New agile process errors in software development

Petri Koivisto

Thesis

TIKO

2010

HAAGA-HELIA
University of Applied Sciences

Abstract

25.10.2010

The Degree Programme in Business Information Technology

Authors Petri Koivisto	Group TIKO08KI
The title of thesis New agile process errors in software development	Number of pages and appendices 57+1
Supervisors Anne Benson	

The purpose of this thesis was to investigate what kind of new errors appear in researched agile process compared to other agile processes in software development. Nowadays, it is a common trend to develop in an agile way, but it is essential to understand what agility really means; especially, it is important to understand what kind of errors appear in agile processes.

The research was carried out as follows: the theory part was written based on available online resources and books about agile development and common errors found in agile processes. The empirical part was conducted using a questionnaire and an interview regarding selected agile development process. The goal was to find errors from it. The results were compared to previously found errors to find out whether there were new errors.

The research revealed ten errors of which four were new errors. New errors were related to the following areas: Requirement management tool was not used properly. In addition, there was a common misunderstanding when the release starts. Moreover, there are misunderstandings regarding which requirements should be part of the release. In all of these areas, there had previously been errors.

The thesis concludes that all of these errors are visible due bad management of the process. Furthermore, this causes lack of communication in the process and it leads to the situation that everyone does not know how the process works or what their role in the process is. Thus, this leads to the situation that even though the release always delivers what it promises, 57 percent of the people involved in the process are not happy with it.

According to the research, the processes should be properly managed to get all the errors fixed. The manager of the process needs to be constantly ready to adopt changes to the process if they are needed to make the process better. Communication is the first customer satisfaction measurement which needs to be in place as the errors appear through communication.

A follow up research would be needed in the future to see if the same errors still exist and what areas in the process need more optimizing.

Key words
Error, agile, process, software development

HAAGA-HELIA
University of Applied Sciences

Tiivistelmä

25.10.2010

Tietojenkäsittelyn koulutusohjelma

Tekijät Petri Koivisto	Ryhmä TIKO08KI
Opinnäytetyön nimi Uudet ketterän prosessin virheet ohjelmistokehityksessä	Sivu- ja liitesivumäärä 57+1
Ohjaajat Anne Benson	

Tämän opinnäytetyön tarkoituksena oli selvittää, millaisia uusia virheitä esiintyy tutkitussa ohjelmistokehityksen ketterässä prosessissa muihin ketteriin prosesseihin verrattuna. Nykyään on yleinen suuntaus kehittää ketterällä tavalla, mutta on todella tärkeää ymmärtää, mikä on ketterää ja mikä ei. Erityisesti on tärkeää ymmärtää mitä virheitä esiintyy ketterissä prosesseissa.

Tutkimus toteutettiin seuraavasti. Teoriaosa perustuu yleisesti saatavilla olevaan materiaaliin sekä kirjoihin ketterästä kehityksestä ja yleisimmistä ketterästä kehityksestä löytyvistä virheistä. Tutkimusosa suoritettiin kyselyllä ja haastattelulla tutkitusta ketterän kehityksen prosessista. Tarkoituksena oli löytää siitä uusia virheitä. Tuloksia verrattiin aikaisemmin löydettyihin virheisiin ja tutkittiin löytyikö uusia virheitä.

Tutkimus osoitti, että tutkitusta prosessista löytyi 10 virhettä, joista 4 oli uusia virheitä. Uudet virheet löytyivät seuraavilta alueilta: Vaatimusten käsittelyn työkalua ei käytetä oikein. Yleisesti ei tiedetä milloin release (julkaisu) alkaa. Lisäksi yleisesti ei tiedetä mitä vaatimuksia sisällytetään julkaisuihin. Kaikilta edellä mainituilta alueilta oli aikaisemmin löytynyt virheitä.

Tutkimuksesta voidaan päätellä, että kaikki nämä virheet esiintyvät huonon prosessin hallinnan takia. Tämän takia prosessin kommunikaatio ei toimi ja se johtaa tilanteeseen, jossa kaikki eivät tiedä miten prosessi toimii tai mikä heidän roolinsa on prosessissa. Tämä taas johtaa siihen, että vaikka julkaisu tuottaa aina sen minkä lupaa, 57 prosenttia prosessissa mukana olevista eivät ole tyytyväisiä prosessiin.

Suosituksena kaikkien virheiden korjaamiseksi on prosessin kunnollinen hallinnoiminen. Prosessin hallinnoijan tulee olla kokoajan valmiina omaksumaan prosessiin muutoksia, jos ne ovat tarpeellisia prosessin parantamiseksi. Kommunikaatio on ensimmäinen asiakastyytyväisyyden mittari. Mittarin pitää olla paikallaan, koska virheet ilmenevät kommunikaation kautta.

Tulevaisuudessa olisi tarpeellista suorittaa jatkotutkimus, josta ilmenisi ovatko samat virheet yhä esillä ja mitkä prosessin osat tarvitsevat lisää optimointia.

Asiasanat
Virhe, ketterä, prosessi, ohjelmistokehitys

Table of contents

1 Introduction

This section specifies what the subject matter of the thesis is and what the research problem is. In addition, it specifies how the research was conducted. This section also specifies the structure of the thesis.

1.1 Subject matter

Subject of the thesis is new agile process errors in software development. Research target is to find out, are there any new errors in defined agile process compared to previously found errors in other agile processes. I selected this subject for my research, as it is quite trendy to do software development with agile methods. To back up this statement, Abrahamsson stated in an article published in 2007 that agile gives you a competitive advantage (Tietoviikko, 2007). Now days, to get this competitive advantage, in 2010 research by Forrester Research, 45% of the answerers mentioned agile as their way of doing software development (Taft, 2010).

Graig Larman (2007) has introduced common errors found from agile development and its processes in his book: Agile & Iterative development. These errors are introduced in the theory part of the thesis. Otherwise, most of the researches found about errors in agile processes, for example (Stankovic 2010; Väyrynen 2009), are related to testing, which reveal errors found from code. According to Larman (2007), and IBM developer library (2010), common existing errors, which have been found, are related to scope, communication, agile development approach, methodology, requirements gathering and documentation. As well there are common errors in management of the process and tools used in the process.

1.2 Research problem

Researched phenomenon is creeping into agile processes, as there is a common feeling that everything is doing just fine in the development process. The key reason for this is that there is no constant feedback available from the customer or the feedback is not taken seriously at all (Wells 2009). For example, there might start appearing occasional things like schedule problems, end result of the work is not what it is supposed to be or there are more and more development work needed in iterations to accomplish the work needed, even the iteration

scope has not been increased. First these errors are taken as deviations, and in a longer time period they are adapted as a way how the process works.

Reason why it is important to research this topic is to see, if the agile development has been done the right way in the researched process. Even more important is to see, what possible errors in this researched process might exist? This is really important, as the researched process has evolved from waterfall model towards agile practices during a certain time frame. At the moment, it has nuances to be called an agile waterfall, as it has the elements from agile models and original waterfall model. Even tough the research concentrates on new errors found from the researched process, it is absolutely important to find out all existing errors in the development process and provide a systematic approach to fix them.

Main outcome of the research is to bring out new errors, which have previously just been badly managed things inside the process. Everyone had thought that these things should be fixed, but no-one has not categorized these items as errors. Reason for this is that constant feedback has not existed or it has not been taken seriously into this process, so the errors could have been properly fixed by the manager of the process. On the other hand, these results could be easily utilized in any agile development process to be presented as lessons learned, as an easy way to prevent errors which might appear sooner or later.

