaller releaseses will decrease 'launch time heroic'."
9.5.2016	"Quality is difficult as you can polish applications as long as you want. Biggest question is where goes the line for the 'good enough'."
	"Team need to find balance between delivery speed and quality. Therefore 'improved quality' and 'faster time' might have a conflict."
3.5.2016	"Most of the objectives had same what we listed. Hard to find more."
	"If I would like to add something maybe it would be related on planning and importance of it."
19.5.2016	"All objectives are good."

FIGURE 35 Method evaluation comments regarding objectives

5.3.3 Principles

Interviewed people thought that listed items in the method are good but they highlighted couple important matters during the interviews. First one was value-based prioritization. Interviewed persons emphasized that in software development this kind of prioritization should be clearly defined. As long as it is just value-based prioritization without any defined formula it is just product owner's opinion of the most valuable feature or epic. Therefore all software methods were prioritization is done should include transparent prioritization rules. In the method there is mentioned UI consistency but interviewed persons also brought up that code consistency is equally important. When building several products with consistent way of coding the maintainability remains better. No silos –principle shouldn't effect on responsibilities as in the team those should be defined by team members. Interviewed persons highlighted that no matter team members are closer together it doesn't mean that everyone is doing everything. Team members have their own competences and responsibilities and those should remain no matter organizational silos are removed. As a part of iterative development and process control interviewed persons thought that product backlog refinement is in the crucial role. They emphasized that iterative development doesn't mean developing without planning and therefore they highlighted the meaning of the product backlog refinement. (FIGURE 36 Method evaluation comments regarding principles)

Date	Comments from interview

2.6.2016	"Value based prioritization should be defined well."
3.5.2016	"Need to refine early enough and enough at general. At behind there can be principle that you have to plan enough before you start doing anything."

FIGURE 36 Method evaluation comments regarding principles

5.3.4 Rules

Every one of interviewed people thought that listed items in the method were good but they manage to find some points which could be more specified. At generally this kind list of rules can be way more extensive depending on the team. We discussed about DoR and how it should also include non-functional requirements which they thought to be forgotten quite often when doing software development. Interviewed people thought that there should be rules which are related on measuring. They wanted to understand how it would be possible to measure development at individual level example measuring feature complexity and realized time of development. They also bring forth that measuring might be quite complex depending on software architecture. One of the most important finding was architecture which was seen often too stabile. Interviewed people emphasized that architecture should be developed continuously during the software development. Importance of architecture was more important when created system includes several integrations from system to another. Used libraries and component chooses should be refined in regular timeframes. Persons thought that many times when creating software it will remain on market several years' or even decades. During the product lifecycle outdated architecture, libraries and components will cause technical debt and therefore those should be updated regularly. One thing which came up was rules regarding testing responsibilities. Interviewed people highlighted that when testing has partial responsibilities between testers and developers rules and responsibilities should be defined exactly. Interviewed bring forth also rules about removing obstacles on personal level not only a team level. They saw that in the agile rules often team is removing obstacles on a team level but not necessary on individual level and eventually effort of team is a sum of individuals. (FIGURE 37 Method evaluation comments regarding rules)

Date	Comments from interview
2.6.2016	"It is important rule that we must like changes, but we cannot do those during the sprints. If we would do changes during the sprints it would destroy our predictability."
	"That 'understanding architecture' is good but it is sometimes really hard when development is done in small pieces. On SAFE there is architectural runaway where this kind of things should considered."

9.5.2016	"These are good but we should also monitor somehow individuals. Example how individuals have evolved during time. Monitoring only team can lead on situation where some members will just pick easy tickets. So at least we cannot measure it with amount of tickets."
	"In addition for UI consistency there should be code consistency. It will make life easier in a long run. There could be mentioned also simplified code. It is not always best solution to write shortest or the most effective code."
	"Framework selection has quite big meaning when we are watching products after many years. Example if you haven't updated the code base for a while and then you start doing it again."
3.5.2016	"These are good. Maybe on this should also highlight that it is everyone's responsibility to bring problems forth."
	"Team should remove problems from the team also not only from software."
19.5.2016	"Good list. I love that 'live by values' –rule."
	"Hard to find anything to add."

FIGURE 37 Method evaluation comments regarding rules

5.3.5 Processes

When discussed about processes people highlighted that iterative way of running process wasn't clear enough in method's chart. They thought that when doing automatized or robot testing TDD is one of the key things. When doing automatized or robot testing it causes more work during the development phase but it saves time from testers and especially when time goes by and product evolves and complexity increases it will be beneficial. One of the interviewed person highlighted that there could be also convention testing mentioned in the process. Also when discussed about processes interviewed people bring forth that architecture renewing should be part of the process. At least it should be evaluated regularly. When discussing about architecture they highlighted that there should be responsible person to thing architecture for the future. Documentation was emphasized also several times during the interviews. No matter agile development doesn't emphasize documentation people still thought that it is important especially when people in the team are changing and product lifecycles are long-lasting. One interesting point which came up was UI/UX validation. They thought that UI/UX designer should validate layout, usability and user experience before new release is launched. Interviewed people highlighted that testing is not planned well enough in this process. Testing is part of the process but it should be planned in some point of the process. Prototyping was also one of the topics which came up. Now the method doesn't include prototyping which

is often done during especially when ramping up new products. (FIGURE 38 Method evaluation comments regarding processes)

Date	Comments from interview
2.6.2016	"Upper part of chart is confusing as it looks like 'waterfall'."
	"Teams should have clearly defined environments for different purposes."
	"Maybe there should be pair-programming mentioned."
9.5.2016	"Convention testing is missing, otherwise looks good."
	"Architecture update regularly is important and it is good to see that it is here."
3.5.2016	"Documentation should be highlighted in the process."
	"In development phase there could be also some sort of small pilots or prototypes."
19.5.2016	"Iterativity should be shown more clearly."
	"Testing planning should be covered in some part of the process."

FIGURE 38 Method evaluation comments regarding processes

5.3.6 Roles

Every one of interviewed people though that roles in the method were good. They emphasized that individuals might have multiple roles in the team. Interviewed people also bring forth that rules and responsibilities between testers and developers should be specified in software development processes more thoroughly. Nowadays when automated testing or robotized testing is increasing developers thought that their role is to create relevant unit tests, test scripts to robot and automatized testing. Whereas end-to-end testing or acceptance testing against acceptance criteria should have own tester role in the team. (FIGURE 39 Method evaluation comments regarding roles)

Date	Comments from interview
2.6.2016	"This is good but I see that testers role should be separated. Example when you have to test bigger system and match it with requirements. Of course unit testing goes for development but when speaking about system and integration testing these should be done by testers."
9.5.2016	"Tester role is quite hard if it goes on deep technical level."
	"Someone should validate that UI match also with requirements and usability is on wanted level."
3.5.2016	"It is hard to find testes who really understand the code. Someone has always to do code review. Code review should be part

	of the process always and it is developer responsibility typically."
	"I see that developer role has become more near to tester role nowadays. Automation is used more and more and therefore coding skills are needed while doing the testing. In practice it means that developers are writing the test scripts while coding."
19.5.2016	"It would be great if developers would have more UI skills. This way team could utilize their members better."
	"Developer need to have tester skills."
	"Depend on testing type which role there should be. If you are doing testing like acceptance testing or any wider testing then you should have own role for that."

FIGURE 39 Method evaluation comments regarding roles

5.4 Most important observations from the interviews

During the interviews it came clear that DevOps as a term was more or less unclear for several interviewed persons. Most of the interviewed people knew something about it but it wasn't clear for all. No matter people didn't know details about the DevOps they still spoke a lot of those things which are actually crucial when going towards DevOps. Almost everyone bring up the importance of communication and collaboration between all relevant stakeholders in the process, which is actually the core of DevOps. DevOps wants to break to silos in the organization and it wants to bring specialists work together. Without knowing the details of DevOps interviewed persons bring forth also criticism about DevOps. In the interview it was discussed about different aspects regarding objectives between development and operative teams. One of the most important thing that was realized is essential conflict between these two (Dev & Ops) groups; development team wants to change all the time something in the software when at the same time operative team wants to keep system as stable as possible and minimize changes. When these groups might have this kind of fundamental conflicts in their objectives it might cause problems. It is clear that the most important thing in the team is creating common goals for a whole team which includes operative and development people. At the same time whole team needs to understand that things are changing all the time in environment where they are working. Team needs to embrace the change as mentioned in the Agile Manifesto (2001). When everyone in the team has same objects and they understand each other better they can provide tools, processes and ways of working which are leading the team towards common goal.

Increasing automation level is not always so easy. During the interview it is noticed that there are several things which can be automatized but also several things where automation doesn't necessary increase the efficiency or it is hard to implement. Increasing automation helps especially when; system complexity has increased, system includes several target platforms or devices, generally tasks or

operations which have clear steps and which are recurring regularly. Especially interviewed persons bring forth that it's easy to do automated testing for basic use cases but typically errors or bugs doesn't occur in basic use cases and therefore creating automated testing is difficult to build comprehensively.

Architecture doesn't get enough attention. Often systems have architecture decided at the beginning but it stays as it is once decided and implemented. During the product lifecycles and development processes team should evaluate the architecture more often and if they see risks they should react on it. On scaled agile framework (SAFe) there is a process which pays attention on architecture in every iteration but example on Scrum it is not so clear part of methodology.

Clear responsibilities between testers and developers are missing. When team is changing the way of working example by automated testing or robot testing they are still using same processes and roles what they have decided or took from some software development method. This is causing dubiety about the responsibilities between these two roles. During the research is realized that there should be more accurately defined testing responsibilities between these roles.

Rules and principles list can be almost endless. In this kind of frameworks you can create list of rules or principles but at the end of the day those should evolve and updated by the team. It is good to have some fundamental matters listed but in every team there is own dynamic and nuances which are affecting this kind of lists.

Values were hard to define for most of the people. During the research was found out that when asked about values in this kind of method it was pretty difficult for them. Possible reason behind this is abstraction level at daily work is so much higher and people don't actually think so much this kind of matters in their daily work. If values were generally difficult topic for the interviewed persons it was surprising to see that every one of the interviewed people highlighted the meaning of collaboration. Software development is nowadays so often team work and communication between people that everyone realizes that communication skills and collaboration between each other is essential matter.

5.5 What's new was found out during the interviews?

If looking this research as a whole there didn't come any big surprises. Of course during the research and interviews it was seen that interviewed people had various different angles when they observed the method. Maybe one of the interesting a new thing was to realize that people sees this kind of methods too technical or team-centric. Methods don't take into account personal or individual perspectives. Many of the interviewed persons bring forth that example processes are always thought from the team perspective and there is not necessary nothing from individual aspect. Especially one word came up multiple times when speaking about values and it was passion. Many of the interviewed persons were thinking that without passion on individual level team cannot achieve great results as a team.

Creating a team with various competences (UI/UX, Ops, Dev) is precious not just for the team but also for the individuals. Interviewed people see that their competence is increasing when there are people which are great for some certain skill that they don't personally have. Example developers bring forth that their UI/UX skills are increasing all time when working in the same team together with UI/UX specialists.

Pair working and peer support is something what should be used more. Interviewed persons highlighted that sometimes they are doing difficult and far-reaching decisions without getting enough peer support. If possible software development processes should guide to work as a pair in high risk tasks.

6 UPDATED METHOD

On the interviews there was introduced method which was analyzed and commented by professionals. Based on these comments method was iteratively developed for the next version and in following chapters these changes are covered. In further phase when method is updated professionals have possibility to answer on survey which aims to get understanding of maturity level of the method. Survey results are available on chapter 7.

6.1 Values

After interviews method is refined and updated based on comments from the interviewed people. During the interviews it came clear that there must be individual perspective. Therefore values chart includes totally new main area which is called Individual (FIGURE 40 Values (after interviews)). Under the Individual main area I gathered important values like “Passion for the work” and “Being proud of own work”. These both are related on individual effort for the team work. Individuals want to be part of something bigger. They want that their work has purpose and they want to help their customers by creating awesome products and services. Interviewed people see that all team members need to be trustworthy. If you cannot rely on your team member it easily tears team down. When all members have commitment to their work and they also deliver what they have promised team archives the best results. Based on the comments collaboration and quality areas were also updated. Visibility to all members and stakeholders was seen so important thing that it was raised on this list. Speed of development came up also in the discussion. This was raised on the list as it is important part of the method if we want to improve time to market.

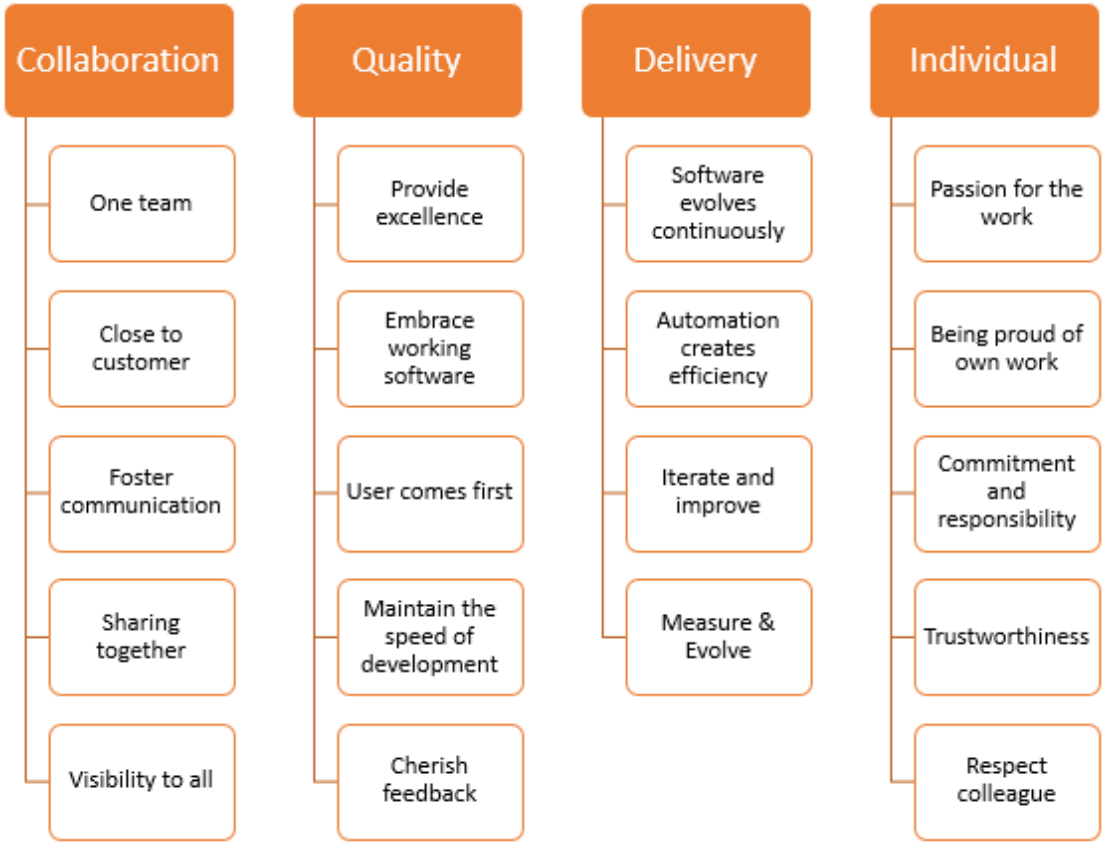


FIGURE 40 Values

6.2 Objectives

After interviews it was clear that there are couple things missing from this method's objectives list. "Value creation" in every aspect is the main objective what this kind of method should have (FIGURE 41 Objectives). All the time and in every task team is creating value. Value can be created to external or internal parties. It can be related on product or it can be related on end-users. Main thing is that the team can always see some stakeholder who gets value from the tasks we are working. "Reducing silos" is also one of the new goals. This is actually one of the main things in the DevOps so it should be on there also earlier. One of the important new goal is also "Common ways to work". When team has common ways to work it decreases the uncertainty. When uncertainty can be decreased it helps team to focus on important things. Interviewed people see that "Satisfying working environment" should be on this list. This has relation on individual main area on values. Team results will be better when their methods are in good shape. When methods supports the team work it will create more satisfying environment to work. As brought up product lifecycle can be even decades so it is important to have goal which focuses on product maintainability.

FIGURE 41 Objectives

6.3 Principles

Principle list is also updated after interviews. As other method artifacts this has now more principles related on team working. One important change was also ideology of acceptable failure which should be available in the method. This means that in the team is approved to try something new and fail with it. With this kind of culture organization is supporting innovative team where members can try something new and it doesn't need necessary always need to be something what team will necessary deploy to any products. Main ideology behind of this is that team will also learn from failures. Most important thing in this kind of tech spiking culture is ability to stop wasting time with things that doesn't work. "Create loved products" is one of the change that is related on method's values (FIGURE 42 Principles). At the same time when team is developing technical solutions they need to have passion to create products which are loved by users. At general all of these method artifacts have got changes which connects those more near to each other.

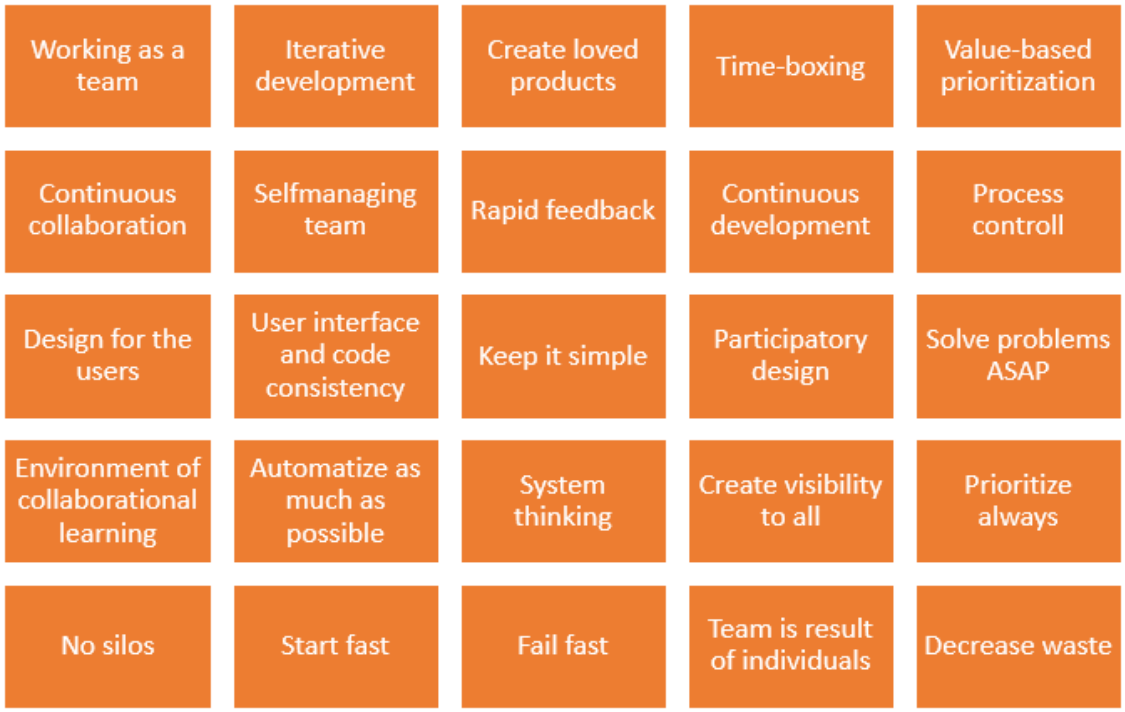


FIGURE 42 Principles

6.4 Rules

During the interview there came up couple rules which were pretty important for the method. One of the most important things was architecture which need to up-to-date all the time. Too often team have one-time planned architecture which doesn't evolve during the product lifecycle. Therefore it is important to update architecture during the development process. With this kind of rule team also can refactor and improve those solutions what they have done earlier. In the rules list there is couple new collaboration related rules added in the list; "No solo-playing" and "Ask if something is unclear" (FIGURE 43 Rules). With these rules objective is to highlight the meaning of team work. Individuals doesn't necessary always see that their work and success might have relation example on schedules and other important matters. Testing was one of the things which was discussed quite a lot. Interviewed people bring forth that testing responsibility is nowadays shared. It is not anymore so clear than it has maybe been previously. Developers will usually do unit-, component-testing and also they update automated testing scripts. In the development process there are typically testers who are focusing on feature-, E2E- and acceptance testing. Team has also responsibility to keep up-to-date tools and technology stack which they are using. This has to be continuous process where "tech stack" is evaluated regularly and if necessary it will be updated to the next version. There might be situations example that one technical component, layer or framework is getting old and team has noticed that is important to start move on new option.

FIGURE 43 Rules

6.5 Process

Process is the part of the method which is giving directions to daily work. Interviewed people highlighted that iterativity should be displayed more clearly in the process chart. In the planning section there are now new process phases: "Prototyping" and "Change Management" (FIGURE 44 Process). Prototyping is part of the development process where team is testing something new. It can be in tech spike where team is testing new techniques or it can be prototyping of new feature. Even in some cases team can evaluate alternative solutions for planned features. Change management can be either part of the maintenance process or it can locate in planning phase. This has dependency to the required speed of updates. If release includes changes which have extremely high priority it is necessary to do changes as soon as possible but in the normal development cycle changes are coming either from service management or product management processes. One new phase in the development section is UI/UX review. This was seen good enhancement to the process as too often UI/UX specialists are involved to the process in the planning section but they don't necessary review and approve their plans at the end of development.

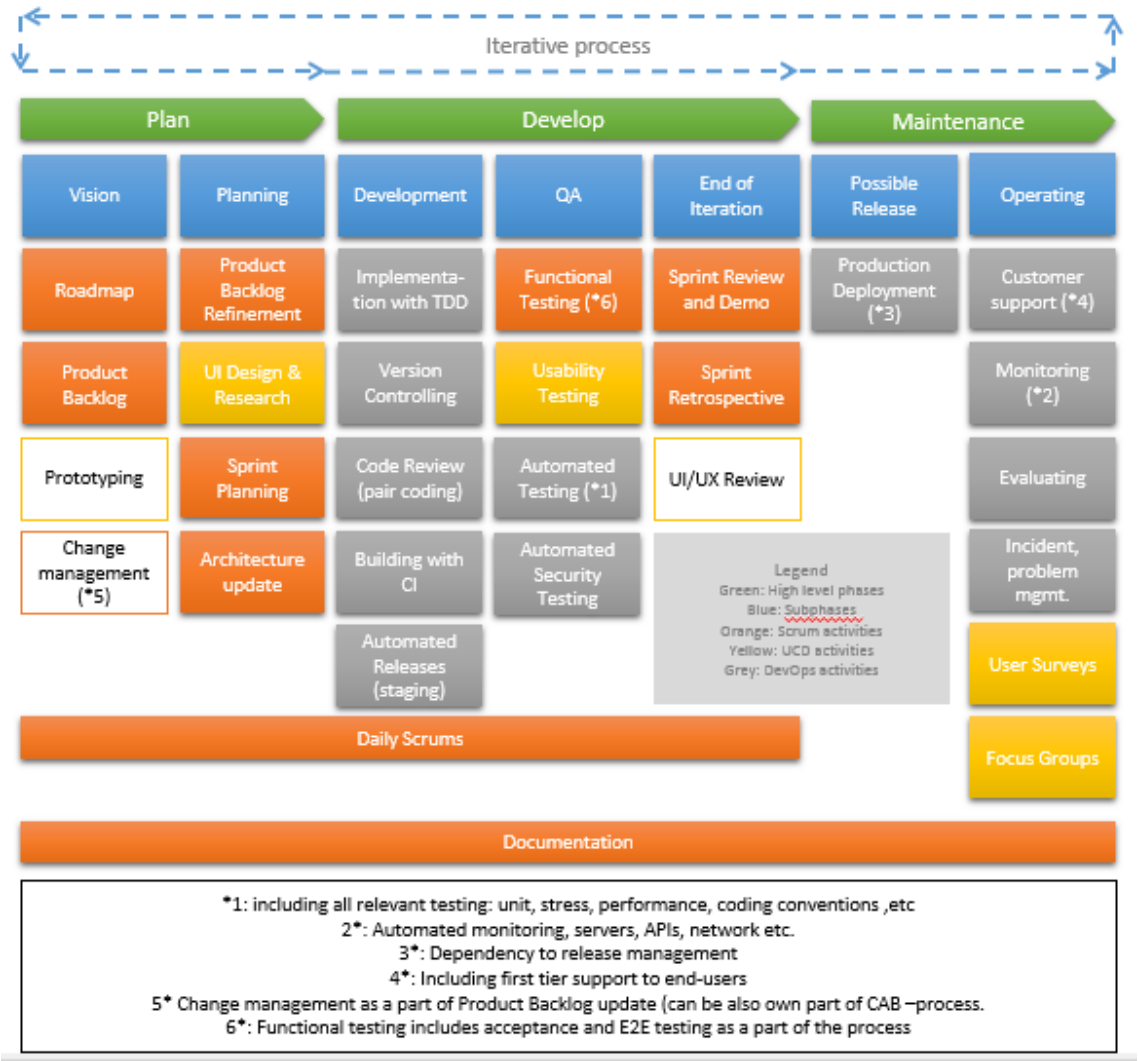


FIGURE 44 Process

6.5.1 Changes to the process´s inputs and outputs

Input(s)	Activity	Output(s)
Market trends, Strategy, Vision, User Surveys and Focus Groups	Roadmap	Documented Roadmap with preliminary schedules and high level epics
Roadmap, change management, User Surveys and Focus Groups	Product Backlog	Documented Product Backlog with user stories
Business needs, tech spike ideas, alternative options for features	Prototyping	Possible better solutions/feature, possible new usable technique/framework/tool
Change requests from continuous services,	Change Management	Change to current product

Change request from business epics		
Documented Product Backlog with user stories	Product Backlog Refinement	Refined backlog with technical details which fulfills DoR.
User Surveys, Focus Groups, Visual Guide	UI Design & Research	Mockups/wireframes to complement backlog items
Refined backlog with UI mockups/wireframes	Sprint Planning	Sprint backlog
Roadmap, Product Backlog	Architecture update	Updated architecture
Sprint backlog	Tasks for development	Allocated tasks (tickets) to team members
Sprint backlog, allocated tasks and user stories	Implementation with TDD	Releasable functionalities and test cases for automatized testing
Version controlling system, agreed branching rules, programming guidelines	Version Controlling	Releasable functionalities in correct source code branches
Automatized code review system, manual code review (if used pair coding), implemented backlog items	Code Review	Reviewed code, source code ready for testing
Committed code and automatized CI scripts	Building with CI	Compiled code
Implemented, tested, builded and compiled code	Automated Releases (staging)	Newest software version available in all decided environments
User stories with test cases and testing documentation	Functional Testing	Tested functionality and updated testing documentation
User stories with usability guidelines and visual guide	Usability Testing	Usability tested user stories
Coded test cases in same time with development	Automated Testing	Automatized tests and test reports
Security requirements and security test cases. OWASP top 10, CWE/SANS TOP 25 Most Dangerous Software Errors	Security Testing	Security/thread report and possible improvement requirements

Sprint backlog and reports	Sprint Review	Communicated features and changes to the system
Feedback from previous sprint	Sprint Retrospective	Action plan for improvements
UI/UX plan	UI/UX review	Acceptance or change requests to the product management
Software increment, release plan	Production Deployment	New software version to production environment
Customer feedback	Customer Support	Support and possible requests for the team
Monitoring tools (OS, Hardware, network, automatized reporting)	Monitoring	Automatized reports from environments
Environment reports and Roadmap	Evaluating	Hardware update plan
System monitoring, automatic reporting, user feedback, focus groups	Incident and problem mgmt.	Triaged incident and problem tickets
New available software version in correct environments	User Surveys	SUS report
New available software version in correct environments	Focus Groups	Changes to vision, roadmap items and user stories

6.6 Analyze of the method after changes

Interviews verified that the blueprint and the artifacts in the method were in good shape but it can still can be refined. After changes to method it is in better shape but same time it is clear that there is possibility to iterate more and more after further feedback rounds. At the same time when adding more details in the method it would increase the method's complexity. One of the research targets was to keep method simple and therefore it is good to have some limits which matters are reasonable and which are not. Mainly it is discussion what is good enough and in this shape method is quite near to settled research objectives. There are still some artifacts or parts inside the method which could get more attention. To pinpoint some of those objectives could be more refined and also the method's process. In the process there are some parts which are not necessary opened enough and those would need some fine tuning and more closely specifying. With this maturity level most of the teams would benefit by using this method. Of course method is not comprehensive when we are going to daily work but it will gave a framework for it. Teams which are already in good level

with their methods can still find some parts of this research which would help them to improve their work.

7 SURVEY RESULTS

After interviews the method was updated for the next version. The purpose of refinement was to improve method based on professional's comments and reviews. After refinement professionals can review how well the method would work in practice with the survey (FIGURE 45 Survey results).

Role of responder	**Abbreviation**							
Mobile Developer	MD							
Software Developer	SD							
Junior DevOps	JDO							
Senior Software Developer	SSD							
Role	**MD**	**SD**	**JDO**	**SD**	**SD**	**SD**	**SD**	**SSD**
Values	**Collaboration**							
One team (collaboration)	5	4	3	4	4	5	3	5
Close to customer (collaboration)	4	4	4	5	3	4	4	5
Foster communication (collaboration)	5	3	3	5	3	5	4	5
Sharing together (collaboration)	4	5	4	5	3	4	3	5
Visibility to all (collaboration)	4	3	3	5	2	4	5	4
Values	**Quality**							
Provide excellence (quality)	4	5	4	5	4	4	5	5
Embrace working software (quality)	5	5	3	5	4	5	5	5
User comes first (quality)	5	4	4	5	5	4	5	4
Maintain the speed of development (quality)	4	4	4	4	4	3	4	3
Cherish feedback (quality)	4	4	4	5	3	4	5	4
Values	**Delivery**							
Software evolves continuously (delivery)	4	4	4	4	4	3	3	3
Automation creates efficiency (delivery)	5	4	4	5	4	4	2	4
Iterate and improve (delivery)	5	4	5	4	4	4	4	5
Measure and evolve (delivery)	4	3	4	4	3	4	4	3
Values	**Individual**							
Passion for the work (individual)	4	4	4	4	4	5	3	5
Being proud of own work (individual)	3	5	4	4	4	4	3	5
Commitment and responsibility (individual)	5	5	4	4	4	5	5	5
Trustworthiness (individual)	4	4	4	4	4	5	5	5
Respect colleague (individual)	5	4	4	4	4	4	5	4
Objectives								
Value creation	5	4	4	5	3	4	5	5
Reducing silos	4	4	3	4	2	3	3	5
Higher productivity with lower costs	4	3	4	4	4	3	3	3
Improved predictability	4	3	4	4	4	5	4	4
Better customer and end-user satisfaction	4	5	5	5	3	5	5	5
Faster time-to-market	3	3	4	5	3	3	4	4
Improved collaboration	4	3	4	4	4	5	4	5
Satisfying working environment	4	4	4	5	4	4	5	5
Lower risks	4	3	4	4	4	3	5	4
Improved quality	5	4	4	4	4	4	5	5
Common ways to work	4	4	3	4	3	3	2	5
More maintainable software	5	5	4	3	5	5	5	4
Principles								
Five most important principles	No silos, Iterative development, Rapid feedback, Fail fast, Team is result of individuals	Working as a team, Design for the users, Keep it simple, Team is result of individuals, Prioritize always	Iterative development, Automatize as much as possible, Continuous development, Value-based prioritization, Prioritize always	Continuous collaboration, Environment of collaborational learning, No silos, Automatize as much as possible, Rapid feedback	Continuous collaboration, Design for the users, User interface and code consistency, Rapid feedback, Solve problems ASAP	Working as a team, Continuous collaboration, Rapid feedback, Continuous development, Prioritize always	Working as a team, Design for the users, Iterative development, User interface and code consistency, Prioritize always	Working as a team, Continuous collaboration, Iterative development, Selfmanaging team, Automatize as much as possible
Overal value of principles (scale 1-5, 5 best)	4	4	4	4	4	5	5	5
Rules								
Five most important rules	Refine in every sprint before planning, Help colleague, Keep backlog prioritized, Follow together agreed rules, Be proactive	Help colleague, Keep architecture up to date, Be willing to learn, Understand the architecture	Refine in every sprint before planning, Help colleague, Keep backlog prioritized, Make sure that you have backlog ready for incoming sprints, Identify points where things can go wrong	Help colleague, Identify waste and try to reduce it, Ask how automation can help, Keep architecture up to date, Understand the architecture	Be truthfull, do not gather technical debt, Be willing to learn, Ask if something is unclear, Know your product, Be proactive	Refine in every sprint before planning, Responsibility to team members, Ask if something is unclear, Slice requirement to small enough, Understand the architecture	Responsibility to team members, Keep information available to all, Code conventions, Ask if something is unclear, Be proactive	Refine in every sprint before planning, Help colleague, Share obstacles, Make sure that you have backlog ready for incoming sprints, Commitment to rules
Overall value of rules (scale 1-5, 5 best)	5	4	4	5	5	4	5	5
Process								
Overall value of process (scale 1-5, 5 best)	4	4	4	4	3	5	5	5
Roles								
Overall value of roles (scale 1-5, 5 best)	5	4	4	4	4	5	5	4
Overall								
Overall value of method (scale 1-5, 5 best)	4	4	4	4	4	5	5	5

FIGURE 45 Survey results

Age distribution in the second iteration was between 20 to 33 years (FIGURE 46 Age distribution). Youngest ones had less than year experience from the software business as the oldest ones had eight year experience. These people who involved to the survey were mainly software developers but some of those had also experience about operative work with servers (Ops experience).

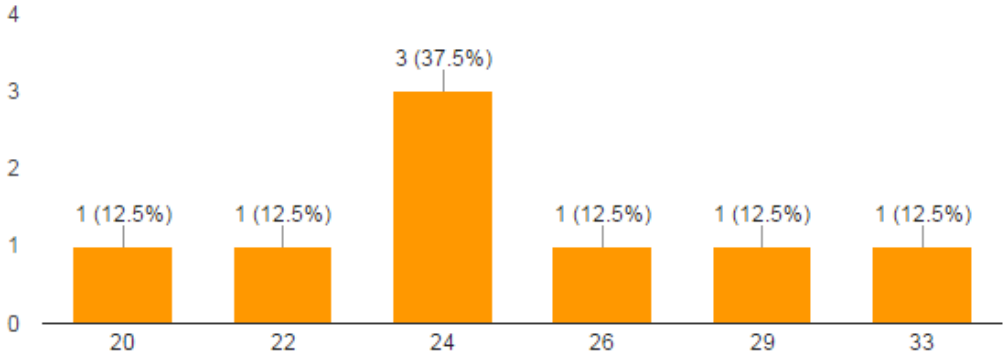


FIGURE 46 Age distribution

In first question purpose was to evaluate which values were most important for the team. Each value had numerical value between one to five and five. In the survey grade five means that value was really meaningful for the team and the grade one was opposite.

In general people who answered to survey are seeing that collaboration section inside the values is important matter. “One team”, “Foster communication” and “Sharing together” values gathered the highest grades regarding the meaningfulness for the team. It was interesting to see that responders doesn’t see that being close to customer is not as relevant as collaboration within the team. This might be related on roles in the team. Developers and people who are doing the actual code doesn’t see the interaction with customer so relevant as example Scrum Masters and Product Owners see it. (FIGURE 47 Survey - Values - One team, FIGURE 48 Survey - Values - Close to customer, FIGURE 49 Survey - Values - Foster communication, FIGURE 50 Survey - Values - Sharing together, FIGURE 51 Survey - Values - Visibility to all)