Research problem focuses on following areas: Schedule, requirements, tool used for requirements, roles, business ownership, communication, meetings, and management of the process and overall satisfaction of the process. All of the areas had previously found errors. Errors found from the projects were excluded from the research.

1.3 Research method

Research was done through a questionnaire, sent to persons working with the researched process, and through an interview with the business owner of the process. Research method was chosen, as it was an easy way to get answers from all participants in the process. This was possible, as the sample was small. Based on the answers, it was possible to get as accurate data as possible for the research.

Questionnaire (Appendix 1) consisted of 12 questions and 3 possible answers: Yes, no and can’t say. There was a request to provide a short comment, if the answer was no or can’t say.

There was also a possibility to provide comments as overall feedback. All no and can't say answers were considered to be errors.

Interview and questionnaire results were combined together and presented in their own tables. Results are grouped to relevant sections where they belong. These groups include the process itself, requirements, releases and management of the process. Questionnaire and interview goal was to get as deep insight as possible about the researched process. Results are presented directly without commenting. Comments are included in conclusion.

There are two criteria for research to be successful. First is to find out, if there are new errors found in the defined process compared to the ones previously found from other agile processes. Second, to provide recommendations how to fix all found errors. All the found errors and recommendations how to fix them are available in conclusion.

1.4 Structure of the report

Theory part of the thesis builds up the foundation for this research. It introduces basics of software development and software engineering. Moreover, it specifies how agile software development broke loose from waterfall model through iterative development and what are the different ways of doing agile software development.

In addition, there is an introduction what is a process and what value does it bring, when you are using processes. Furthermore there is a specification what an agile process is and how agile process works. As well there is a definition what roles need to be in place to make process successful and what rules needs to be followed it to be agile.

As the thesis concentrates on errors, it gives an overview what an error is and how error is defined. Base on the error definitions, thesis introduces what agile development process errors have been previously found, and explain why they are errors.

Research part of the thesis introduces the defined process which is in scope of the research. Base on this, thesis gives a step by step introduction how the defined process works, and what are the rules and roles of the process. Moreover, research specifies the environment variables where the process is performed.

2 Software development

Software development is described easiest with software development methodology or system development life cycle. “A software development life cycle model is either a descriptive or prescriptive characterization of how software is or should be developed” (Scacchi 2001, 1-3.)

Origin of the software development states back to 1950: s. In the 1960: s originated the development life cycle term, waterfall model. There are steps in the software development life cycle, which has their own techniques to mitigate risk and produce quality deliverables (Hypethot 2010.)

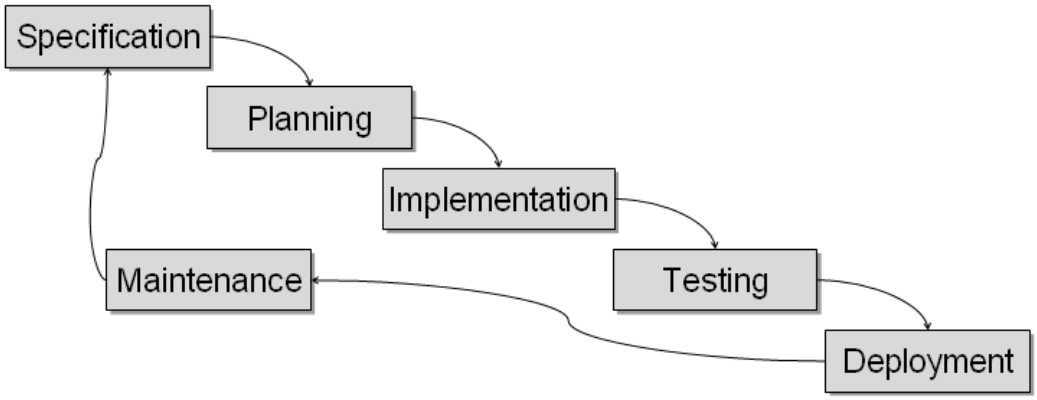

figure 1. Software development picture (Puurunen 2010)

Development lifecycle has the following areas: Specification, planning, implementation, testing, deployment and maintenance (*figure 1*). It is important to understand, how software development activities should be performed and in which order (Scacchi 2001, 3.) This is the role of the process owner or project manager to see that the process flows smoothly from one area to another (Rossberg 2008, 24).

Requirement analysis and specification are where the scope, use cases and technical guidelines are defined (Puurunen 2010). This is the area, where the subject matter expert role is most crucial. That role defines what the software needs to do (Bogue 2005.)

Planning of the system is where the information model, usability and technical architecture are described (Puurunen 2010). These area roles include developer, who transfers the need to

something concrete; architect who defines the overall picture of the system and subject matter expert who approves the proposed solution definition (Bogue 2005; Rossberg 2008, 23-26.)

Implementation, is where the requirement specification is coded to user interface and database and technical architecture is implemented (Puurunen 2010.) These area roles include developer, who codes the defined concrete to actual system and architect who oversees that code development is done according the overall picture limits (Bogue 2005; Rossberg 2008, 23-26.)

Integration and testing includes where the implementation work is tested and documentation is created, to make sure that performance of the system is valid against the requirements specified (Puurunen 2010.) These area roles include developer, who need to make sure that actual coded system is integrated to existing system and architect who verifies that defined overall picture has been achieved (Bogue 2005; Rossberg 2008, 23-26.)

Deployment of the system is where the user acceptance is provided to take the system into use. Users are also trained to effectively use the system (Puurunen 2010.) These area roles include the subject matter expert, who tests and approves the developed solution and creates the training material needed for the users and developer, who makes the necessary changes needed based on user acceptance test results (Bogue 2005; Rossberg 2008 23-26.)

Maintenance is a transition period. It has its own steps to make sure that the software development process is working. Corrective maintenance takes care of the diagnostics of bug identification and fixing. Remedial maintenance takes care of the small changing needs of the customer. Predictive maintenance takes care of the software development itself. Requirement management act as a centralized way of collecting upcoming enhancement ideas from different parties how to develop the system (Puurunen 2010.) These roles include operations and maintenance people, who take care of the system until it retires. Roles include as well the developer, who enhances the system, based on change requests raised by the subject area experts (Rossberg 2008, 25.)

In short, software development can be summarized as a way of enabling creation or producing software. This is described more detailed in software engineering.

2.1 Software engineering

"Software engineering is a systematic and disciplined approach to developing software. It applies both computer science and engineering principles and practices to the creation, operation, and maintenance of software systems" (University of Waterloo 2001.) The term software engineering was first time mentioned in a European conference in 1968 by Fritz Bauer (Wang & King 2000, 6).

Software engineering is divided to 10 sub-areas. 1st sub-area is software requirements. Those can be defined as easiest, to be a property which must be presented in order to solve some problem in the real world. Software requirements are divided still to product and process requirements, functional and non-functional requirements, and system and software requirements.

Product requirement is a requirement, which is developed in software. Process requirement tells how that shall be done. Functional requirement describes what the software should do. Non-functional requirement specifies the quality how software should act. System and software requirements describe system as a whole. Software requirements are not just requirements. Requirements need to be analyzed, specified, and validated for software.

2nd sub-area is software design. In general, design is a form of problem solving. Software design is done with Case tools and unified modeling language. Case tool can as simplest be a use case, where is defined what happens and when (Swebok 2004.) Unified modeling language defines through diagrams, what would otherwise be too difficult to be presented through text (Uml 2010).

3rd sub-area is software construction. Construction means basically development. Development is done through coding with programming language. Programming language is communication specified by human as a solution to a computer (Swebok 2004.)