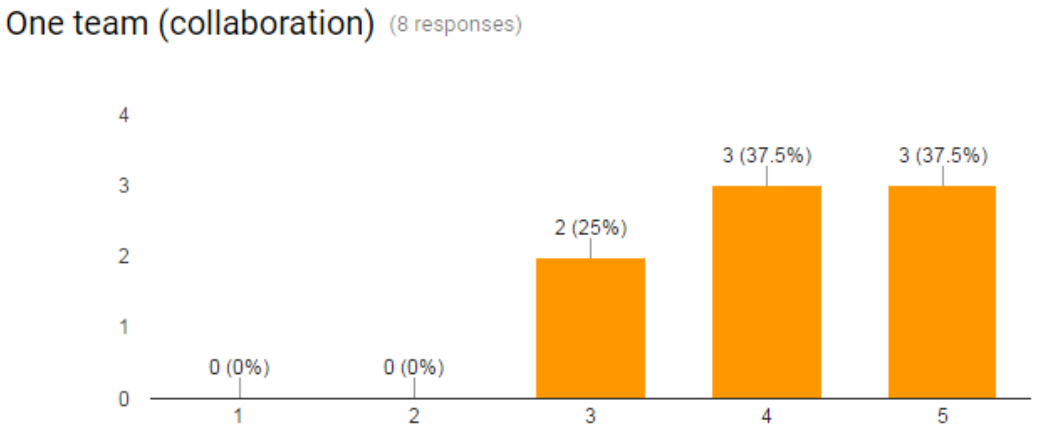


FIGURE 47 Survey - Values - One team

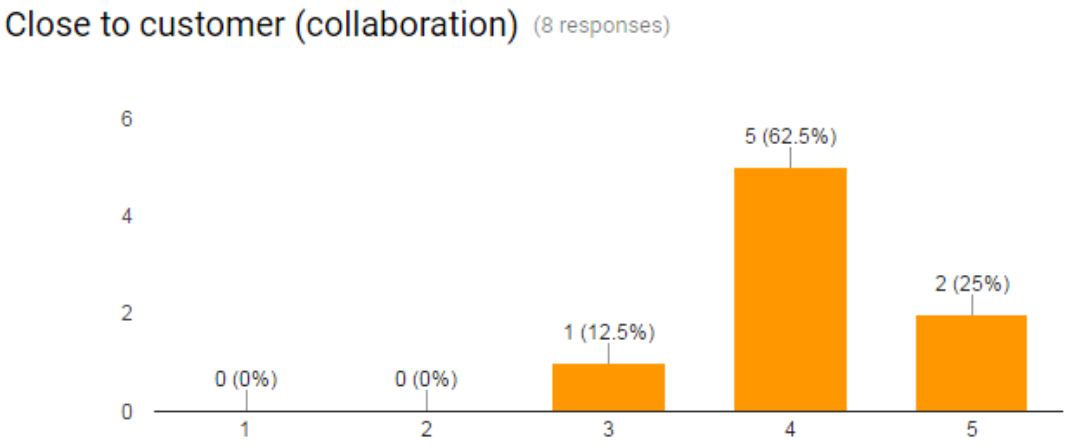


FIGURE 48 Survey - Values - Close to customer

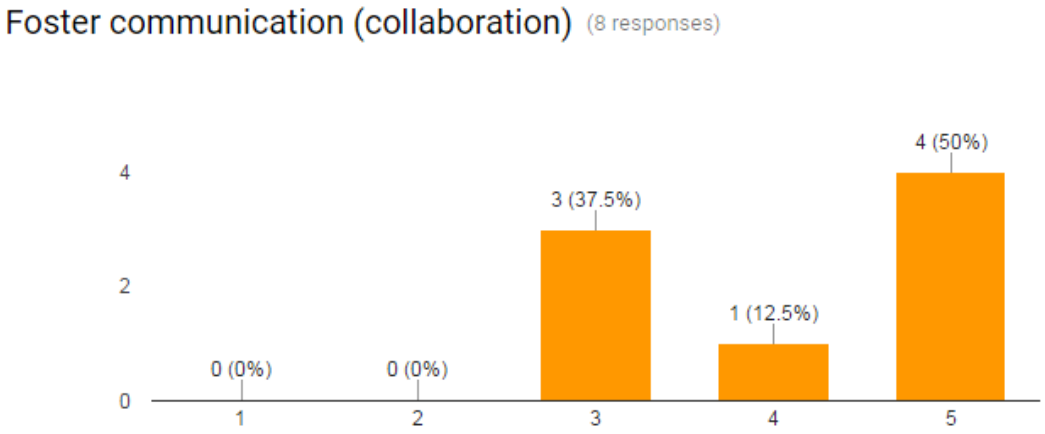


FIGURE 49 Survey - Values - Foster communication

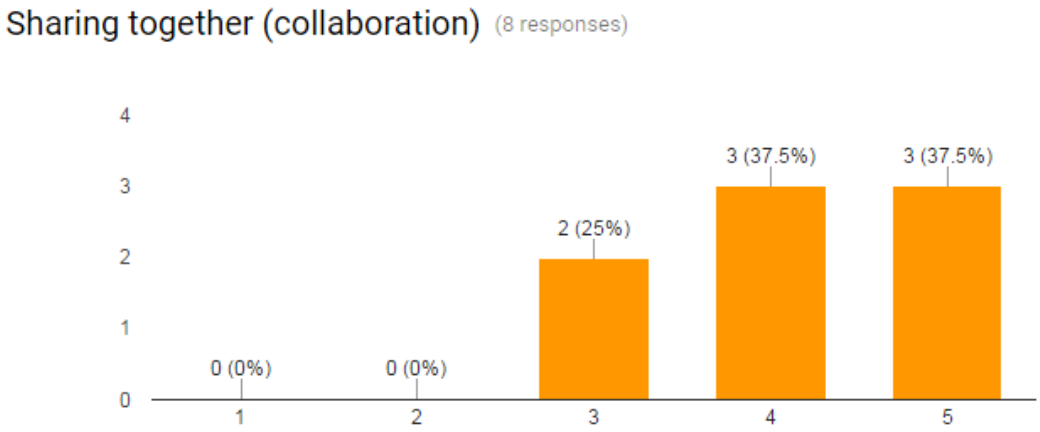


FIGURE 50 Survey - Values - Sharing together

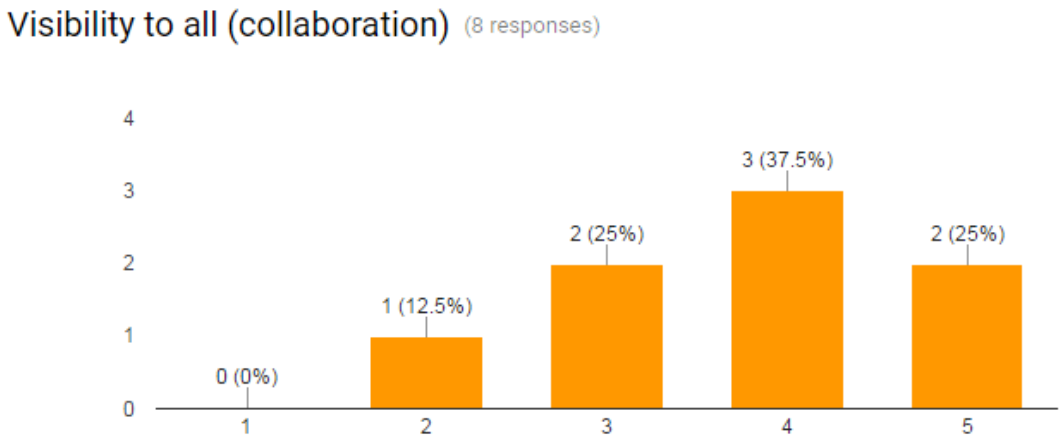


FIGURE 51 Survey - Values - Visibility to all

Second section under the values was quality. Under the quality section there was following values under evaluation: "Provide excellence", "Embrace working software", "User comes first", "Maintain the speed of development" and "Cherish feedback". Most of the responders see that working software is the main thing from the quality perspective. Responders saw this value so fundamental that without working software it is hard to provide quality software services. As previous section in this section maintaining the speed of development is one of those values which are more related to business and especially it is important to Product Owner and Scrum Master roles. On these roles predictability of the team effort is in important role and they need to understand and forecast output of the team in every sprint. If there is volatility in team velocity it will make their work more difficult but it is not necessary seen within the developers or operative specialist roles. (FIGURE 52 Survey - Values - Provide excellence, FIGURE 53 Survey - Values - Embrace working software, FIGURE 54 Survey - Values - User comes first, FIGURE 55 Survey - Values - Maintain the speed of development, FIGURE 56 Survey - Values - Cherish feedback)

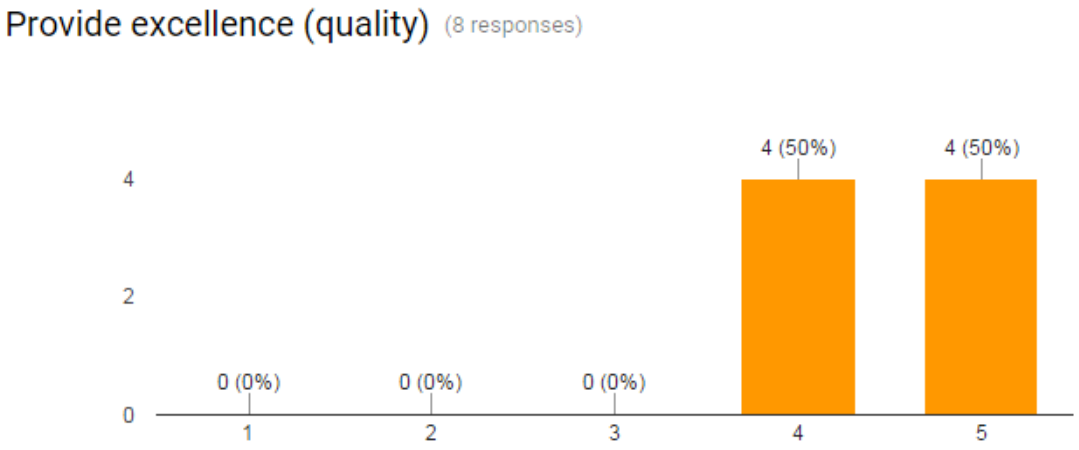


FIGURE 52 Survey - Values - Provide excellence

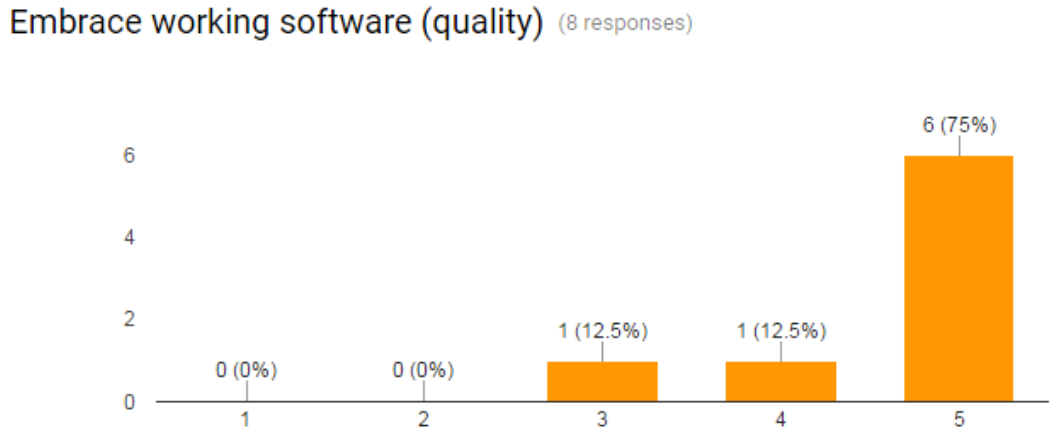


FIGURE 53 Survey - Values - Embrace working software

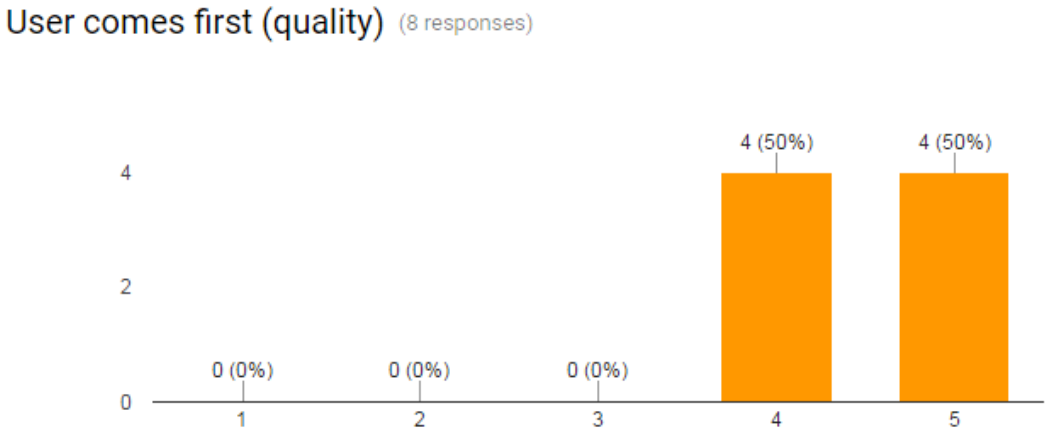


FIGURE 54 Survey - Values - User comes first

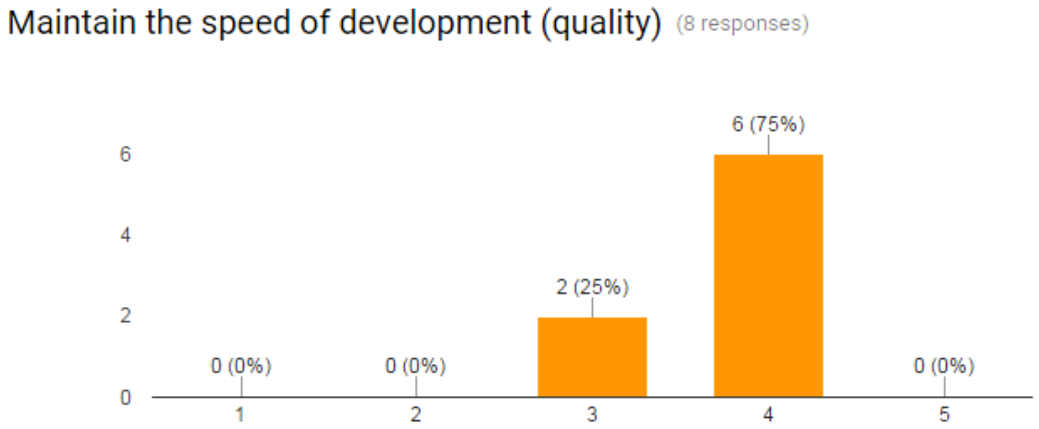


FIGURE 55 Survey - Values - Maintain the speed of development

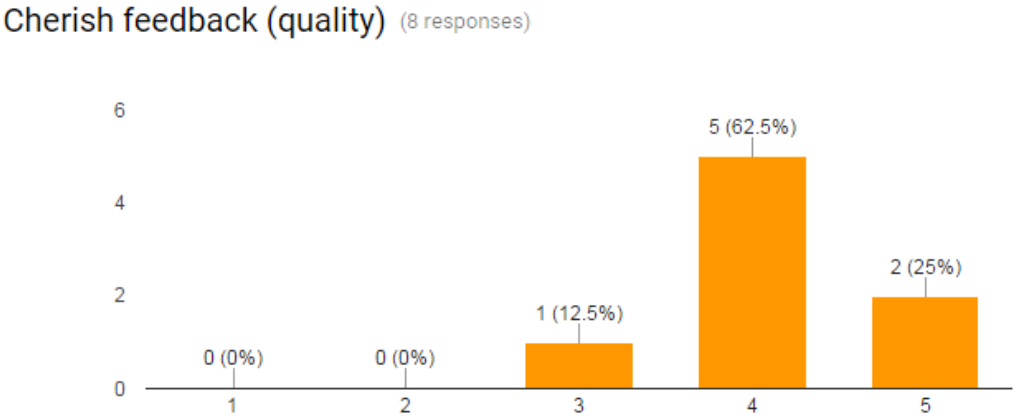


FIGURE 56 Survey - Values - Cherish feedback

The third section under values was delivery. Values under the section were: "Software evolves continuously", "Automation creates efficiency", "Iterate and improve" and "Measure and evolve". As a section delivery didn't get so high grades from the responders as other main value categories. Responders see that under the section values related on automation and iterative development were the most important ones for the team. (FIGURE 57 Survey - Values - Software evolves continuously, FIGURE 58 Survey - Values - Automation creates efficiency, FIGURE 59 Survey - Values - Iterate and improve, FIGURE 60 Survey - Values - Measure and evolve)

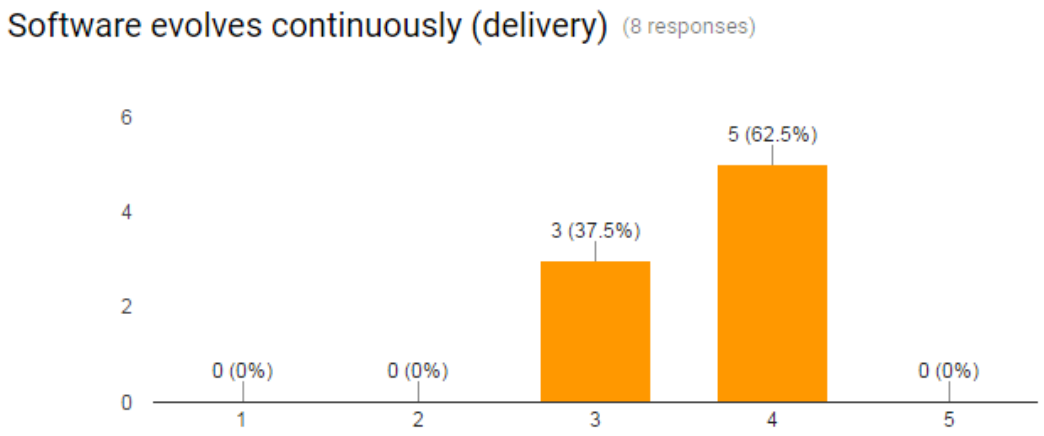


FIGURE 57 Survey - Values - Software evolves continuously

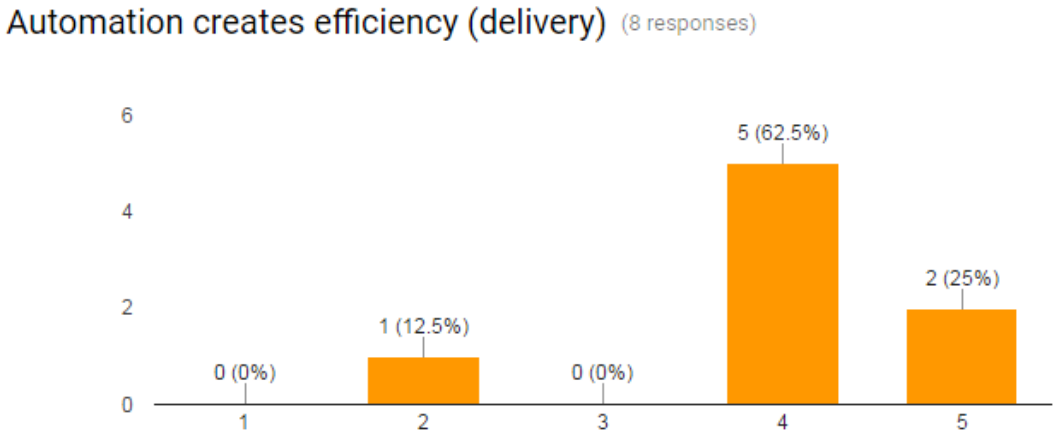


FIGURE 58 Survey - Values - Automation creates efficiency

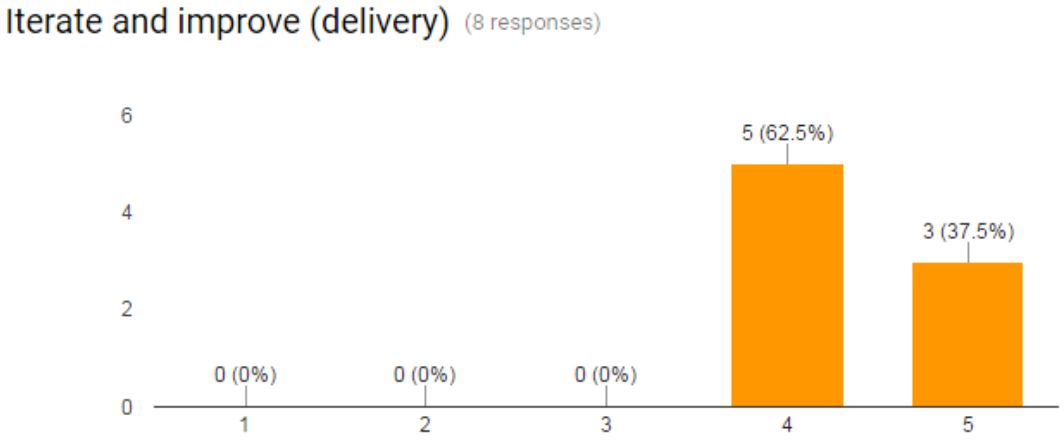


FIGURE 59 Survey - Values - Iterate and improve

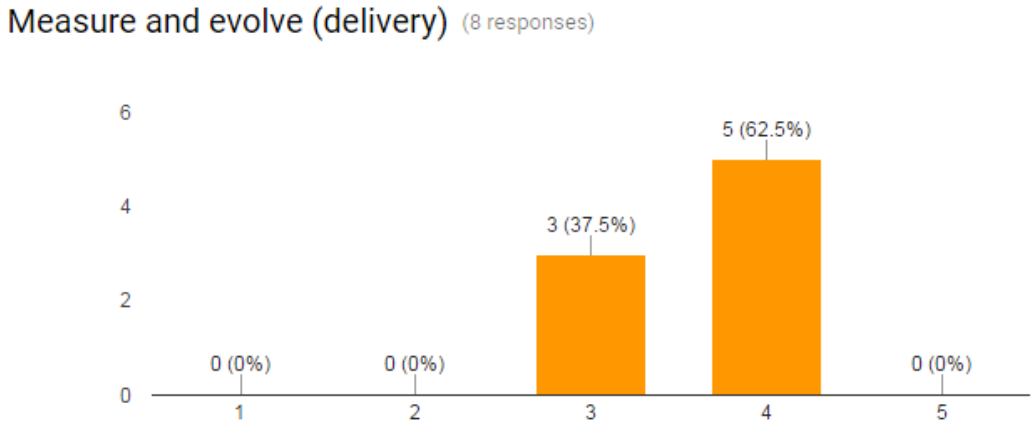


FIGURE 60 Survey - Values - Measure and evolve

Fourth section under the values was individual. Under the individual section are values regarding individual team member. These values under this section were: "Passion for the work", Being proud of own work", "Commitment and responsibility", "Trustworthiness" and "Respect colleague". As seen already in the interviews "Commitment and responsibility" –value is the most important value from team member individual perspective. Team works well when everyone on the team can rely on colleague and they can be sure that everyone commits to their work and take care of their tasks. In the survey people had also possibility to add and comment each section. One of the responders highlighted that there could be also value which would refer to personal development. This is something what team should also consider within the team how each of the team members develop their personal skills. In practice this kind of personal development can happen example by rotating people between the projects, learning by doing and giving opportunity to learn via trainings. (FIGURE 61 Survey - Values

- Passion for the work, FIGURE 62 Survey - Values - Being proud of own work, FIGURE 63 Survey - Values - Commitment and responsibility, FIGURE 64 Survey - Values - Trustworthiness, FIGURE 65 Survey - Values - Respect colleague)

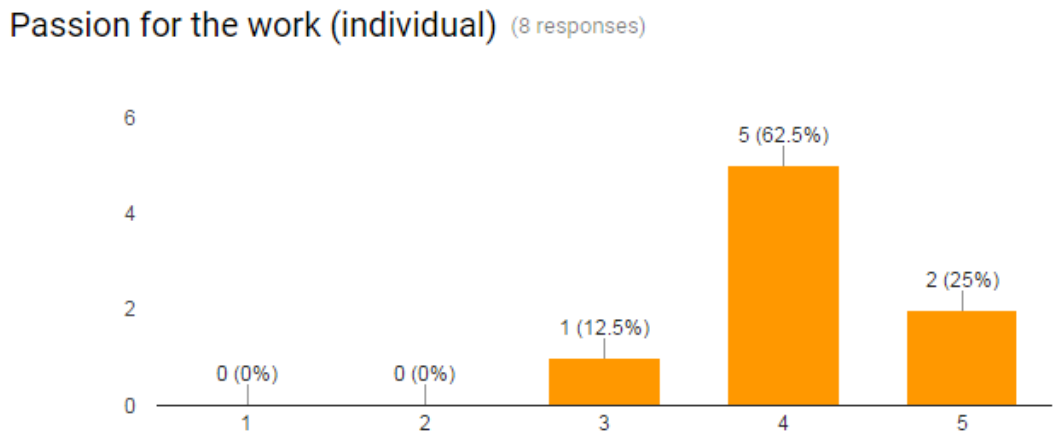


FIGURE 61 Survey - Values - Passion for the work

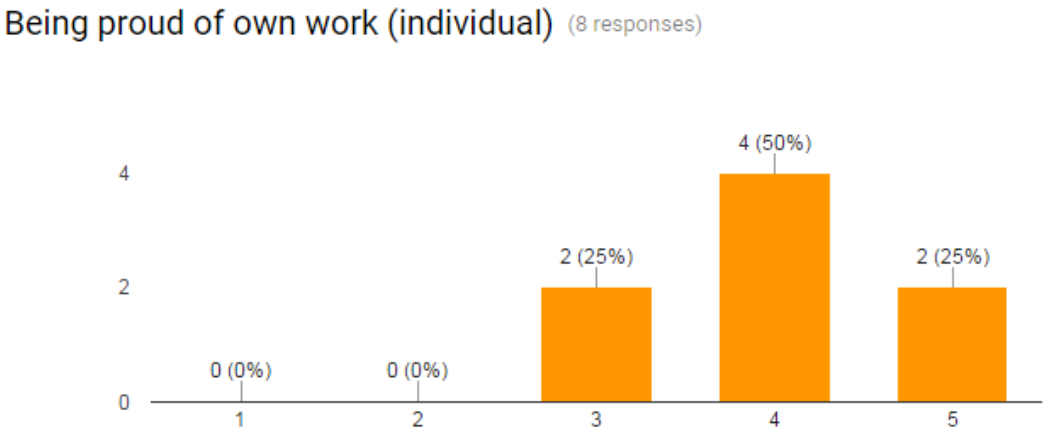


FIGURE 62 Survey - Values - Being proud of own work

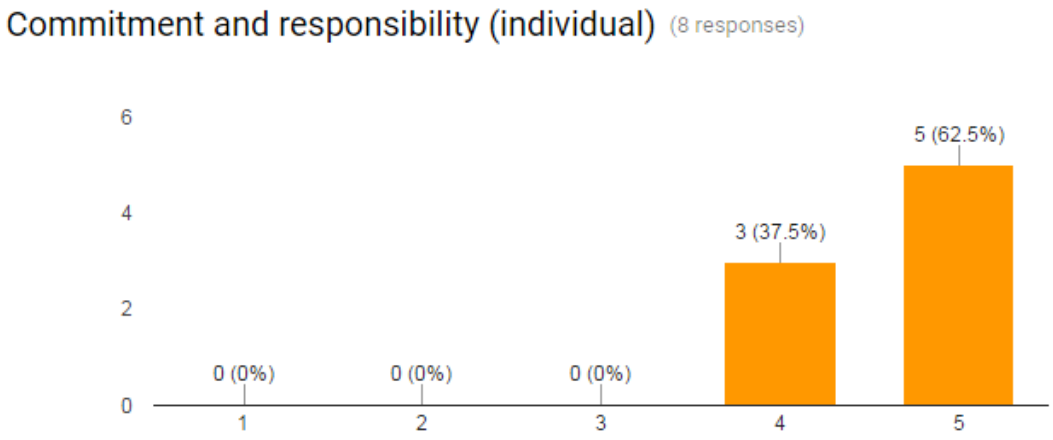


FIGURE 63 Survey - Values - Commitment and responsibility

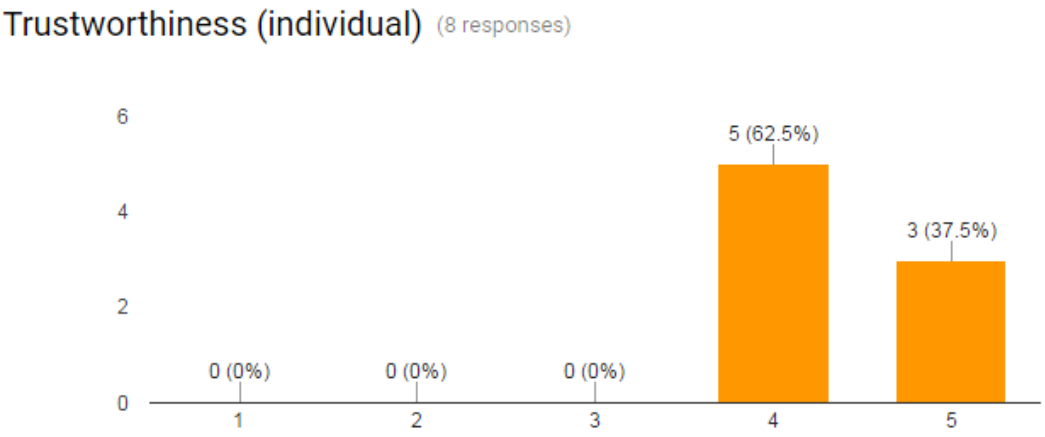


FIGURE 64 Survey - Values - Trustworthiness

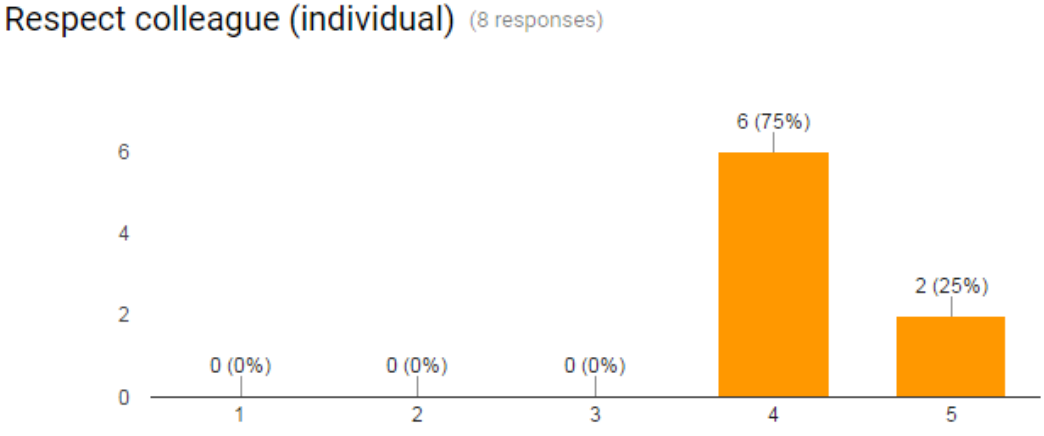


FIGURE 65 Survey - Values - Respect colleague

In the survey method's objectives were listed as previous artifacts and every responder has possibility to give grade between one and five for the each objective. Grade number five was the most meaningful for the team and grade one was the opposite. Method objectives which were listed in the survey were: "Value creation", "Reducing silos", "Higher productivity with lower costs", "Improved predictability", "Better customer and end-user satisfaction", "Faster time-to-market", "Improved collaboration", "Satisfying working environment", "Lower risks", "Common ways to work" and "More maintainable software".

Most of the responders were thinking that "Value creation", "Better customer and end-user satisfaction", "Satisfying working environment" and "Improved quality" with "More maintainable software" are the most important objectives. In the results it is interesting to see that "Faster time-to-market" is not so important for the team members. This shows again that team members doesn't see business objectives as important as example business or product owners. This kind of behavior is quite normal as people who work with development and software want to make sure that they provide good service or products to customers, they provide value and great user experience for the end-users within the satisfying working environment. Those matters are important for them and making business with created products is important for those roles which have business responsibility. (FIGURE 66 Survey - Objectives - Value creation, FIGURE 67 Survey - Objectives - Reducing silos, FIGURE 68 Survey - Objectives - Higher productivity with lower costs, FIGURE 69 Survey - Objectives - Improved predictability, FIGURE 70 Survey - Objectives - Better customer and end-user satisfaction, FIGURE 71 Survey - Objectives - Faster time-to-market, FIGURE 72 Survey - Objectives - Improved collaboration, FIGURE 73 Survey - Objectives - Satisfying working environment, FIGURE 74 Survey - Objectives - Lower risks, FIGURE 75 Survey - Objectives - Improved quality, FIGURE 76 Survey - Objectives - Common ways to work, FIGURE 77 Survey - Objectives - More maintainable software)