4th sub-area is testing. Testing is a process performed by developers, where they try to find errors from the developed product, and ensure quality by doing so (Testing Brain 2010). There are different kinds of testing, which can be performed. Three (3) main test areas are unit, integration and system testing. Unit testing verifies that the developed part of the software is functioning as specified. Integration testing verifies the functionality of separate soft-

wares which are integrated together. System testing verifies the functionality of the whole developed system (Swebok 2004.)

5th sub-area is the software maintenance. Software maintenance relates to changes done after software delivery. Maintenance itself means the idea to keep the developed state, and keep the failures away from the software. Software maintenance is a process to modify the developed system to meet changing needs, fix errors and improve the system performance when needed. Software maintenance is not only about software. It includes the documentation, requirement analysis, specification and design (Grubb & Takang 2003, 5-7.)

6th sub-area is software configuration management. Software configuration management defines the lifecycle of the developed system. It identifies the areas, and manages and controls the change. It verifies records and reports the configuration information activities. It includes as well process management, so it could be called as well process configuration management (Keyes 2004, 1-4.)

7th sub-area is software engineering management. Software engineering management defines planning, coordination, measuring, monitoring, controlling and reporting activities of developed software and its maintenance. Main part of it can be defined, to be like project management of software engineering. There are also few more aspects, which should be included in this area: Personnel management and communication management. Personnel management takes into account, how to motivate people working in the process, and how they should be trained to be able to achieve the long-term goals for software engineering process. Communication needs to be in the scope as well, as it measures the performance of the people working in the process, and if they understand the end-user needs, and their complex requirements (Swebok 2004.)

8th sub-area is software engineering process. Software engineering process should not be stated to be a lifecycle model. Software engineering process consists from practices, which cover all aspects from development and management point of view. In short, this can be defined so that it includes all the previous sub-areas under its umbrella. Software engineering process needs to be established so that it is stabile and re-usable (Wang & King 2000, 5.)

9th sub-area is software engineering tools and methods. Software engineering tools can be defined as tools, which are created to assist the software lifecycle processes. In short, this in-

cludes all the tools used in other sub-areas. Software engineering methods are defined to three areas: Heuristics, formal and prototyping. Heuristic methods concentrate on informal approaches. Formal methods are based on mathematical approaches. Prototyping methods use the software engineering approaches from various forms of prototyping. These three topics don't overrule each other, but concentrate to present distinct concerns (Swebok 2004.)

10th sub-area is software quality. Main point of the software quality is, to define that desired quality is built inside the developed software. This involves that all the previously defined sub-areas are taken care of in the software engineering process. Characteristics of the quality should include that it is functional, reliable, usable, efficient, maintainable and portable. Basically this means that developed software needs to have required functions available in reliable way. Software needs to be easy to use, and it is efficient to the user. Software needs to be easy to modify and maintain, and be transportable to different environments. Those above mentioned characteristics reveal, if the developed software is high quality or not (O'Regan 2002, 1-6.)

3 Iterative development

Iterative development approach is to start doing software development in cycles, rather than trying to do everything all at once (Wisegeek 2010). "Iterative Development adds agility to the development process" (Wells 1999). In iterative development, you schedule your development into x number of iterations, which are 1-4 weeks long. X (*figure 2*) in this sense is an unknown factor, as you don't exactly know how many iterations you are going to need in development to finish it.

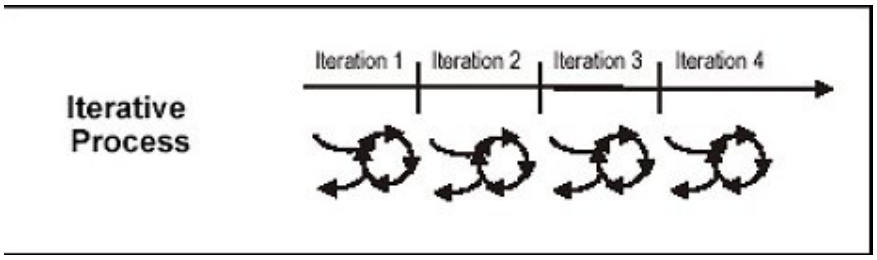

figure 2. Picture of Iterative development process (IBM, 2010)

Advice is to do one week iterations, even they seem very short. Iteration is the core of the development process, so iteration length should be always constant or equal. This makes managing, planning and measuring of the process easy and reliable.

First rule in iterative development is that, you don't schedule your iterations in advance. Iterations are planned in the beginning of every iteration in a planning meeting, where is seen what can be done during next iteration. This way it is easy to manage changing user requirements. This way there shouldn't be any surprises that you are not aware of. "It is also against the rules to look ahead, and try to implement anything that it is not scheduled for that iteration."

Those extra requirements will be implemented in the release plan, when it becomes important enough or most important story in release plan in the product backlog. It is crucial that the iteration deadlines are written to the stone and those are not changed no matter what.

Easiest way to follow the progress of the iterative process, is to do it during iterations. The requirements should be developed in prioritized order chosen by the customer. If it seems that all the requirements scoped into iteration are not finished during it, they need to be replanned and estimated and possibly removed from the iteration (Wells 1999.)

Iterative development may seem like a chaos or spontaneous to outsider, but it is not. Iterative development has strict rules it needs to follow, and it is a well organized way for fast and efficient development (Wisegeek 2010.)

4 Agile software development

Agile indicates matters, which have caused problems in software development. Past 50 years the development methods of the waterfall model have always brought out the same problem. Requirement specification has always been done the wrong way. At the best, requirements can show what the end users want. Generally, requirements have been done based on the demands of the end users, who can do absolutely nothing with the done product. What is wanted or what is needed, are two different matters. It is indeed worth examining, what the end users need and not only listen what the end users want to have.

Between the publication of the end product and specification of the requirement, the demand of the need might have changed. And this has caused even more problems between the needs and the requirements. This however, causes the fact that the biggest development work has been done to the wrong area, before it has been noticed, what should have really been done.

In the late 1990: s, several new software development methods began to get attention in the field of the software technology. These agile methods were combinations from old, new and improved old methods. (Nodder & Nielsen 2009, 4-6.) Agile Software Development is now considered to be a concept, a philosophy and a methodology. Agile software development promotes an iterative approach to software development using short and light development cycles, where the requirements or backlog items are broken down as manageable items. This promotes as well, different deliverables as outcomes in the development (Gatherspace 2010.)

4.1 Agile manifesto

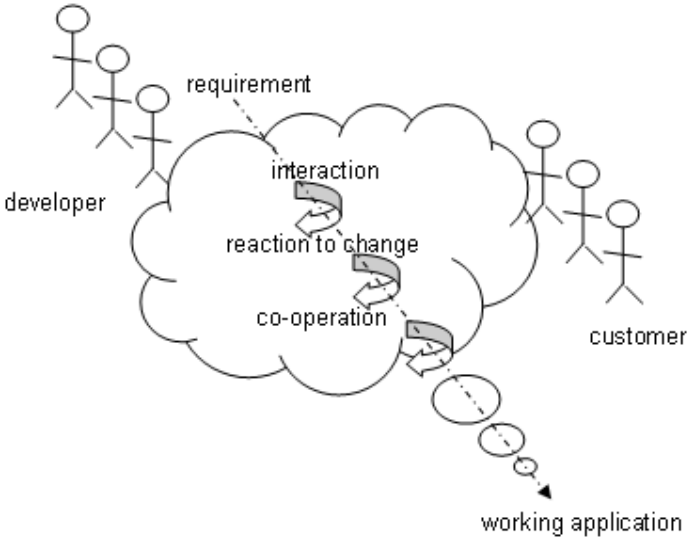

figure 3. Picture of agile manifesto (Petri Koivisto based on agile manifesto definition 2010)

Agile development is constant change, where the most important parts of the software development have been divided to four values, known as the agile manifesto (*figure 3*). 1. Individuals and interaction are valued more than processes and tools. 2. Working application is valued more than overall documentation. 3. Customer co-operation is valued more than contract negotiations. 4. Reaction to change is more valued than following the plan.

According to Jim Highsmith, agile manifesto was established on February 2001 on a ski lodge in Utah, where 17 people needed an alternative way to documentation driven software development. There was also a doubt among the participants, can they ever agree on anything substantive. This agreement was the start of the Agile Alliance, whose aim is to assist software development methodologies in new, more agile ways (Agile Manifesto, 2001).

4.2 What is agile development?

Agile methods are collections of best practices, which fit into certain types of projects and to certain types of organizations (Nykänen 2009, 3.) You need to also understand that there is no

penalty in agile, there is just feedback. Agile is about learning, and if you find something to improve, you must improve it.

To become really successful, you must build a circle of trust with the product owner and all the stakeholders. You must convince them that this is the way to work and you benefit from this approach. This is achieved by commitment, co-ordination and excellent results. If your releases are high quality, and you release what you promise every month, that is not a problem (Coplien 2009.)

Agile development can be divided to different kinds of methods, which have their own best practices. It is important to remember that none of these methods may not be taken directly in use as is. The best development process for the needed purpose may be a combination of few or all of these. Here is the introduction to most common ones, which are in use today.

4.2.1 Scrum

When you are thinking about agile processes, the first thing that usually pops up is Scrum. Agile is not only scrum even it has been said that scrum has won the agile war (Coplien 2009). “Scrum is a rugby offensive term, where a team pushes forward to score as a single unit” (Gatherspace 2010). Scrum is thought to be more as a communication method than development method, which is based on following rules. 1. Customer priorities the list of user stories (requirements) they want. 2. Scrum development is broken down to 1-4 week sprints, which are equal to iterations. 3. Scrum team is a multi-functional team, which has all skills needed to complete the development.

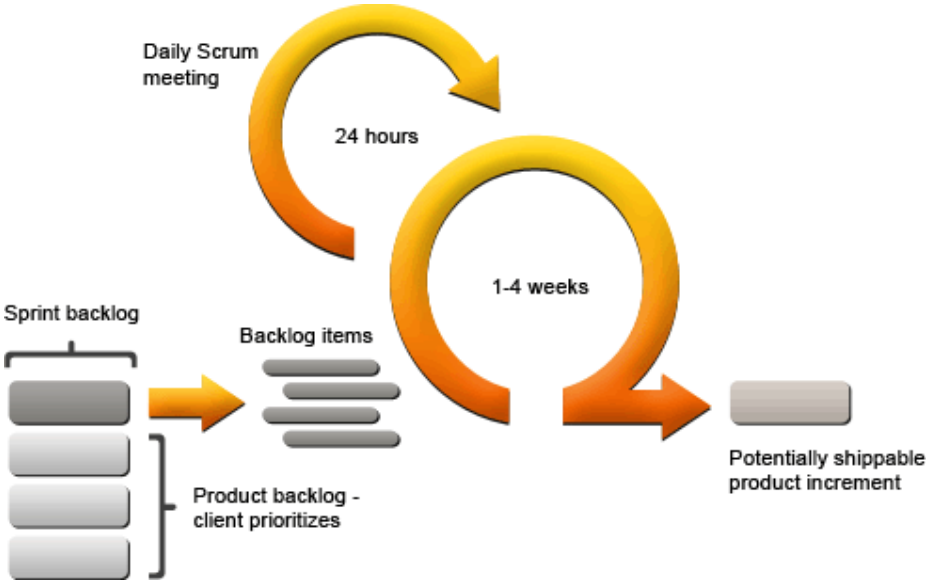

figure 4. Picture of scrum process (Reaktor 2010)

Scrum development process (*figure 4*) starts with a product backlog. Top items of the backlog are then moved to sprint backlog. Scrum team is together responsible that all the user stories chosen to the sprint are developed, integrated, tested and documented during sprint.

In daily scrum meetings, each member of the team describes what he/she has done, and what the goal is for the next period. He/she also estimates how much time is still needed to complete the development from the undeveloped user stories point of view. Outcome of the sprint is a potential done product (Reaktor 2010.) Anyone outside the scrum team can participate to these meetings, but only persons who can speak are the scrum team members (Larman 2007, 119).

User stories can be deleted or added at any time during the scrum development process. The decision about, when to actually release any functionality or deliverable is done by product owner. The team evaluates at the end of every sprint, ways to improve scrum process in future sprints, as well as how to improve the end product (Scrum Alliance 2010.)

Daily communication backs up the fact that scrum is a communication process. This way it brings up possible problems, which can immediately be taken care of (Reaktor 2010.) Of course scrum to be successful, it needs to be adapted to the organization and organization needs, so the organization can change the way of thinking in development processes. The most crucial role in scrum to be successful is the role of the Scrum Master. Scrum Master

needs to reinforce the development and sprint goals so that development team stays always in focus (Larman 2007, 132-134.)

4.2.2 Extreme programming

Extreme programming can be explained in one sentence as follow. “Instead of delivering everything you could possibly want on some date far in the future, extreme programming process delivers the software you need as you need it” (Wells 2009). Extreme programming (XP) is actually a variation of iterative development, which is used successfully in smaller companies. Extreme programming stresses customer satisfaction, and always aims to deliver the software which is developed on time (*figure 5*).

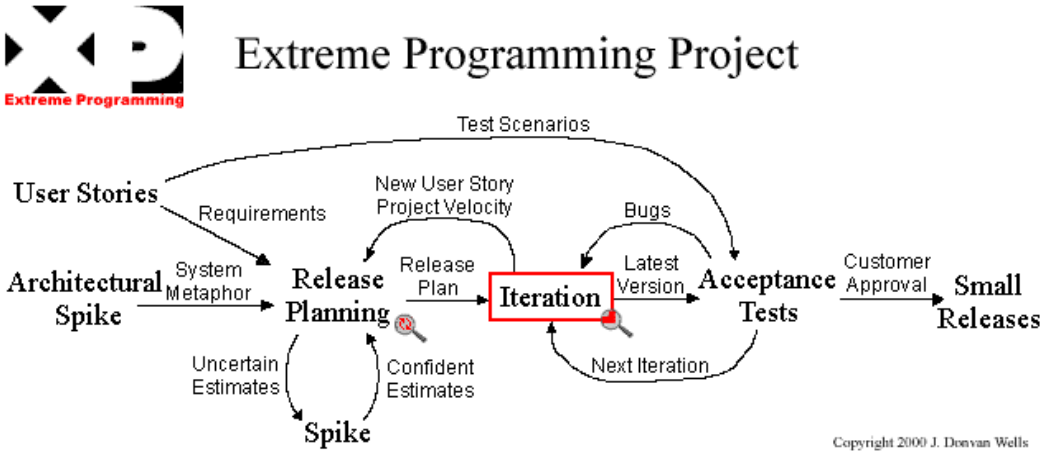

figure 5. Extreme programming picture (Wells 1999)

All begins with user stories, where customer defines the requirement. It is brought to release planning for estimation and against the priority; it is taken into iteration for development. Before the end result is deployed, it needs acceptance from the customer. Process starts again from the beginning and everything happens in small releases.