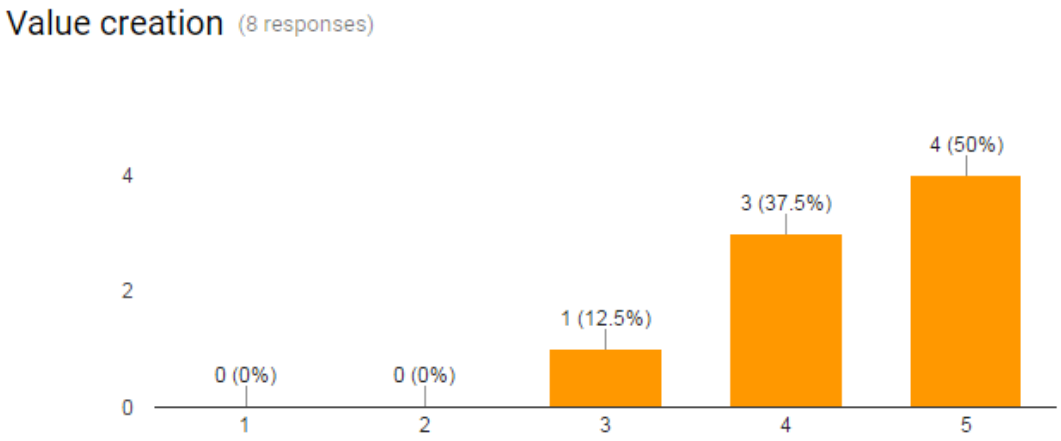


FIGURE 66 Survey - Objectives - Value creation

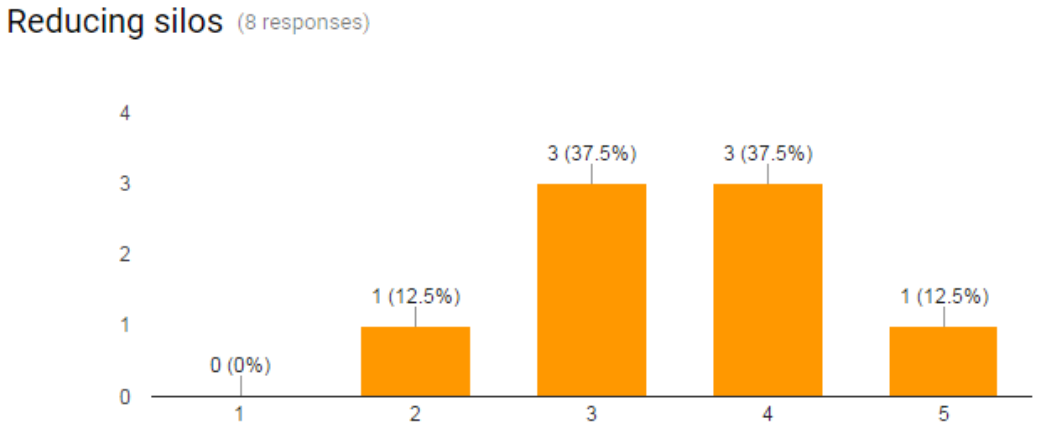


FIGURE 67 Survey - Objectives - Reducing silos

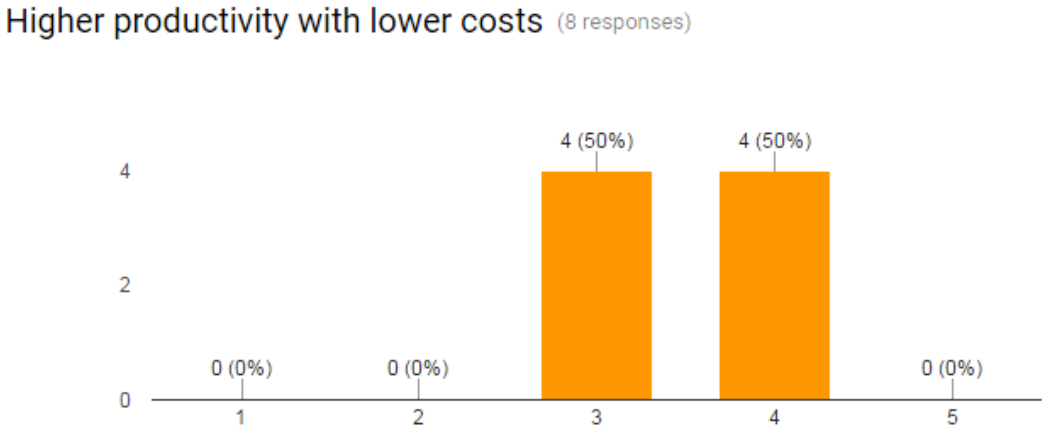


FIGURE 68 Survey - Objectives - Higher productivity with lower costs

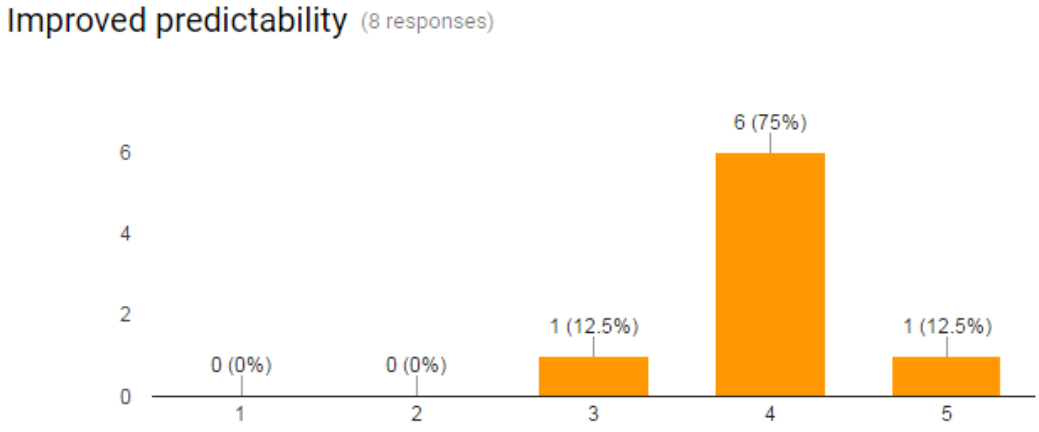


FIGURE 69 Survey - Objectives - Improved predictability

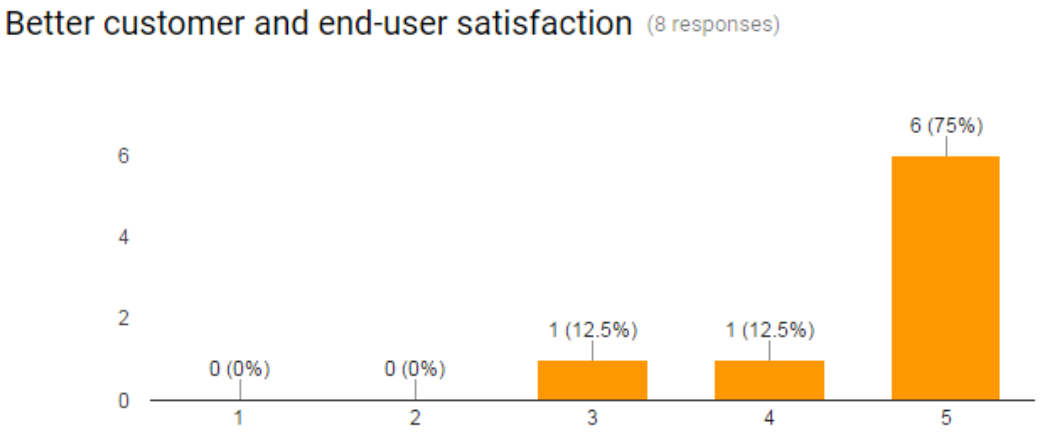


FIGURE 70 Survey - Objectives - Better customer and end-user satisfaction

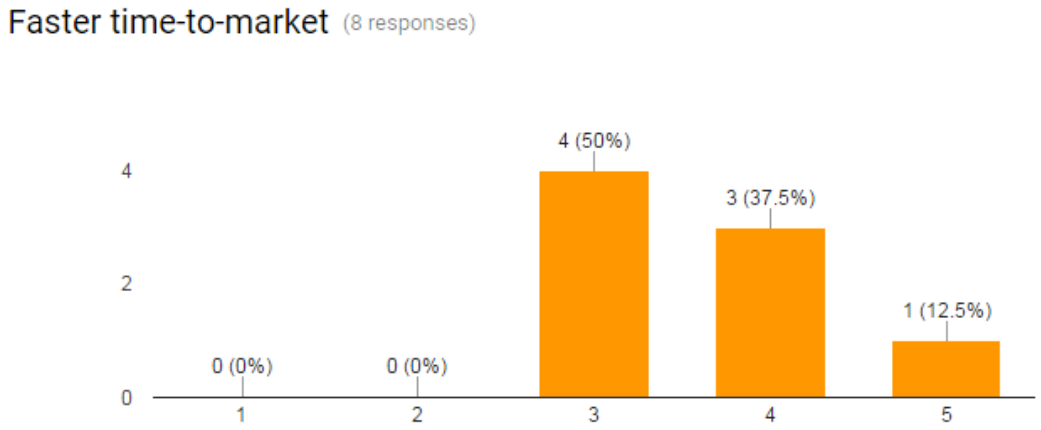


FIGURE 71 Survey - Objectives - Faster time-to-market

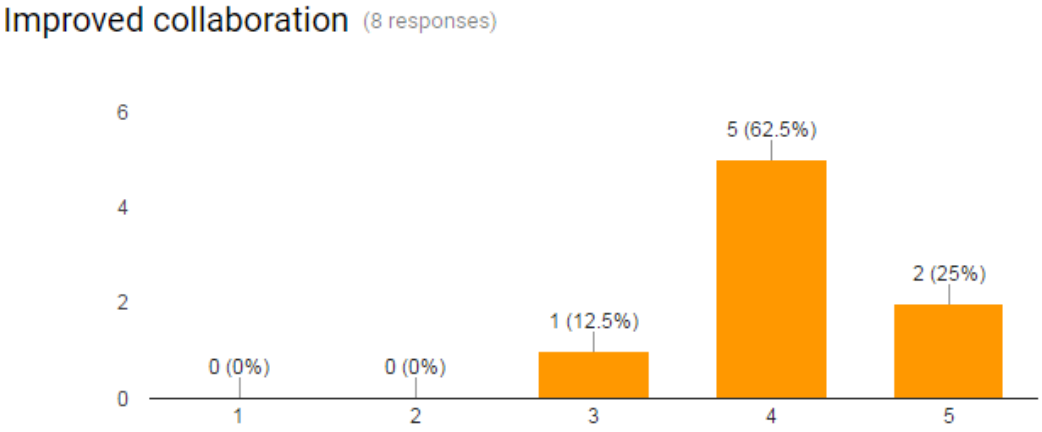


FIGURE 72 Survey - Objectives - Improved collaboration

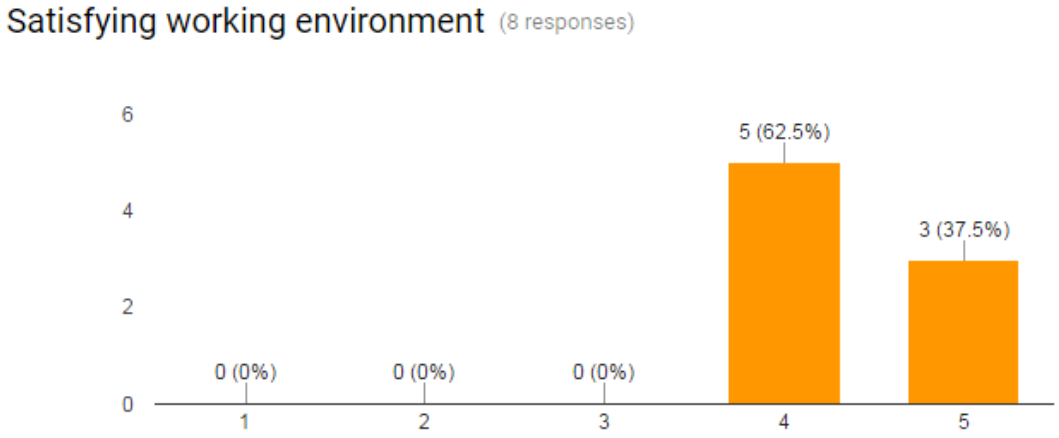


FIGURE 73 Survey - Objectives - Satisfying working environment

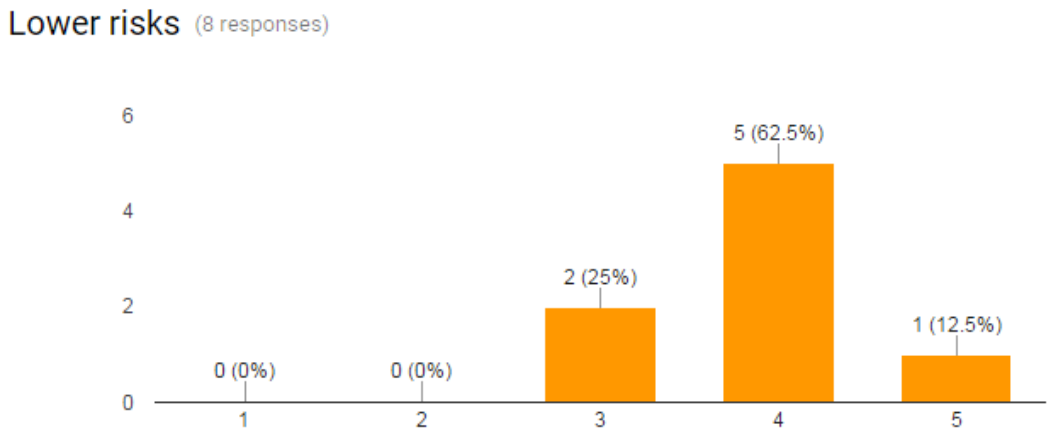


FIGURE 74 Survey - Objectives - Lower risks

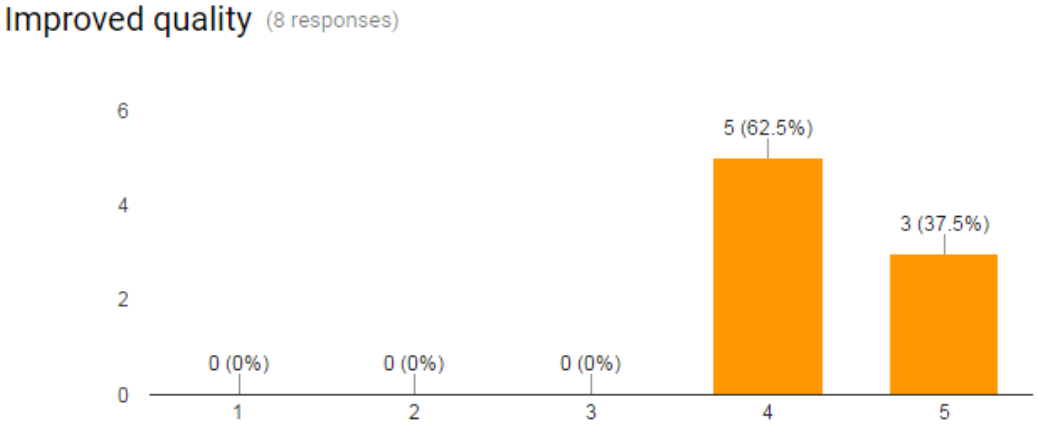


FIGURE 75 Survey - Objectives - Improved quality

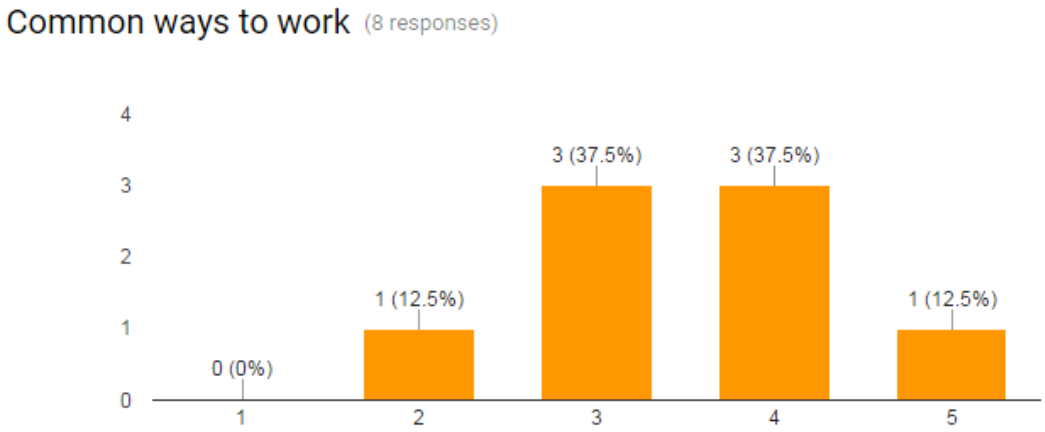


FIGURE 76 Survey - Objectives - Common ways to work

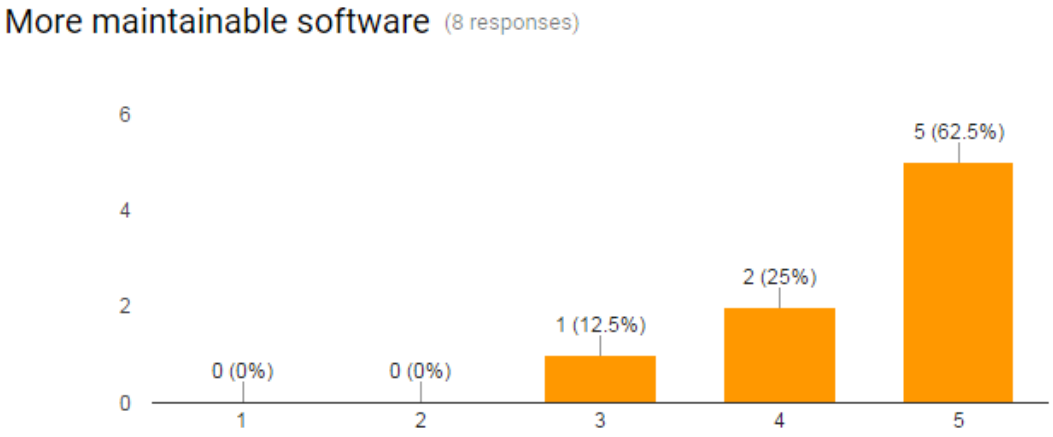


FIGURE 77 Survey - Objectives - More maintainable software

One part of the survey were principles for the team. These principles were shown as a group for the responders and they evaluated how well these principles would work in practice. Responders had possibility to give grade between one and five. Grade number five was the best one and one was opposite. Responders evaluated which principles are most meaningful for the team when they want achieve good results. All responders gave grade four or over for the next listed principles (FIGURE 78 Survey - Principles - Overall rating). Each responder choose five the most important principles for the team. They see that "Working as a team", "Continuous collaboration", "Iterative development", "Rapid feedback" and "Prioritize always" are the most important principles (FIGURE 79 Survey - Principles – The most important principles). All these principles are the fundamentals of agile software development. "Teamwork", "close collaboration

within the team", "with customer with prioritized backlog" and "iterative development" are crucial in a sense of team results. Responders also highlighted that getting feedback in fast pace makes their daily work easier.

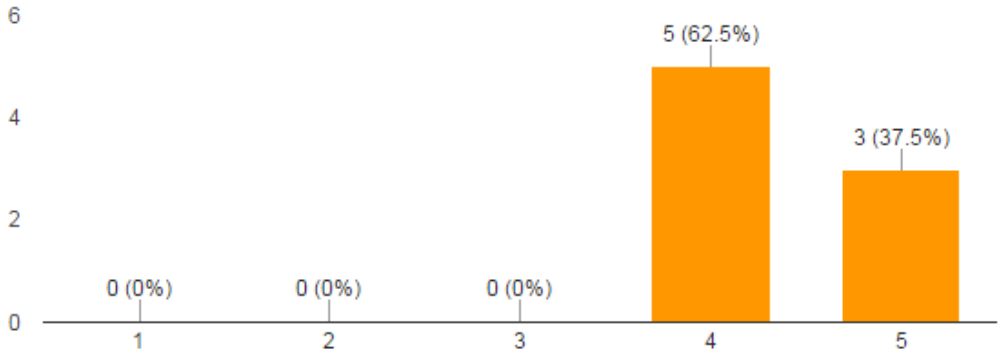


FIGURE 78 Survey - Principles - Overall rating

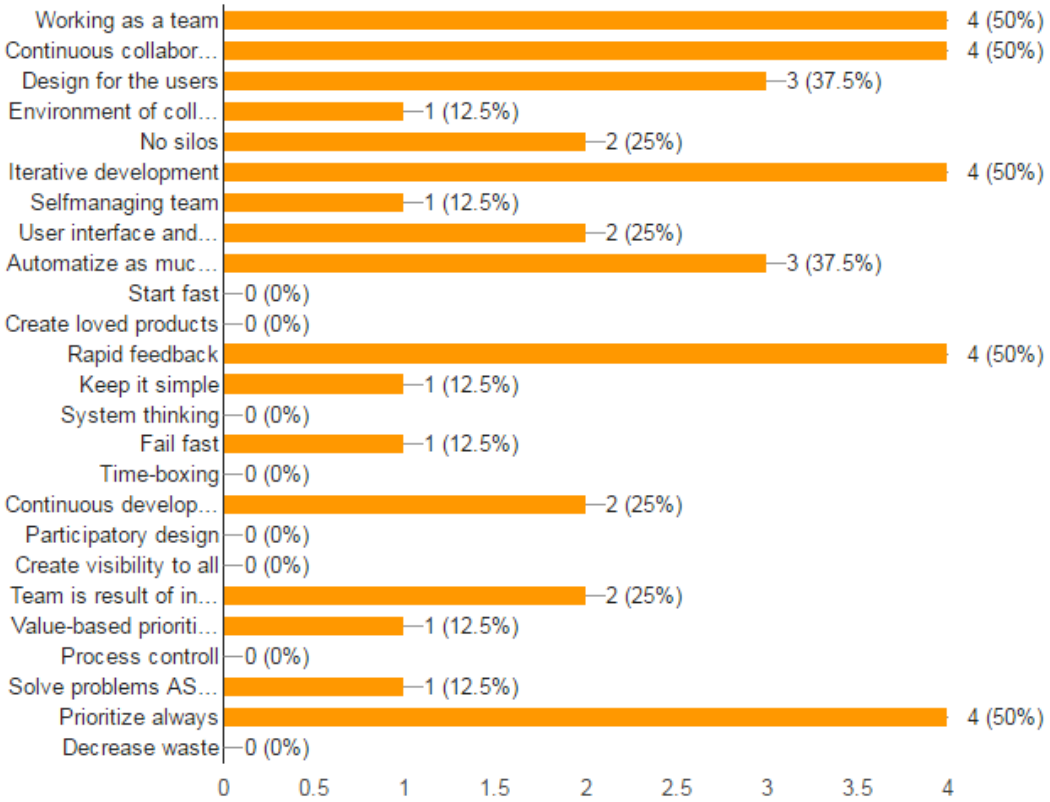


FIGURE 79 Survey - Principles – The most important principles

Like principles, the rules were evaluated similarly. Each member evaluate between one and five how well team would work by following the listed rules. Over 60% of responders think that by following listed rules team would work really well. Responders think that "Helping colleague" is the most important rule. This

confirms that in software projects team working is really important and team members are demanding and expecting to get help from colleagues. In daily work backlog refinement was sawn also important as team members work will be easier when they have clear tasks available. Other important rules that responders emphasized are "Ask if something unclear", "Understand the architecture" and "Be proactive" (FIGURE 81 Survey - Rules - The most important rules).

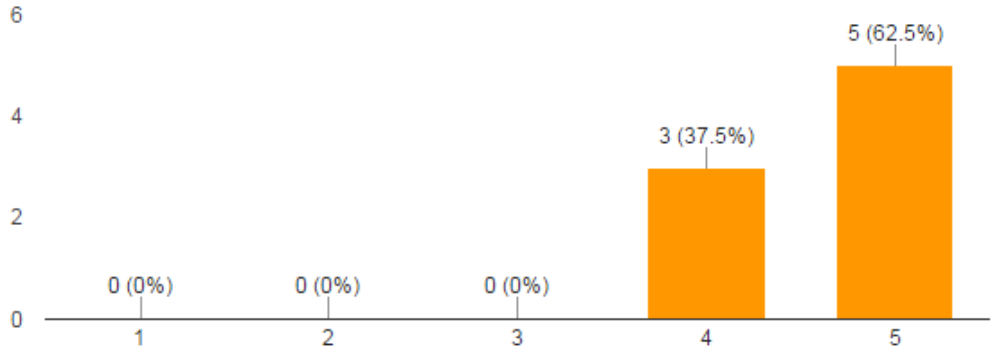


FIGURE 80 Survey - Rules - Overall rating

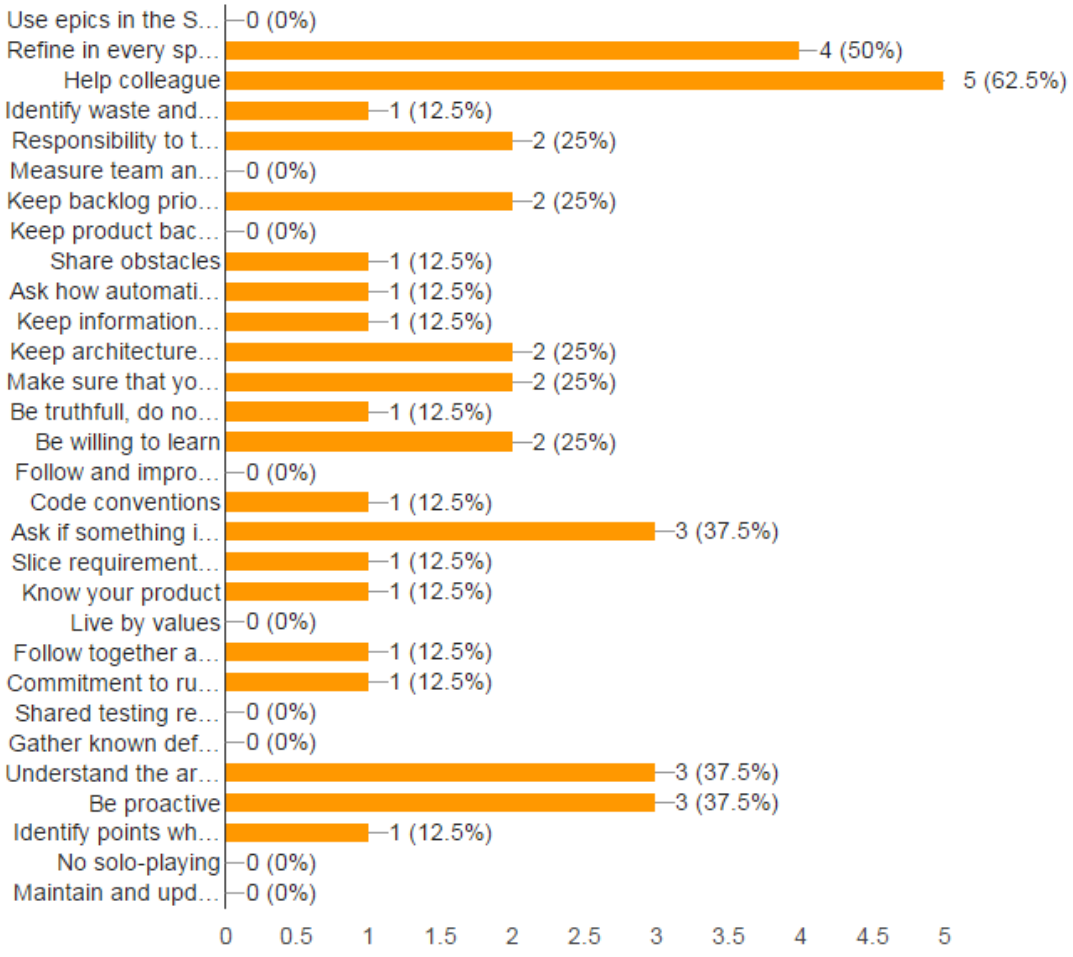


FIGURE 81 Survey - Rules - The most important rules

Processes and roles were evaluated with overall rating. As previous questions in the survey grade one was the lowest one and five was the highest rating. Almost 90% of responders thought that process would work well or very well for the team (FIGURE 82 Survey - Process - Overall rating). Also roles seem to match very well for the purpose (FIGURE 83 Survey - Roles - Overall rating). As previous sections responders had possibility to give comment and one of the responders gave feedback that roles cannot be so tightly matched and there can be several roles for one person. This actually confirms researcher's own thoughts about the person with several roles depending on individual competences. At the end of the survey responders were asked to answer how well method with previously mentioned artifacts would work. All of the responders thought that as an overall the method would work well or very well for creating and maintaining software and the same time fulfilling the customer and end-user expectations (FIGURE 84 Survey - Overall rating of whole method).

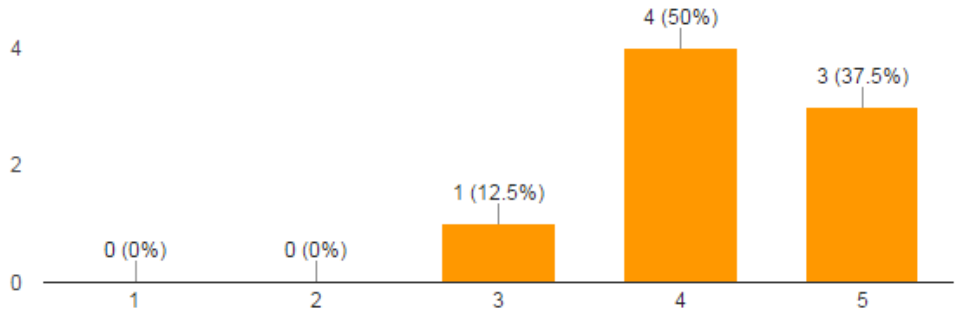


FIGURE 82 Survey - Process - Overall rating

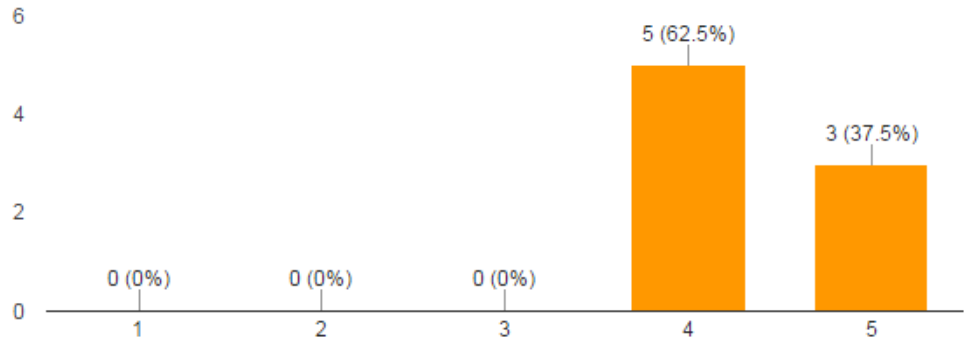


FIGURE 83 Survey - Roles - Overall rating

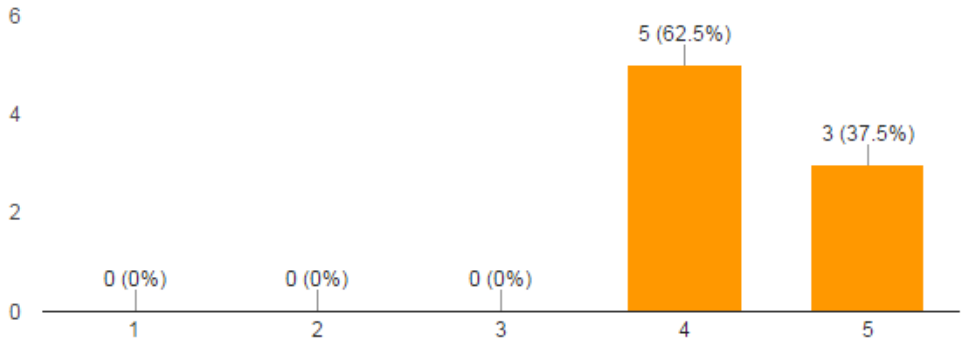


FIGURE 84 Survey - Overall rating of whole method

8 DISCUSSION

The main question for the research was: "*how to continuously develop and maintain software while fulfilling customer and user expectations?*" Result of this research is a method which structure is based on ISD method model. Baseline for the method comes from currently widely used methods like Scrum, DevOps and UCD but research's uniqueness comes from combination of these methods. The research´s purpose was to find method which would create relevant artifacts for the teams which are working with the software. Created method can be used as a baseline for the teams but it doesn't include detailed instructions for daily work (as don't other public method either).

As mentioned earlier theoretical background for this research comes from Scrum, DevOps and UCD methods. Scrum method provided the software development part and DevOps expanded the method with IT operations perspective. The reason why UCD has brought to this method is user experience and usability of services and software. Combining UCD together with Scrum and DevOps it was possible to create method which has usability view as a part of the software development and maintenance.