Extreme programming development process uses five core ways: communication, simplicity, feedback, respect, and courage to be successful. Everything in the process needs to be simple and clean. Key to success is constant communication between customers and developers, and that ensures constant feedback from the first development day onwards.

Other important thing is that extreme programming is created to accept changing business requirement even last minute in the development lifecycle. In extreme programming, there is always a team which focuses on team work. Team includes manager of the process, developers and as the important part the customer, who aims together to provide quality deliverables in the development process (Wells 2009.)

4.2.3 Agile Rational Unified Process framework

Original idea for rational unified process was to unify modeling languages for software development. As rational unified process was introduced at the same time as agile methods, there is discussion ongoing, is it comparable to agile software development (Wisegeek 2010.)

We should take a look at some of the findings that are available. Rational unified process has adopted many of the concepts and techniques agile have been built on. You can use rational unified process agile or non-agile way. You can define yourself, which one you want to use and how agile you want to be (Ambler 2007.)

Agile rational unified process is a process framework with a collection of methods. Process framework is a set of pre-defined process concepts, where you can choose the one you want to use. First step for development in agile rational unified process is to define a development case (Wisegeek 2010.)

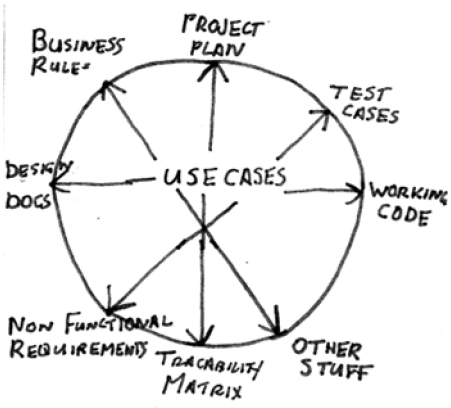

figure 6. Agile Rup picture (Agile 2010)

Agile rational unified process (*figure 6*) is based on use case driven development. This means, that development is done based on visible user requirements. It uses iterative approach, and is

centered on architecture, which is built early in the development process. But it differs from other agile processes in a way that end user requirement must be built into the final documented product completely (Buzzle 2010.)

5 Process

This section defines what a process is, and what is the value of using one. Section also defines what an agile process is, and how it is managed.

5.1 What is a process?

Process is a group of consecutive functions, which are logically related to each other. Process begins from somewhere, ends in somewhere, and is carried out often or repeated. Process has to produce surplus value, in other words has to benefit the customer. Customer can be internal or external, person or organization. Performance of the processes is always estimated from the customer's point of view (Rehn 2010.)

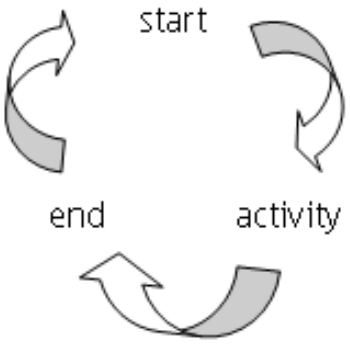

figure 7. Simplified process picture (Koivisto 2010)

If we look only a simplified view of a process, it has a start, an activity and an end. As repeated, it starts always again (*figure 7*). To understand fully what is meant with a process, we can define it as a sentence and break it down, to get deeper insight to it.

Process is a group of related consecutive functions triggered from an event, which ends up in a specific result for the customer of the process. Process success depends on specific areas of the sentence. 1. Specific result needs to be something, which can be identified and measured. 2. Customer of the process needs to be identified and receive a real benefit from the specific result. 3. Trigger is a way for an event or events to turn into a specific result in a traceable way.

4. Function can be a step, task, action or activity. 5. Group of related functions must relate to each other and completion of one function initiates the next function. Related functions must be back trackable to the initiation event. 6. Process can be defined as a main process or as a support process (Syracuse University 2008.)

Process can be small or large. Large process can be divided into smaller sub-processes. Small processes can create together a large process. Process can be renewed, defined, developed, measured and analyzed. Presentation of a process is done with a flowchart (*figure 8)* (Rehn 2010.)

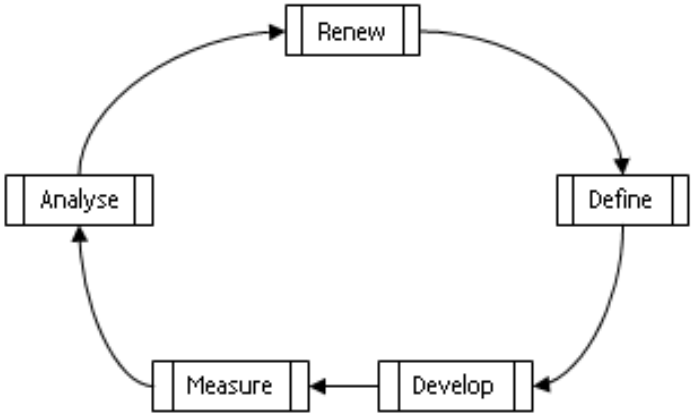

figure 8. Picture of a flowchart (Koivisto 2010)

5.2 Value of a process

What is the value of a process? Process is one of the most effective tools to problem solving there is available. The reason for this is that, if you don't know what you are doing, it is really difficult to achieve anything important. In other words, if you don't know what you should lead, how do you know how to lead it (Six sigma 2010?)

When you have a process in place, you have also following benefits. It keeps things simple, communication and feedback is constant. Simple means that everything is done with small manageable steps to achieve the end goal. Communication and feedback means that inside a process everything is worked together as a team, feedback is adapted to the process as it comes in (Wells 2009.)

5.3 Agile development process

Now as we know what agile development is and what is a process, we should combine those together as an agile development process. When agile or iterative development methods are used, there shouldn't be any process problems. You still need to remember that agile is only a framework for the software development, which does not give you any final answers how to do the development, or fully functional process what you should follow.

You need to keep in mind as well that agile is not a competition against other developers or models they use. Agile is a way how you want to develop. Agile development methods are really ways of working methods, with which a software production is intensified, and the software is developed to correspond better to customer's real needs.

The most important factor is to meet the customer's needs, by publishing continuously and early as possible new useful versions of the developed software. Agile process also approves change requests, even if the development is at the final stage, so that the change is harnessed into the customer's competitive advantage.

The agile development process is built around motivated individuals. To make the process successful, they need to have an environment and support, which they need to be able to trust that they can get the work done. In the development team, the most efficient way to transmit information is taking place through face to face discussions. Constant attention to technical quality and to good structure and planning increases agility.

Keeping it simple and having a skill to maximize the amount of work what does not need to be done is essential. Development team should also think with even intervals, how it could become more productive than before and thus adjust and edit its operation according to findings. Functioning software is the primary measurement or success criteria for this progress.

In agility, unnecessary work (litter) is reduced, problems are brought up to be repaired and processes are constantly optimized. Following these simple practices, agility provides you opportunities to be better and better. Agile process is about individuals and interaction, customer cooperation and reacting to change. Agile process includes common practices, iterations, common lexicon and learning from the earlier experiences. Risks are attempted to be mini-

mized by dividing the software development into short stages, iterations (Agile methods 2010; Coplien 2009; Larman 2007, 9; Nykänen 2009.)

5.4 Iteration

Every iteration is a small 1 - 4 weeks “project”, which includes all software development phases. Iteration should basically provide functional software, which is fit for publication. At the end of every iteration, it is estimated what has been achieved and as well decided, what is going to be done in the next iteration. Iteration is a quick time period, because it must be possible to change the direction of the development fast to react to new important requirements which may arise.