The method generation process started from with theoretical method hypothesis which was evaluated with IT professionals. Based on evaluations and interviews the method was refined, improved and updated to the next version. The end result of method is combination of literature and practical experience of IT professionals. Updated version of method was evaluated with the survey. This survey included questions where responders evaluated how well created method would work in their work and within their teams.

In practice this method brings values, objectives, principles and rules to follow in daily work, working process with inputs and outputs for each activity, roles for the members and example toolkit for the team. This method can be used by any team which is providing software services. As the survey results proves IT professionals believe that the method would work well or very well. All responders evaluated the method between four and five grades (five was the best one). (FIGURE 84 Survey - Overall rating of whole method)

8.1 Comparison to continuous software engineering

In software development and maintenance it is nowadays important to keep and maintain the speed of development even the project phase has ended. End-users and customers are expecting continuous development after the system procurement also. In the market there are several tools which are helping teams to improve continuous software engineering but also same time it is a change for the development culture. Fitzgerald and Stol (2015) have researched continuous software engineering and they see the same fundamental change from waterfall

method to flow-based engineering where various continuous processes are following each other. They introduced model where DevOps follows BizDev processes iteratively. In their research Continuous Planning and Continuous Budgeting sub-phases are part of the Business Development phase and under the Development there are sub-phases: Continuous integration, Continuous Deployment, Continuous Delivery, Continuous Testing and Verification, Continuous Security, Continous Compliance and Continuous Evolution. Under Operations there are sub-phases: Continuous Use, Continuous Trust and Continuous Runtime Monitoring. And all of the above create environment for continuous improvement and experimentation with innovation. If compared to created method to Fitzgerald and Stol (2015) model this method contains same fundamentals as method in this research. Their research includes also large scale agile development aspect for the enterprise level which wasn't in the scope of this research. On enterprise level organization can leverage SaFe framework. The method which is introduced in this research can be also utilized together with the SaFe framework as it is planned to be used in team level. In SaFe teams are called Agile Teams. DevOps is also covered in their research and it contains most of the parts same matters which are bound to the created method in this research. Fitzgerald and Stol (2015) have mentioned leadership principles; "Customers. Focus everyone on improving customer outcomes, not on hierarchical relationships; Organization. Organize as a network of lean, accountable teams, not around centralized functions; Responsibility. Enable everyone to act and think like a leader, not merely follow the plan. Autonomy. Give teams the freedom and capability to act. Do not micromanage them; Values. Govern through a few clear values, goals, and boundaries, not detailed rules and budgets; Transparency. Promote open information for self-management, do not restrict it hierarchically." These principles are in line with research results and the created method complies with these. Fitzgerald and Stol (2015) covered also budgeting, business strategy and planning perspectives which were out of this research scope. As in this research Leanthinking was one of the fundamentals so it has been on Fitzgerald's and Stol's (2015) research. They are focusing on flow and batch sizes and the same time keeping focus on decreasing waste and continuous improvement. Otherwise their research results are pretty much aligned with created method and the biggest difference is that on this research many of the artifacts are covered and refined to deeper and more detailed level. Fitzgerald's and Stol's (2015) research confirms that the created method in this research has included right components and fundamentals no matter the used terms are slightly different in some artifacts.

8.2 Contribution to science

When comparing the results to literature it can be seeing that currently literature focuses on certain areas of software development but it doesn't cover the change when teams need more competences based on business needs. As seen in the last year typical boundaries are fading away and teams are built to match objectives

and business needs. Example if business wants to highlight the meaning of usability of services and cybersecurity in the strategy these changes must reflect on team´s allocation and to professional's daily work. At the moment literature focus on certain parts but as research proves ISD method model can be used to leverage currently used methods. As the business needs will change also in the future this kind of research can be used to understand how change can be managed and what things will change in a fundamental level. As seen during the research bringing new competences to the team will also effect on fundamentals (like values) in the team. When business changes and new competencies have brought to team it has to reflect example on values. Like notified during the research old silos are still alive example between operative and development team members. Without bringing the new perspective and updates to values and objectives team members might have target misalignment between each other's.

If compared to the literature this created method is completely something new what current literature doesn't cover. It brings new perspectives to software development. The wideness of research is also unique as it covers main areas during the whole product lifecycle (FIGURE 85 Context of the method). The method includes all relevant ISD method model parts and it can be leveraged many ways in future researches. Every artifact of the method is thought to be tightly combined with each other and example every process phase has own inputs and outputs. All the roles and responsibilities are planned to match with processes. The method includes also example tools which are suitable for the software teams and those match together with other artifacts in the method.

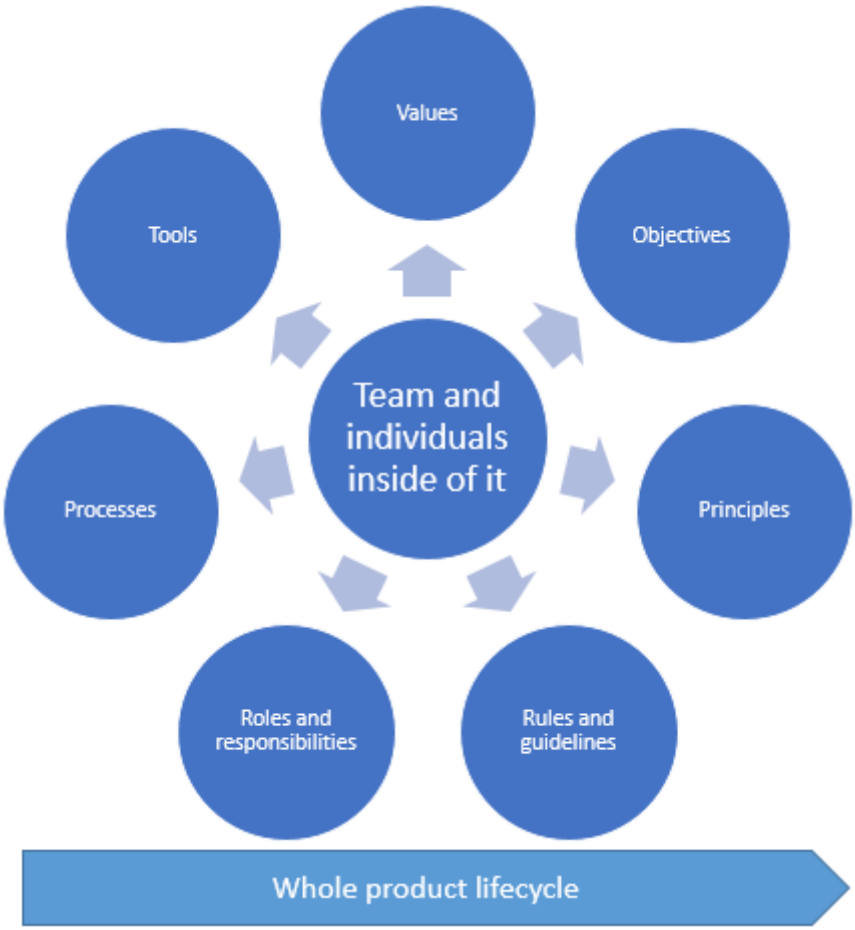


FIGURE 85 Context of the method

In the beginning of the research is introduced very limited process chart which has been used several years in the sector. This process includes in high level following processes; requirement, planning, delivery and development phases (FIGURE 2 Software Development Process). During the software development evolution processes have become iterative (FIGURE 4 Agile Method). Iterative software seems to work better especially in software business where final result is unclear at the beginning of software development project. One part of this research has been the process for the software team which includes all activities from the beginning of development all the way until maintenance phases with iterative approach. The process is rather complex if compared to methods in the background but all activities inside of the process has own purpose and each activity complements other. This process can be utilized in any software team and users can develop it further. The method includes also inputs and outputs for the all process's phases and these must be updated same time when updating the process. Inputs and outputs of each activity helps team members to understand how each activities are related on others and which kind of results each activity should generate.

Swartout (2012, 71) and Hüttermann (2012, 12) encouraged to build software that is made by and for the people. As Hüttermann (2012, 12) and Swartout (2012, 71) highlighted the meaning of user likewise the UCD´s philosophy complements the software development purposes by putting the user of the product, application, or experience at the center of design process (Pratt, 2012, 12). Swartout's, Hüttermann´s and UCD´s approaches for the software development are more meaningful nowadays when services must to be thought from the user perspective already at the first phase. The market nowadays is requiring software companies to create products and services which make users feel happiness, satisfaction and even love those used products (Kraft, 2012, 12). Values (Agile principles) in the Agile Manifesto are also still valid and many of them (if not all) are good to agile teams understand (FIGURE 5 Agile Principles (Agile Manifesto, 2001). One part of the method was values and those are introduced in chapter 6.1. These values are essential to understand for everyone who is working in the software business. Values are evolving during the time but there are many long lasting values which remains no matter time flies by. Example high level values collaboration, quality, delivery and individual are those which will remain no matter time changes. Already on this research has seen that individual perspective will raise in future and soft values should be part of the methods. Method's values are combination of other development and maintenance methods and it is updated based on professional´s comments.

If method values are setting the normative ethics for the software development those are reinforced in the research with objectives, principles and rules. All previous artifacts are introduced in chapter 6. These artifacts compliments the method's values. Objectives gives the understanding for the team what they should be aiming for. Research's end result regarding these artifacts compliments Pratt's (2012, 12) philosophy where end-user is put in the middle of development. In the method user satisfaction is not only part of the development process it is included in the all artifacts in the method. In the objectives part of it user-salience is seen in 'better customer and end-user satisfaction' which was also one of the most important value when asked from the professionals. Previous artifacts also complements the Agile and Lean artifacts and they are trying to concentrate on people/individuals, fast/continuous delivery, quality and simplicity (Poppendieck & Cusumano 2012). The method's principles and rules are near to daily work and same time those are guiding team members in their actions. Principles and rules have relation to values and objectives but those are also combined tightly to daily processes. Without relation to 'higher' level values and objectives, rules/principles doesn't shift to daily work. Also process and tools are combined tightly to rules and principles and they compliments others. As Longbottom (2014) said that DevOps essentially tries to bring development and operational teams closer together through automation to support speed of the business requires this method includes also includes set of tools to support his idea. These tools compliments the other artifacts inside the method. Mainly those are designed for cloud-based web-application development but they can be used in

other development and environment types also. If used in other type of environments users should update the tool list. Tools are also evolving all the time, so it is good to understand that it is on team responsibility to update tool and technology stack regularly. This way team can make sure that their tools and used technologies are supporting the other artifacts in the method.

The method has also content in every artifact which was related on individual development and it includes matters which are looking the software development and maintenance from individual perspective. These aims to bring team members more near to daily processes and other artifacts. People are in essential role in the software business and therefore they are involved in the method as is highlighted in the CD and DevOps layers (Swartout, 2012, 87).

The method can be used in the teams as it is or teams can take part of it to use. It is important to understand that every artifact can be or even must be updated based on the organization, business or team needs. Anyway it provides a baseline for the software development and maintenance work and same time focus is kept at end-users which was the original objective of the research.

8.3 Contribution to practice

The result of this research is a method which can be used as a baseline for the software development and maintenance work. Every team no matter have they worked earlier together or not can found something to their daily work from the method. The method provides fundamental artifacts for the daily work like values, objectives, principles, rules, roles and processes. Without this kind of fundamentals it is more difficult to start or run successful software teams. While researching the method and doing the survey there was founded also focus areas which are important to software teams and leaders of those teams. As the survey proved all responders think that generally method would work well or very well (FIGURE 84 Survey - Overall rating of whole method). In the survey professionals highlighted different matters what they seem to be most important. These are the most important findings of the research and every software team and team leaders should ask from themselves how we can improve these things for become a better team. Based on the research every team and team leaders should focus on improving collaboration, working software; quality of it, maintainability and user-experience, team commitment and carrying responsibility (FIGURE 86 Values to focus for the team and team leaders).

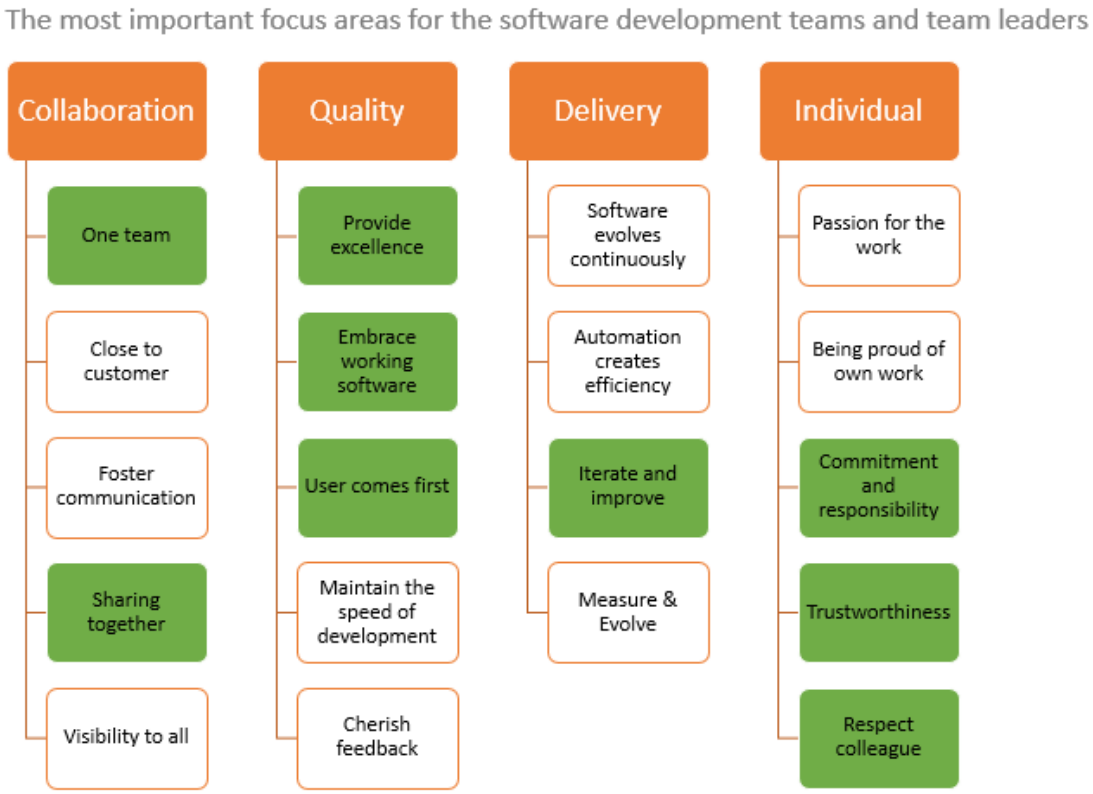


FIGURE 86 Values to focus for the team and team leaders

When setting the targets and objectives for the team and himself/herself leaders should focus on value creation, better customer and end-user satisfaction, satisfying working environment, quality improvements and maintainable software (FIGURE 87 Objectives to focus for the teams and team leaders).

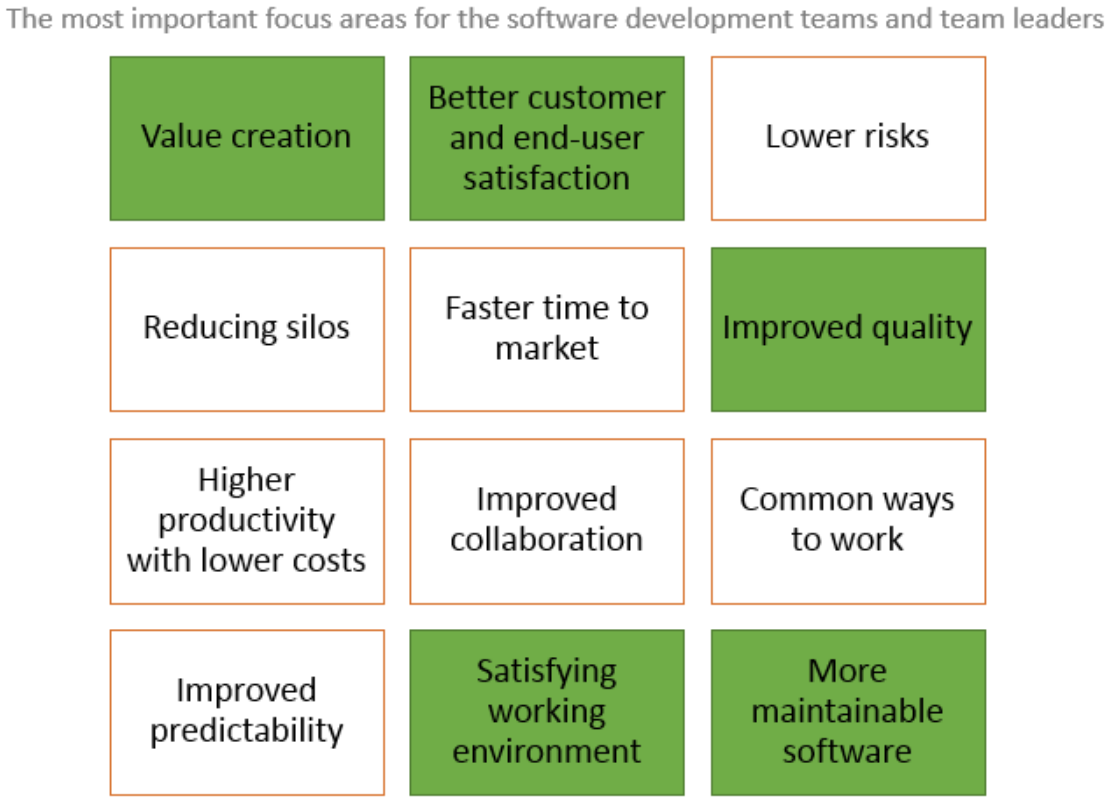


FIGURE 87 Objectives to focus for the teams and team leaders

Most important focus areas regarding the principles shown that answers are aligned with values and objectives. Professionals want that team and leaders focus on; working as a team, collaboration, designing for the users, automation, rapid feedback and prioritization.

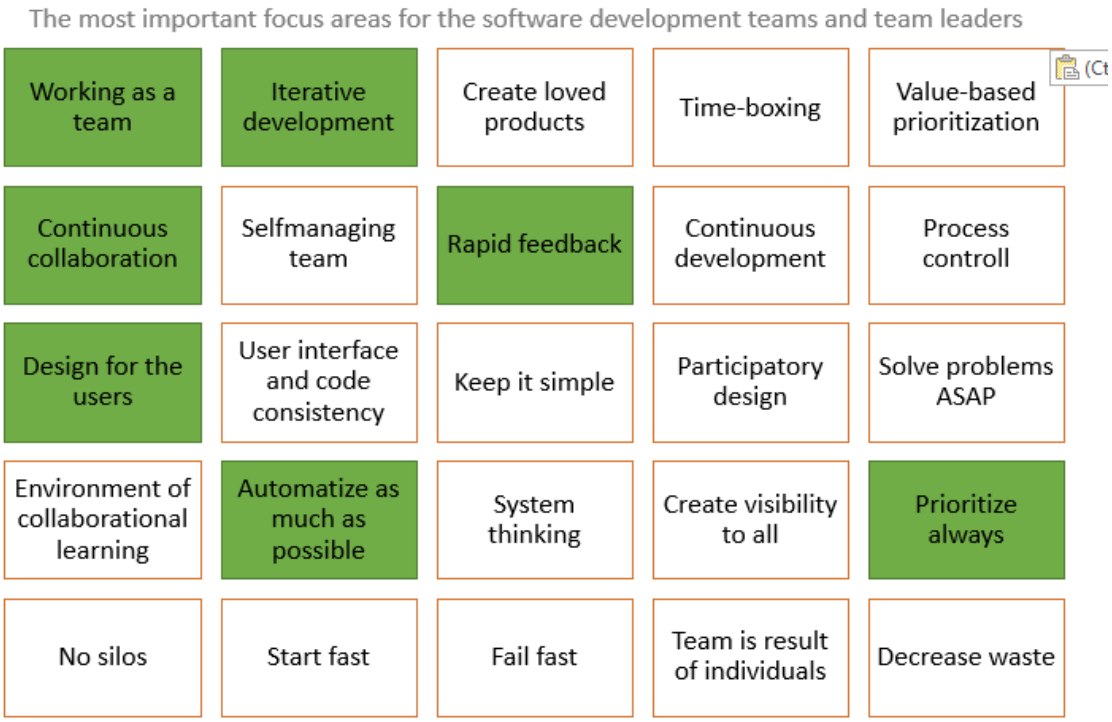


FIGURE 88 Principles to focus for the teams and team leaders

When team and leaders are creating working environment they should focus on creating environment where everyone helps each other. Professionals have highlighted that helping colleague is the most important rule in the team and teams should focus on that. Other rules what teams and leaders should focus on are: Backlog refinement, asking when team member doesn't understand something, understanding the architecture and proactive attitude. The most important areas within the rules artifact are related on team culture and openness. When the team and leaders can create environment where people can act proactively, ask help from team member and getting help from them team is already in the right track (FIGURE 89 Rules to focus for the teams and team leaders).