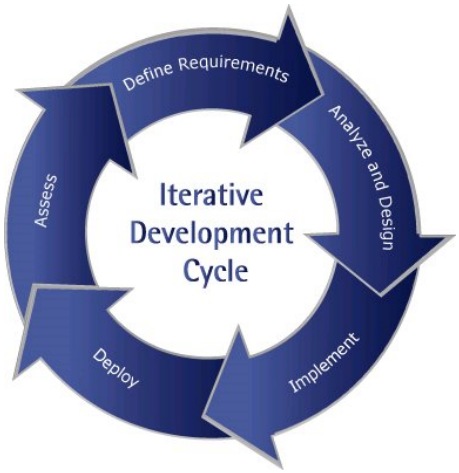

figure 9. Picture of iteration (Refresh Software 2010)

Iterative development process (*figure 9*) is a continuous cycle, which begins always again, when previous cycle ends. It includes always definition (assess) of requirements, analyzing and designing them, implementation and testing of them and deployment of those parts of requirements, which are ready to be deployed (Refresh Software 2010.)

5.5 Agile waterfall

Interesting aspect is as well that agile development method tries to get rid of waterfall development model. But if the agile development is done the right way, development consist of

series of small waterfalls with iterations and feedback (*figure 10*) (Coplien 2009; Larman 2007, 106; Nykänen 2009.)

figure 10. Agile waterfall picture (Defined Consulting, 2010)

Good reference to the term waterfall is that ”agile is a boat in a canal which goes down, when water is let out in an organized way” (Lehtovirta 2009). Other aspect to the waterfall model is that waterfall is usually based on fear. Agile waterfall model is considered to be fearless. Fear in this context means fear of risks which may appear. In agile risks are seen as opportunities (Coplien 2009.)

5.6 Agile development process roles

This part defines the roles needed for keeping the agile development process alive. Agile has several roles. Roles combine together a team (*figure 11*), which consists usually from 15-50 people depending on the size of the process used. There might be different names defined to the role. It depends which agile process is used. Process is not manageable without these agile roles.

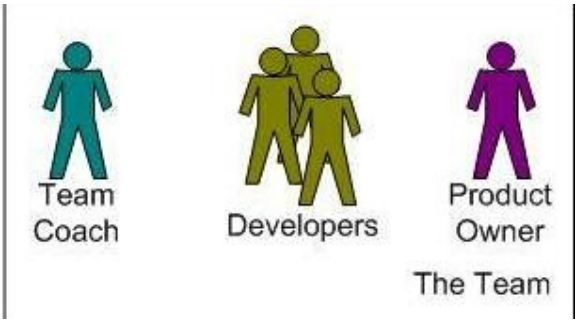

figure 11. Agile team picture (Ambler 2009)

Team coach or project lead. In scrum this role is called Scrum Master. Team coach or facilitator is responsible for facilitating the development process and development team. Facilitator also makes sure that the team doesn't face any problems during the development process. Possessor of this role needs to have soft skills of project management.

Developer is responsible for the creation of the system, as well as delivery of it. Possessor of this role needs to have technical skills. There can be many developers in a development process.

Product owner is the customer of the agile development process and represents the stakeholders. Product owner is the only person in the agile development process, who is responsible for the prioritized work item list or requirement list. This work item list is called a product backlog in Scrum. Product owner needs to provide correct info for the team, as well as make fast decisions. Stakeholder can be anyone who has a direct or indirect interest to the agile development process (Ambler 2009.)

5.7 Agile development process rules

There are some rules that apply to agile development. Rules can be divided to five relative areas: Planning, managing, designing, coding and testing. Planning rules consist of requirements, releases, release plan and iteration plan.

Requirement/user story is lines of text written by the customer describing what the system should do for him/her. Requirement should bring long term return on investment. Release is an iteration which produce tested working software frequently to be demonstrated for customer. In scrum release is called a sprint.

Release plan is a set of user stories in prioritized order. In scrum, release plan is called product backlog. Iteration plan includes the most valuable user stories selected to iteration. In scrum iteration plan is called sprint backlog (Scrum Alliance 2010; Wells 1999; Wells 2009.)

Managing rules consist of openness, visibility, communication, pace, velocity, meetings and fix. Open visibility is something what all interested parties should be able to see, what is stored in the release plan, as well in the iteration plan, and see what is released when. Open visibility

leads to open communication. Open communication makes sure that the feedback loops keep tight and everyone can say what they think (Shore & Warden 2008, 88, 115.)

Pace or release rhythm should always be equal. This means in practice that every release needs to be equal length. This enables better planning of the velocity of the release. Velocity is a simple measure how much work can be done in releases. Don't promise too much velocity in a release. Too optimistic promises gives you only trouble. Simple rule is that next release velocity is the total sum of the finished user stories during previous release, based on the estimations. The optimal velocity comes through time, when the development teams test their limits and what they have learned (Shore & Warden 2008, 264, 382; Wells 2009.)

Meetings are important part of the rules. Every day the development team should have a meeting, where is shared, what the developers have done and what they are going to do (Wells 1999). In scrum, this meeting is called daily scrum. In scrum, there are also sprint planning and sprint review meetings. Sprint planning is the meeting, where team and product owner decide the scope of the sprint. In sprint review team shows to the product owner what it has accomplished during the sprint (Scrum Alliance 2010.) Sprint review can be called as a demo as well.

Fix it means that fix the process if it gets broken. Follow the defined rules, but if something is not working, change it to something which works (Wells 1999.) Designing rules consist of simple and metaphor. Simple is keeping everything simple and lean with the four rules: It needs to be testable, understandable, browsable, and explainable. There is no point of doing it the hard way to waste time (Shore & Warden 2008, 326.) Metaphor in short, is to make things self explanatory. This refers to quality and naming, without huge documentation to explain it (Wells 2009.)

Coding rules consist of availability, standards, collective feedback and burn down chart. Availability means that the customer needs to always be available for the process. Standards means that development needs to be done according to agreed standard (Shore & Warden 2008, 58, 271.) Collective feedback gives everyone the opportunity to share new ideas. "Any developer can change any line of code to add functionality, fix bugs, improve designs or refactor. No one person becomes a bottle neck for changes" (Wells 2009.) Burndown chart gives a glance to the development, how much of it is still remaining in iteration (Scrum Alliance 2010).

Testing rules consist of unit tests and acceptance test. Unit test means that all developed components are tested automatically. Code cannot be released, if it is not tested (Shore & Warden 2008, 163, 264.) Acceptance test is done based on user stories. Acceptance test is an expected result from the user story. Approved acceptance test reflects that system is functioning as it should base on customer requirements (Wells 2009.)

5.8 Management of agile development process

Process management is a method founded during 1990:s for activity renewal and implementation or execution of change. It was done from customer's point of view, and the main point for this process was the elimination of unnecessary work and taking innovation into use (Hannus 1994.) Process management is constant management of change. Management of a process is equal to project management, where you need to consider from process point of view, people working in the process and performance of the process itself (Thom 2010.)

The role of the manager of the agile development process is to be a facilitator, who makes sure that everything happens at the right time and developers have the right information at hand. When problems appear, the facilitator will solve the problems him/herself or direct the solution of the problem to the right direction (Larman 2007, 119; Nodder & Nielsen 2009, 33.)

If the agile development process is managed actively, there shouldn't be any communication problems. This way everyone knows what is happening and when. Good example of this is the decision making during the iteration. All the solutions take place between development team and the owner of the product. This way the feedback will come fast and process runs smoothly. Without this people management aspect process can not be accomplished. If the process is not effectively managed along, there will be process chaos. Performance management of a process should answer the question, how your process is doing (Thom 2010).