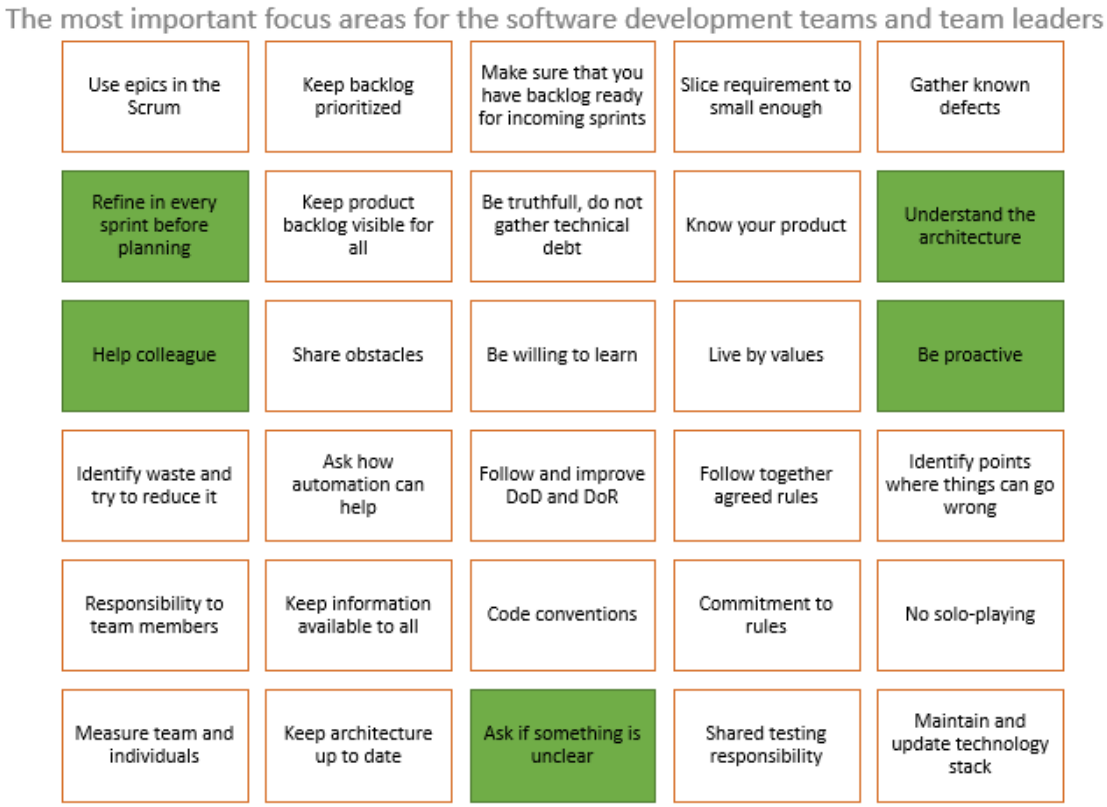


FIGURE 89 Rules to focus for the teams and team leaders

As the results shown professionals think that the process and the roles are in good level but getting in very good level needs still improvements. These improvements will arise to the team within the time when they evolve their own processes, working culture and other matters which are affecting to the team results.

9 CONCLUSIONS

The main question for the research was: "*how to continuously develop and maintain software while fulfilling customer and user expectations?"* In the research was sought a method which would help IT teams continuously develop and maintain software and the same time fulfill customer and end-user expectations as well as possible. At the beginning one of the main purposes was also to bring team members more near each other and that way improve team collaboration. The purpose was to create method which covers whole product lifecycle rather than including some certain phases from it.

The research was started by scoping it. During the scoping it was already clear that it would be one of the most tedious tasks in this research. When speaking about software development and methods around it is easy to expand scope too much. During the research focus was on methods which were assumed to match together with defined objectives. On a high level software service includes typically some server components, network and software. Therefore DevOps and Agile methodologies were chosen. Agile methodologies are giving for the method software development perspective and DevOps completes software development with operative functions. As one of the objectives was also to create method which pursues to better end-user experience therefore UCD was one of the chosen parts of the method. Research started by researching more about these chosen methods and by understanding what those actually includes. When all chosen methods were examined the first version of new method was generated based on the literature and practical knowledge. When creating the artifacts to the method there is a need for method structure. The purpose was to understand what a method typically includes; what kind of components and parts there should be in the method and what kind of information is needed. After understanding what the method should include and examination of DevOps, Scrum and UCD best parts of each method were combined together. One of the most difficult part of this research was actually combining these chosen methods together reasonable way. When creating method purpose was to pick up best parts of each method and all the time keeping focus into creating the process where every activity completes each other. Main thing during the researching was to find out clear inputs and outputs for each activities.

When the method was created and it was aligned with context of the ISD method and material for the interviews was prepared. Created research material was actually useful as it includes charts for evaluating purposes at the interviews. Interviews included open questions and evaluation part where interviewed people had possibility to comment the method. When all interviews were done the method was refined and updated to next version based on the comments got from interviews. After the method update it was evaluated with the survey. Survey's target was to get feedback from the professionals to question "how well this method would work in practice" and "which things are the most important ones in each section".

The end result of this research was the method for software teams to guide their job at high level. The method includes relevant information components and artifacts. The baseline of the method is constructed from chosen methodologies (DevOps, Scrum and UCD). On later phase it has been updated based on professional interviews. When summing up the research there are some main questions to ask. Would the method be useful to software teams? Based on the survey results it would definitely help many teams but at the same time there are artifacts which should be still updated (FIGURE 84 Survey - Overall rating of whole method). As all methods they give fundamentals how teams should work (values, principles, rules, roles and responsibilities) and processes to guide their work but meanwhile there are many moving and changing matters that these kinds of methods doesn't cover. Software teams are always struggling with various issues and best teams are measured how they learn and evolve during the time. Anyhow without this kind methods there is nothing to rely on team work so definitely there is a need for this kind of research work and methods.

Different IT development and maintaining methods are going to right direction when fading out different team boundaries. Beforehand many processes have baseline from some certain small part of product lifecycle aspect as nowadays same teams might take care of products from the beginning at the shutting down point. It is good that the way of thinking how teams should be built is nowadays based on required competencies. When team owns pervasive competences it also creates a foundation for growing up on individual level.

The created method can be used in every software team but also at the same time it is essential to remember that as business changes the method must be updated. On the information technology sector business is changing so rapidly that working methods need to keep up-to-date. The method is not perfect but it is great starting point or baseline for any team. Also older and more mature software teams can find new things inside on it or they can pick just some best parts of it. Survey results will provide great information for the teams, team leaders and business leaders. People in previous roles can find relevant information what professionals appreciate nowadays and what kind of working environment they want to create for themselves and for the team.

9.1 Limitations

When starting this research the preliminary idea of method combination was available. As a purpose was to focus on software development and maintenance by concentrating at the same time on user experience and user interface chosen methodologies were suitable package. At the beginning when locking in the used methods: DevOps, UCD and Scrum it was possibility that there has been maybe already limited some better methods out. By choosing these methods and limiting possible better methods out of the scope in the research was ended up on situation where research cannot actually answer to the original answer. Method which was created might be very good one but no one cannot say is the most

suitable one if compared to research objective. During the research it came clear that the method can be updated endlessly. Software development and maintenance are changing and evolving so rapidly that already while writing down this research many things have changed.

When planning the interviews it was difficult to find best possible approach and structure for the interviews. The original purpose was to ask as many open questions as possible in a way that researcher wouldn't lead people to some certain direction on their answers. At the same time afraid was that without giving some certain framework or fundamental matters answers wouldn't be relevant for the research purpose. Final decision was to end up on interview model where at the beginning the big picture of research was opened and on later part interviewed people had possibility to evaluate the created method. During the interviews some people get slightly confused if not fundamentals were opened well enough. Even it was decided to give relevant background information it was quite hard to get answers for some of the open questions. When pondering this afterwards it would be more beneficial to spend more time on introductions in the interviews.

Interviewed persons were working on pretty similar environments. This situation might distort the answers. Ultimately when doing this kind of research it would be best to get interviewed persons from totally different companies and these companies size should vary. Also projects and history regarding the projects will likely effects on answers. Best possible situation would be that you have as heterogeneous group as possible but they all have relevant knowledge of software development and maintenance methodologies.

The final survey was meant to evaluate the method after interviews and method update. As this kind of method cannot be tested without heavy investments easily on practice it was clear that method's functionality can be only evaluated by asking opinions from the professionals. When analyzing the results it was realized that the survey results are related on role within the team. Each responder evaluated the method from their own perspective. Based on this observation the results might become distorted as business perspective is not covered as well as technical.

In the software business there are many ways to build processes around it. This kind of method works best for the teams which are working with products. On a high level there are nowadays typically projects where team provides certain application for the customer and on these projects customers owns intellectual property (IP) after the project (FIGURE 90 Development approach) is done. Opposite of previous there is software product business where company owns the IP and they are offering it as a service for a multiple customers. The created method fits better for this kind of environment where development continues the whole product lifecycle. Especially bringing UCD as a part of the team is important when the complexity of the software increases during the time. Without having UCD expertise in the team and bounded it in the processes the user experience and usability starts easily become weaker during the time. When creating new products there are process phases which are necessary in the beginning as

"building the business model", "validating the business case", "concept pictures" and "MVP scoping". When these phases are done and business has decided to go forward with the product created method for the teams will be useful.

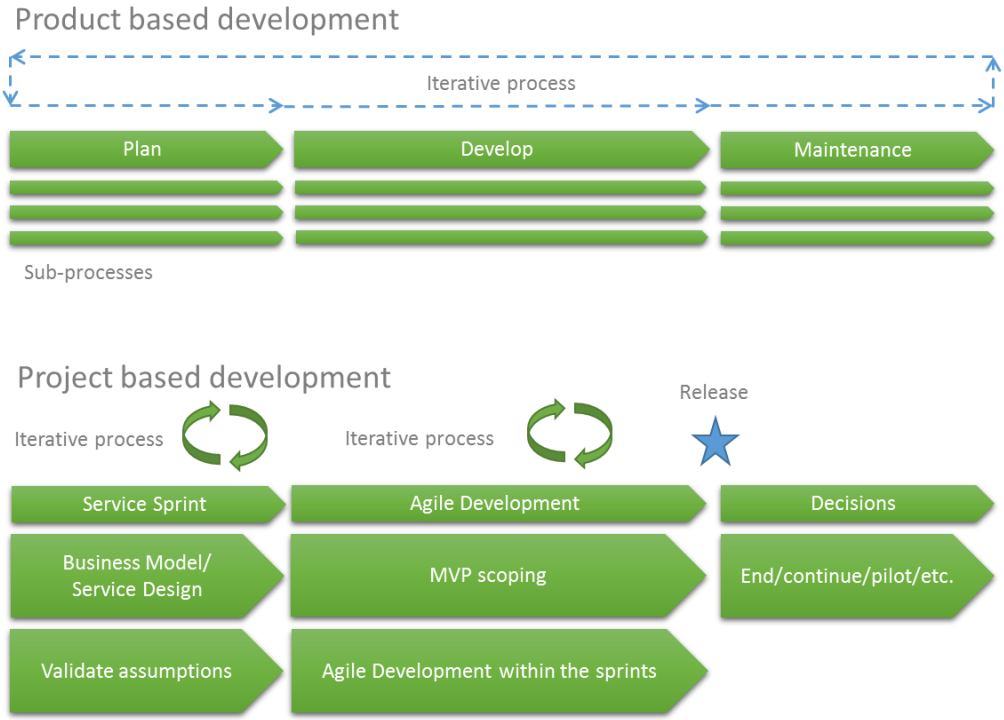


FIGURE 90 Development approach

9.2 Future research

In the future it would be beneficial to do research example how this kind of method would work in the practice. If someone could measure how team works beforehand and afterwards everyone could understand better the actual results. This kind of research requires business commitment and changes which are not necessary easily done. The method artifacts like values are bound deep in the organization culture and changing these will take time. This kind of evaluation and research would take even years depending on the starting point where the team is on the beginning. The method could be combined to some sort of maturity test which would evaluate at the first where the team is at the beginning. Combining this kind of information, planning and implementing the changes and evaluating those after decided period could be seen how things are developed.

As already mentioned earlier every kind of boundaries are fading away in the software organizations. This means in practice that resourcing of every software project starts from customer and project needs. This change is significant as this way project outputs are always better when all the relevant competencies are gathered in the same team. During the research software business has

already started to talk about DevSecOps. In practice this extends first research thoughts about the software teams where all relevant competences should be in the same team. In future someone could find out how artifacts would change if teams have more cybersecurity competence within the team. This kind of knowledge would cause changes on every artifact in the method. As notified already in the research depending on role professionals emphasizes matters which are the most important for their own roles. Of course there are a lot of things which would remain but this kind of expand would definitely bring some minor new things to the method.

When software development methods are evolving all the time it would be good to see what kind of benefits are achieved when comparing to older models. What are the most important achievements and how the team members have seen the change? It would be also important to understand how teams could move from one agile maturity level to another. What kind of matters there should be considered in each steps and how this process could be done effectively?

One of the important future research approach would be also individual perspective in the software teams. Within the research it was noticed that many of methods doesn't cover matters in individual perspective. When discussing about individual perspective in the software development and maintenance method we are on slightly grey area as we are combining pretty technical method perspective to the leadership of the team. Many of the software team leaders would appreciate the information about catalyst which are driving team in good "flow-state" where they are delivering together best possible results. There are many kind of things behind it but it would be beneficial to understand those motivation drivers for the team members. What would be the most important focus areas to achieve best results? Are those new technologies which are driving team members forward? How about good team spirit and how it could be improved in the software teams? Or are the daily methods what they are using the key for the success?

During the research it was examined also what kind of handover phases from different software product lifecycle includes. This is also interesting and relevant point as all persons are not so much involved example during the MVP creation phase and some of the persons are not so much involved with continues services. It would be interesting to understand better which kind of processes there should be example when new products are going to maintenance phase at first time. What kind of information is needed to get Service Delivery Manager up-to-date and how the responsibilities can be fulfilled after moving to the new phase in the software´s lifecycle.

REFERENCES

Agile Manifesto. (2001). Agile Principles. Retrieved 13.2.2015 from http://www.agilemanifesto.org/principles.html

Brinkkemper S. (1996). Method Engineering: Engineering of Information Systems Development Methods and Tools. *In Proceedings of the Fifth International Conference on Information Systems Development* (ISD'96), Wrycza-Zupancic (ed.). Gdansk, Poland.

Cois, C. A., Yankel, J., & Connell, A. (2014). Modern DevOps: Optimizing Software Development Through Effective System Interactions.

cPrime. (2014). Scrum Process. Retrieved 15.2.2015 from https://www.cprime.com/resources/what-is-agile-what-is-scrum/

Dingsøyr T. (2010) Agile Software Development

Farroha, B. S., & Farroha, D. L. (2014). A Framework for Managing Mission Needs, Compliance, and Trust in the DevOps Environment. *2014 IEEE Military Communications Conference*, 288–293. doi:10.1109/MILCOM.2014.54

Fitzgerald, B., & Stol, K. (2015). Continuous Software Engineering: A roadmap and agenda. *The Journal of Systems and Software 123 (2017),* 176–189

Gartner. (2013). Digitalization. Retrieved 28.4.2015 from http://www.gartner.com/it-glossary/digitalization

Gofore. (2013). DevOps – määritelmä. Retrieved 16.2.2015 from https://gofore.com/ohjelmistokehitys/devops-sovelluskehittajan-roolin-evoluutio/

Guang-Yong, H. (2011). Study and practice of import Scrum agile software development. *2011 IEEE 3rd International Conference on Communication Software and Networks, ICCSN 2011*, 217–220. doi:10.1109/ICCSN.2011.6013698

Guo, W., & Wang, Y. (2009). An incident management model for SaaS application in the IT organization. *ICRCCS 2009 - 2009 International Conference on Research Challenges in Computer Science*, 137–140. doi:10.1109/ICRCCS.2009.42

Hussaini, S. W. (2014). Strengthening harmonization of Development (Dev) and Operations (Ops) silos in IT environment through Systems approach ., 178–183.

Hüttermann, M. (2012). *DevOps for Developers*. New York, USA: Apress. doi:10.1007/978-1-4302-4570-4

Iivari, N. (2006). Discourses on 'culture' and 'usability work' in software product development. Oulu : University of Oulu, 2006

Ieee. (1990). IEEE Standard Glossary of Software Engineering Terminology (IEEE Std 610.12-1990). Los Alamitos. *CA: IEEE Computer Society*, *121990*, 69. Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:IEEE+Standard+Glossary+of+Software+Engineering+Terminology+(IEEE+Std+610.12-1990)#0

ISO/IEC (2006). ISO/IEC 14764:2006, Software Engineering -- Software Life Cycle Processes – Maintenance. Retrieved 22.2.2015 from http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=39064

Jayaratna, N. (1994). Understanding and Evaluating Methodologies. McGraw Hill Book Company, London.

Kalermo, S. (2014). Käyttäjäkeskeinen suunnittelu Jyväskylän yliopiston IT-palveluissa. Pro Gradu, Jyväskylän yliopisto

Kamppi, J. (2013). Ketterän menetelmän käyttöönotto. Pro Gradu, Jyväskylän yliopisto

Kennedy, M. (2010). Product Development for the Lean Enterprise. Oaklea Press, USA.

Koskela, A. (2014). Ketterän ohjelmistokehityksen ja kevyen käytettävyystestauksen yhteensovittaminen: tapaustutkimus

Kraft, C. (2012). *User Experience Innovation*. doi:10.1007/978-1-4302-4150-8

Longbottom, C. (2014). ComputerWeekly.com: How to tame the new IT beast called DevOps. Retrieved 16.2.2015 from http://www.computerweekly.com/feature/How-to-tame-the-new-IT-beast-called-DevOps

March, S. T., & Smith, G. F. (1995). Design and natural science research on information technology. *Decision Support Systems*, *15*(4), 251–266. doi:10.1016/0167-9236(94)00041-2

Maukonen, H. (2015). Ketterän ohjelmistokehityksen kypsyysmallien vertailu. Pro Gradu, Jyväskylän yliopisto

Mundra, A., Misra, S., & Dhawale, C. a. (2013). Practical scrum-scrum team: Way to produce successful and quality software. *Proceedings of the 2013 13th International Conference on Computational Science and Its Applications, ICCSA 2013*, 119–123. doi:10.1109/ICCSA.2013.25

Peffers, K., Tuunanen, T., Gengler, C. E., Rossi, M., Hui, W., Virtanen, V., & Bragge, J. (2006). The Design Science Research Process: A Model for Producing and Presenting Information Systems Research. *The Proceedings of Design Research in Information Systems and Technology DESRIST'06*, *24*, 83–106. Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:The+Design+Science+Research+Process:+A+Model+for+Producing+and+presenting+Information+Systems+Research#0

Pesonen, K. (2012). Agile software testing and Scrum. Pro Gradu, Jyväskylän yliopisto

Poppendieck, M., & Cusumano, M. a. (2012). Lean software development: A tutorial. *IEEE Software*, *29*, 26–32. doi:10.1109/MS.2012.107

Pratt A. (2012). Interactive design: an introduction to the theory and application of user-centered design. Rockport Publishers, 2012.

SAP. (2009). SAP User-Centered Design. Retrieved 25.2.2015 from http://www.sapdesignguild.org/resources/ucd_process.asp

Schwaber, K., & Sutherland, J. (2011). The scrum guide. *Scrum. Org, October*, *2*, 17. doi:10.1053/j.jrn.2009.08.012

Scrum Alliance. (2014). Core Scrum. Retrieved 15.2.2015 from https://www.scrumalliance.org/why-scrum/core-scrum-values-roles

Six Sigma. (2015). Yleistä Leanista. Retrieved 14.2.2015 from http://www.sixsigma.fi/fi/lean/yleinen/

Spinellis, D. (2012). Don't install software by hand. *IEEE Software*, *29*(4), 86–87. doi:10.1109/MS.2012.85

Sutherland, J., & Van Solingen R. (2011). The Power of Scrum. CreateSpace, USA.