Process should be constantly under measurement and analyzed based on the findings. It should be renewed and developed like stated in section 4.1. From the operation point of view, there are only three things which are important: Quality, time and cost.

Quality is the value measured by the customer. Value of the time is the timing. Things cannot happen too early nor too late. Cost is the cost lost to unnecessary work. All of this is managed

through flexibility, efficiency and customer satisfaction measurements, which are integrated to each other and one cannot be successful without the other.

Flexibility includes reaction to change and hopefully as well proactiveness. This is in the core of the management of the process and this way it is measured how the process fits into customers changing needs. There are two sides on this element, external and internal. External measures the customer changing needs, internal how fast the process is adapted to changes needed. Together they combine the flexibility measurement.

Efficiency tells directly how the process itself is managed in the aspects of customer satisfaction and flexibility. This gives direct input to quality, time and cost. Basis of the process should be optimized cost with right timing to satisfy customer needs with high quality. Efficiency should also include reducing waste (unnecessary work) where always possible (Hannus 1994.)

Customer satisfaction needs to be constant on all of the aspects mentioned above. This is in place, if the process is flexible and efficient and managed properly. Customer satisfaction should be constantly measured, to get the right feedback, to renew and develop the agile development process. We should also take a look how we can measure customer satisfaction (*figure 12)*.

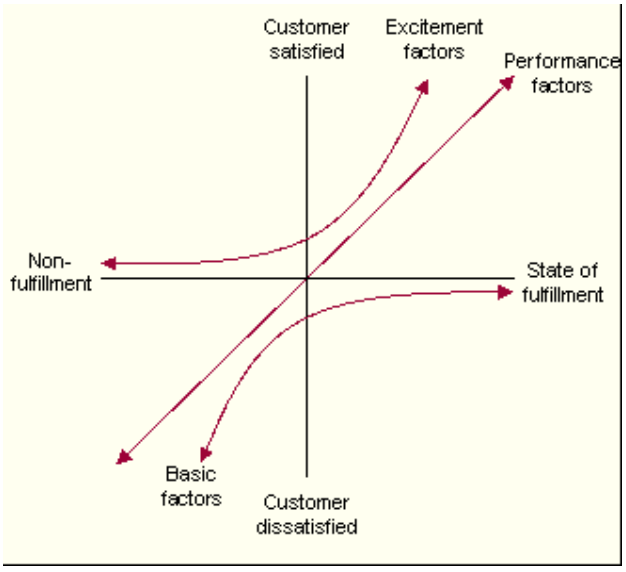

figure 12. Customer satisfaction picture (12 manage 2010)

The customer satisfaction model, from year 1984 by Professor N. Kano, is a technique that can be used for measuring client happiness. Kano's model divides customer satisfaction into three quality categories: Basic, excitement and performance. Basic requirements (basic factors in *figure 13*) are taken for granted by the customer as minimum requirements. Those will end up as dissatisfaction, if they are met or not.

Excitement requirements (excitement factors in *figure 13*) are ones which will surprise the customer in a good way. They will bring customer satisfaction if they are met, but they don't appear as dissatisfaction, if they are not met.

Performance requirements (performance factors in *figure 13*) are ones that are directly connected to customers needs. If the performance of the need is high, it will bring customer satisfaction, but if it is low, it will end up as dissatisfaction.

If we review this from the process management point of view, the process needs to be always functional and it doesn't bring any customer satisfaction still. If the changing needs are met, customer will be satisfied. As an ultimate goal, process to be really successful, it should be developed and renewed in a way, that it will bring excitement to the customer in the end (12 manage 2010.)

6 What is an error in software development

Error is a scientific term for uncertainty (Taylor 1997, 3). Error in software development can be called as well as defect or fault. There are no separation about the types of errors in this thesis, and all together all defects or faults are defined as errors (Peng & Wallace 1995, 1-2.) Error can vary from inconvenience to a catastrophe, and it can occur in any of the stages in agile software development (Senders & Moray 1991, 1).

Simple way of defining an error is that it is something, which departs from what is expected. This can be defined as anomaly. Anomaly is not necessary an error, but a deviation from expected. Based on this, errors, defects or faults can be defined as anomalies.

Error can be defined also to be a difference between observed and a theoretical value. Error can be an incorrect process or part of a process. Error can be an incorrect result based on machine or human action (Peng & Wallace 1995, 1-2.) Human action can be a mistake, a slip or an accident. “Mistake is an error whose result was unintended”. Slip is a poor execution of a good plan. Accident is a consequence of an error, which can decrease the performance of a system unpredictably or randomly (Senders & Moray 1991, 26-28.)

We need to also look at, how errors can be found? This can be done with software error analysis. It is a technique to locate, analyze and estimate data related to errors. It is based on statistics, prediction and reliability. Software error analysis is used especially for software development quality. It uses verification, validation and testing to find errors. Once the error is found, it needs to be analyzed if it is an actual error. It is also important to verify what is the nature and cause of the error, so it can be resolved.

One thing to note as well is that an error can be caused by multiple errors and multiple errors can be caused by a problem in a development process. If the root cause of the problem is not fixed properly, it may cause even bigger errors in the future (Peng & Wallace 1995, 2-1.)

The goal of the process is to be so advanced that there aren’t any errors to be identified. In real life, the goal is to reduce the number of errors to be in minimum by early detection and correction (Peng & Wallace 1995, 2-2.)

7 Errors found from agile development processes

This section handles the errors previously found from the agile development processes.

7.1 System engineering

One of the process errors in agile software engineering is the problems with system engineering and usability, compared to amount of code, which has been developed by the same people at the same time. This has been corrected with developing small pieces, which are made one at a time, or made in advance prototypes.

When development is done with short iterations, the risk is that testing is reduced as well. This has been corrected by test driven development, where test cases are created before development. Another correction is multitasking, but there is a threat to determine, whether the developer can actually develop lot of things at the same time (Nodder & Nielsen 2009, 53.)

7.2 Waterfall model adaptation directly to agile development

In waterfall model requirement specification, requirement analysis, development and testing are allocated into their own sections. Agile sees this as an error, when majority of the requirements analysis and design have been done before the development itself. In agile development they are part of the iterations. Another error is, if the majority of the testing is left at the end of the iteration (Larman 2007, 194.)

7.3 Errors in iterations

Agile sees as an error, if the iteration length is over 6 weeks. Agile recommendation is to have iterations, which are 1-4 weeks in length. Common error also is that iteration length is extended when the goal cannot be met during iteration. The process should be established so, that goal is reduced to meet the iteration.

In agile development iteration is never ready, if it has not been tested. Error in this case is that commonly it is thought that iteration ends on the defined date. Idea of the iteration is that it is done as defined in the process controlled way.

There is a common understanding that every iteration need to release something to production. Error in this is that every iteration doesn't need to release something. It is common that development requires multiple iterations before release.

In agile there has been an error, if the iteration has not had an iteration review or demo during it. Error is as well, if new work or individual is added during iteration. This is an error, as iteration is finished with the scope it has started with (Larman 2007, 123, 195.)

7.4 Scope of the development has been too difficult for iteration

In many cases, development team has tried to develop more things during iteration than it is possible. Agile rule is that less is more. Meaning is to develop only the things which add value for customer.

Common error is also predictive planning. Error in this case is the number of development iterations planned in advance. They are usually also defined how long they are and what they include. In agile, there needs to be a possibility to react to changes fast and turn the development to what ever direction needed and change the plans (Larman 2007, 196.)