Swartout, P. (2012). Continuous delivery and DevOps: A quickstart guide. Packt Publishing, Birmingham.

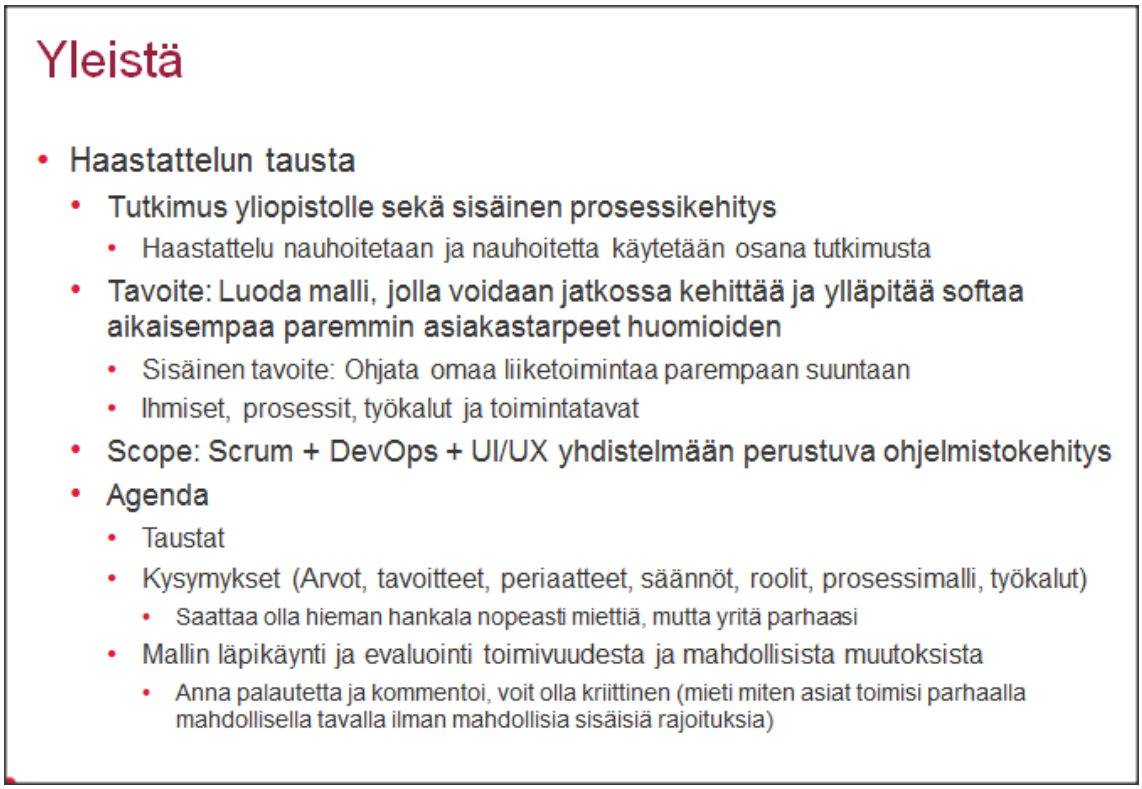
Säde, S. (2004). Cardboard mock-ups and conversations : studies on user-centered product design. Helsinki : University of Art and Design Helsinki, 2001

Tikkanen, K. (2014). Ketterän ja perinteisen ohjelmistokehityksen yhdistämiseen liittyviä haasteita ja ratkaisuja : tapaustutkimus. Pro Gradu, Jyväskylän yliopisto

Vehvilainen, R. (2000). What is preventive software maintenance? *Software Maintenance, 2000. Proceedings. International Conference on*, 18–19. doi:10.1109/ICSM.2000.882971

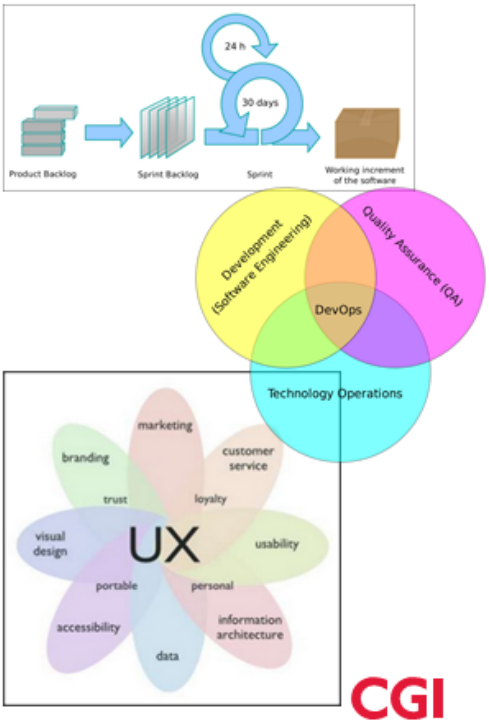
Veneziano, V., Mahmud, I., Khatun, A., & Peng, W. W. (2014). Usability Analysis of ERP Software : Education and Experience of Users ' as Moderators, (1001).

ATTACHMENT 1: INTERVIEW MATERIAL



Käsiteltävien aihepiirien kertaus

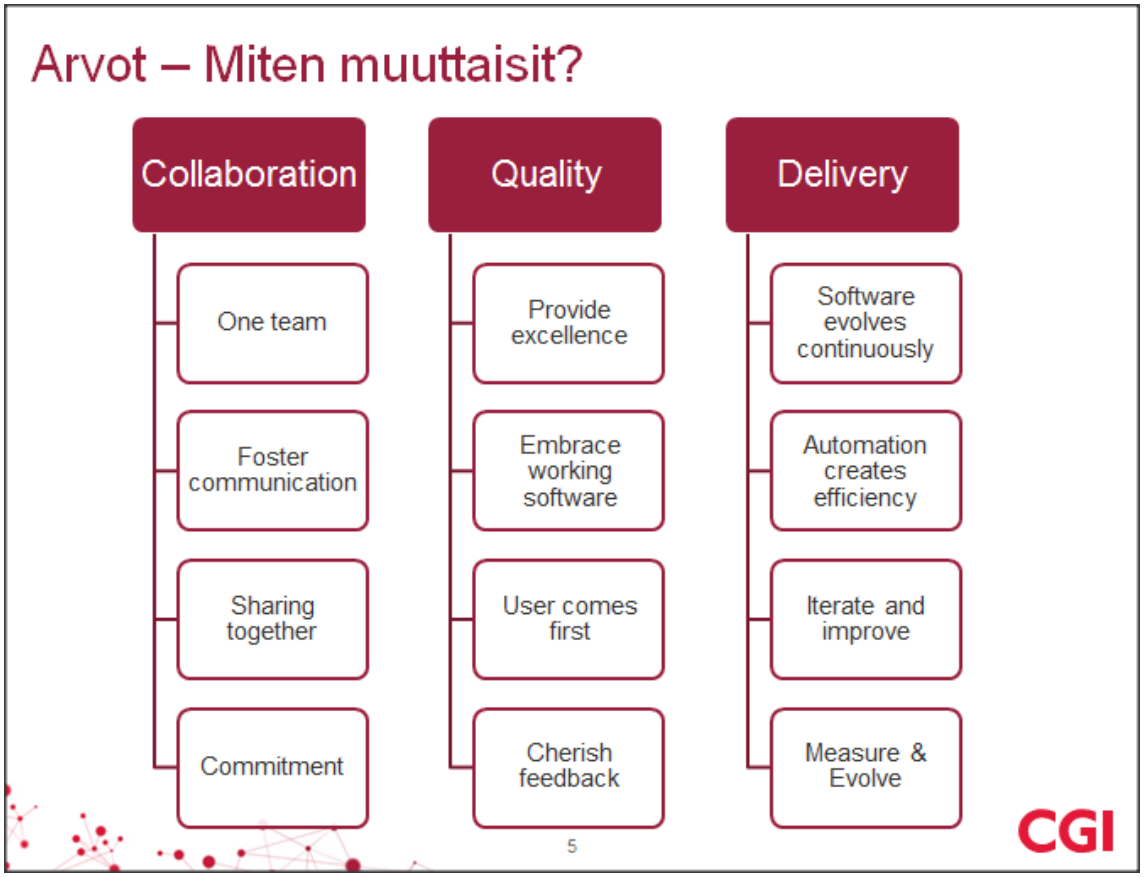
- Scrum
  - Iteratiivinen malli kehittää ohjelmistoja
  - Artifaktit: Product backlog, burndown chart, sprint backlog
- DevOps
  - Ohjelmistokehityksen nopeuttaminen (Software development)
  - Turvallisuuden lisääminen (IT operations)
  - Ihmiset, prosessit, työkalut
- UI/UX
  - Käytettävyys ja käyttökokemus
  - UI design, visual design, sovelluksen interaktiot, informaatio arkkitehtuuri, prototyyppaus

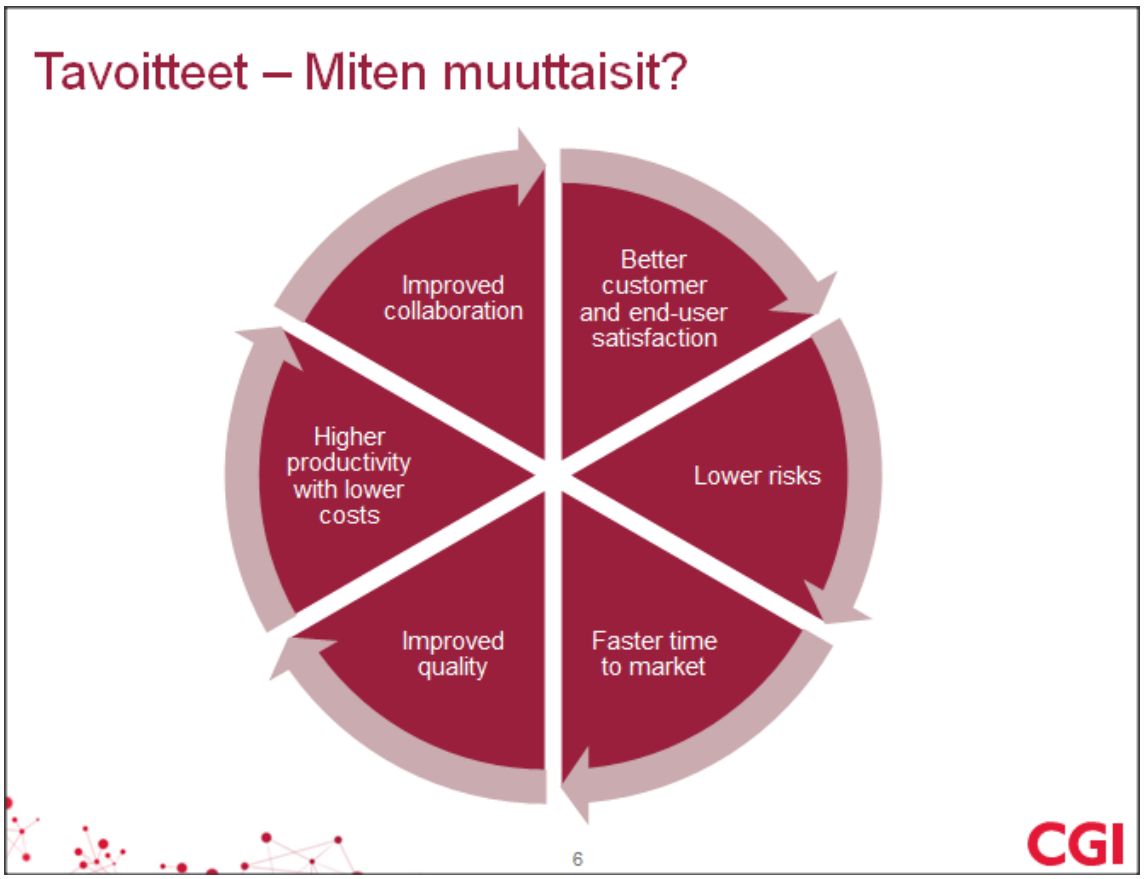


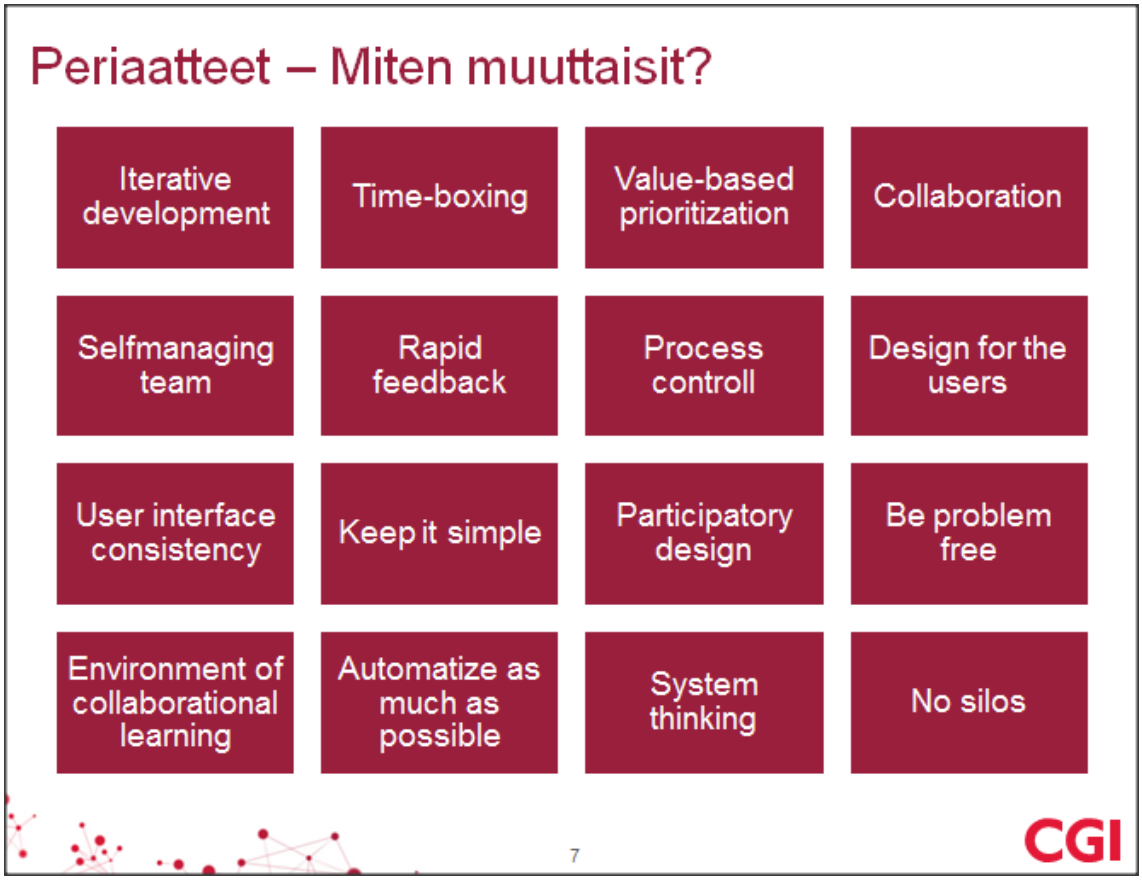
CGI

Kysymykset

- Luettele mielestäsi tärkeimmät arvot, jotka tulisi olla yhteisen tekemisen taustalla
- Luettele tavoitteet, jotka ovat mielestäsi tärkeimmät tässä viitekehyksessä
- Mitkä periaatteet tulisi ohjata tekemistä?
- Mitä sääntöjä tiimillä tulisi olla?
- Mitä rooleja mielestäsi tähän kokonaisuuteen kuuluu?
- Luettele oleellisimmat työkalut

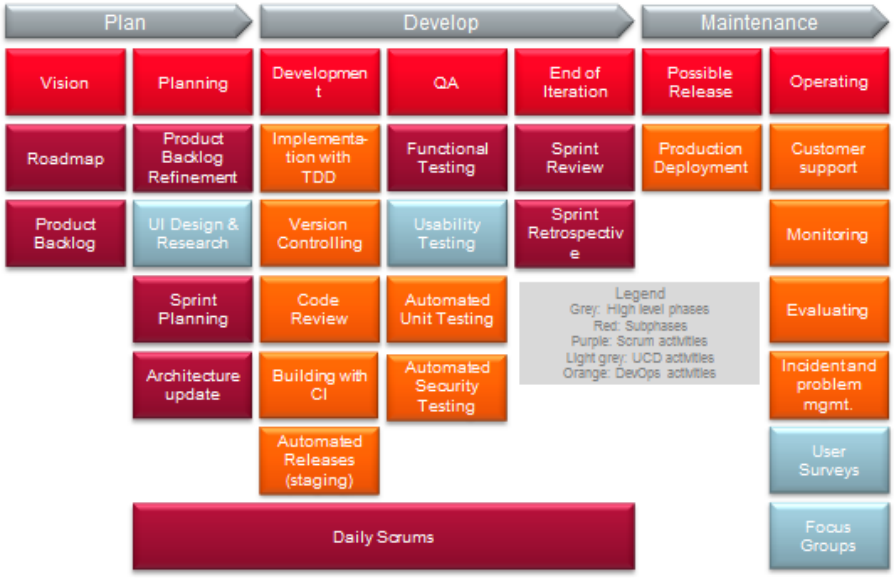

Arvot – Miten muuttaisit?
Collaboration
One team
Foster communication
Sharing together
Commitment
Quality
Provide excellence
Embrace working software
User comes first
Cherish feedback
Delivery
Software evolves continuously
Automation creates efficiency
Iterate and improve
Measure & Evolve
5
CGI


Tavoitteet – Miten muuttaisit?
Improved collaboration
Better customer and end-user satisfaction
Higher productivity with lower costs
Lower risks
Improved quality
Faster time to market
6
CGI


Periaatteet – Miten muuttaisit?
Iterative development
Time-boxing
Value-based prioritization
Collaboration
Selfmanaging team
Rapid feedback
Process controll
Design for the users
User interface consistency
Keep it simple
Participatory design
Be problem free
Environment of collaborational learning
Automatize as much as possible
System thinking
No silos
7
CGI


Säännöt – Miten muuttaisit?
Use epics in the Scrum
Keep backlog prioritized
Make sure that you have backlog ready for incoming sprints
Slice requirement to small enough
Gather known defects
Refine in every sprint before planning
Keep product backlog visible for all
Be truthfull, do not gather technical dept
Know your product
Understand the architecture
Help colleague
Share obstacles
Be willing to learn
Live by values
Be proactive
Identify waste and try to reduce it
Ask how automation can help
Follow and improve DoD and DoR
Follow together agreed rules
Identify points where things can go wrong
8
CGI

Prosessi – Miten muuttaisit?



CGI

Roolit – Miten muuttaisit?

Business Owner

Responsibilities
- Business responsibility
- Targets and objectives
- Business authority
- Financial responsibility
- Resourcing

Product Owner

Responsibilities
- Product responsibility
- Product vision and strategy
- Product Backlog
- Prioritization
- Release scheduling
- Product profitability
- Scrum decision authority

Scrum Master

Responsibilities
- Scrum event facilitating
- Monitoring the Scrum
- Shielding the team
- Removing obstacles
- Collaboration with Product Owner (technical understanding)

Service Manager

Responsibilities
- Collaboration with customer main users or focus groups
- Leading the service maintenance
- Collaboration with Scrum Master and Product Owner
- Service authority
- SLA

Architect/Tech Lead

Responsibilities
- Creates and improves the architecture
- Understands the stucture of whole product
- Creates architecture plans and maintain those
- Understands the ecosystem

Developer

Responsibilities
- Delivery responsibility
- Deliver working code within agreed metrics
- Follows the team rules and guidelines
- Collaborates with team

Tester

Responsibilities
- Testing responsibility
- Taking care of different testing phases which are not necessary automized
- Support with test automation and testing tools

UI/UX Specialist

Responsibilities
- UI vision
- UI/UX design & research
- Research SUS and customer feedback
- Creates and maintains UI/UX guidelines for the Scrum Team

Operative Specialist

Responsibilities
- Running and operating Cloud/ environment and maintaining
- Running and operating server side tools
- SysAdmin

Support Specialist

Responsibilities
- Supports customers and users with products
- Follows agreed service management processes

CGI