7.5 Lack of effective communication

Common error in agile development is as well the lack of effective communication. Agile sees this as a problem, as "communication is a prerequisite for effective coordination". If the communication doesn't flow during the development process, the process cannot be successful, as problems don't come to surface to be fixed. There needs to be as well effective communication with the customer, to get the questions answered, to solve the problems or set the guidance of the development (IBM, 2010.)

7.6 Lack of buy-in for an agile development approach

Common error is that there is not enough buy-in for an agile process to be used in development. Agile sees this as a crucial error, if there has not been customer or high management buy-in for agile development to be used. In agile, this error is the most common error. This brings out the previous error that there is lack of communication and end-user or customer is not actively participating to the process.

There is also the change resistance error. Customer follows the way how they use to work and tend to delegate all the errors back to the development team, to figure out themselves, rather than solving them. In addition, there is as well the development team buy-in. Agile development is a team effort and agile sees as an error, if the team is not performing as it is defined, in rather parallel than in sequence (IBM 2010; Larman 2007, 238.)

7.7 Faulty methodology

Agile sees as an error, if the methodology which is used, is not agile. Many agile processes use incremental approach when they should use iterative approach. The incremental approach is like a sequence of perfect small waterfalls, but they concentrate only to part of the process or problem area. This becomes as a major error as when development is never stable, there is lots of rework needed to stable the development instead of adding new functionality with every iteration (IBM 2010.)

7.8 Poor requirements gathering and documentation

Agile sees as an error, if the requirements have not been prioritized. Agile defines that requirements need to be always in prioritized order. Another error is that based on the requirements, it is really hard to do test cases which end up being guess cases from the customer point of view. Agile defines that all developed requirements needs to be tested with a valid test case. Error is as well that there are no lessons learned from previous iterations. Agile defines that even tough documentation is not the main thing in agile, it doesn’t mean it should be left out completely (IBM 2010.)

7.9 Errors in management of the process

One error is that development team isn't briefed, how agile development works. Agile is a team work, which is built around motivated individuals. Error is as well that there is no one voice of the product owner. There seem to be many masters, who want to prioritize things. Agile defines that there is only one product owner.

Agile sees as an error, if product owner is not involved at all with the development. Agile defines that product owner needs to be always available for the development team to answer questions and decide the direction for development to go forward.

There are also errors in meetings. Meetings tend to be too long. Agile defines that meetings should last only 20 minutes. Common error is also that there are no measurements in development process. Agile defines that there needs to be regular measurement of the impact of the delivery (Larman 2007, 128-129, 238.)

7.10 Errors in tools used

Agile sees as an error, if there are no defined tools in use in development. This becomes a big problem, when there is lots of time wasted trying to use unsuitable tools for the process, or members of the team are not using the same tools (IBM 2010.)

Another tool error is that, there is an understanding that, only high-tech tools can be used for the process. Agile advises to keep the tool usage as low-tech as possible, even to be post-it notes or wall papers. Of course, for documentation purposes, some high-end tools are advised to be used (Larman 2007, 196.)

8 Goal of the research

Goal of the research is to research the agile software development process used, and the errors found from it. Are the errors found from the process new, or same as introduced findings in the theory? Research takes into account that, for the found errors, there is reasoning in the theory part, why they are errors. Another research goal is to identify possible enhancements as recommendations to the process, which should be taken into use, based on the results of the research.

8.1 Method of the research

Method of the research was a questionnaire (Appendix 1) to persons, who have been participating to the process. Questions were about the process itself and about management of the process. As well, there was an interview for the business owner of the process with the same questions as in the questionnaire. Aim of the research was to research, if there are new errors existing in the used process compared to the ones mentioned in the theory part.

Research problem was focusing on following areas: Schedule, requirements, tool used for requirements, roles, business ownership, communication, meetings, and management of the process and overall satisfaction of the process. There were 12 questions and 3 possible answers: Yes, no and can't say. There was a request to provide a short comment, if the answer was no or can't say. There was also a possibility to provide comments as overall feedback.

All of the areas of the research have had previously found errors. Idea for this was that there might be new errors available compared to the ones previously found and defined in theory part. Questionnaire results were combined together and are presented in the results section. Questionnaire comments were compared to results of the interview and connections and differences were also reviewed. Errors found from the projects were excluded from the research.

8.2 Process in scope of the research

The process, which is used, is an agile development process with separated planning and development processes (*figure 13*). It has a business owner and lots of stakeholder's, who can

submit requirements. There is a defined person, called gatekeeper, for certain areas to validate requirements. Process is managed by a person, whose responsibility is to keep requirement tool up to date and call up all the meetings.

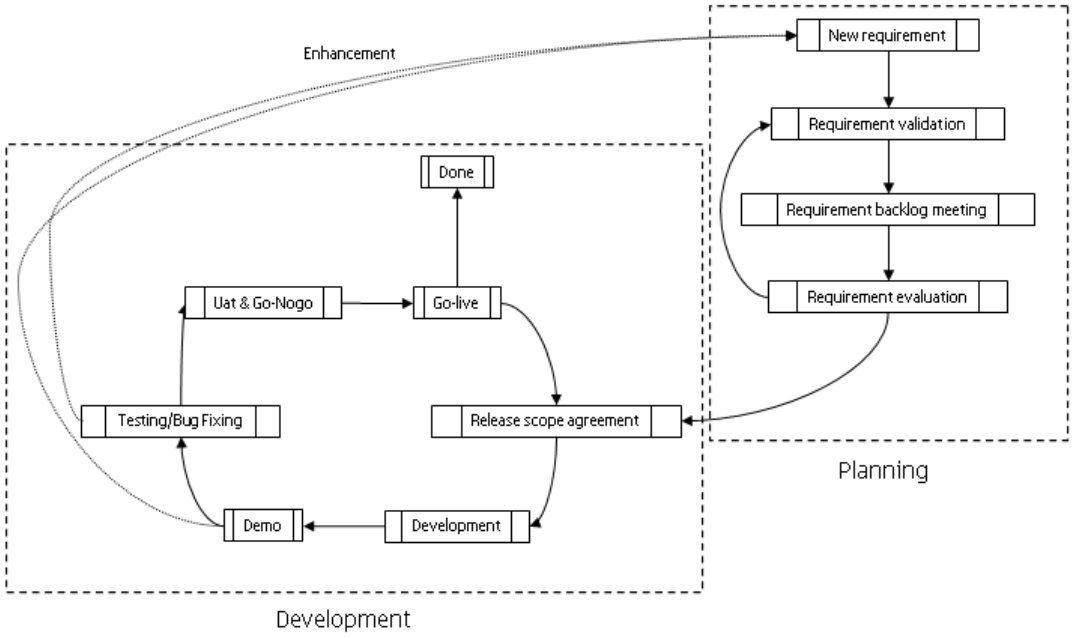

figure 13. Agile process picture (Koivisto, 2010)

Planning process consists of new requirement, requirement validation, requirement backlog meeting and requirement evaluation. Planning is an ongoing process, which feeds requirements to the development process. New requirement is something, which is submitted to one of the gatekeepers. It can be anything, but it needs to have a business owner and a business justification. Business justification is something, which brings value for the developed service.

Requirement validation is done by a gatekeeper, to make sure that the requirement is valid and it is not a duplicate for the existing ones. If it doesn't have a business owner or business justification, it is placed in the bottom of the queue in the tool. Requirement backlog meeting is a place, where all the gatekeepers share their top requirements as candidate requirements, to the next release (iteration). This occurs always before Release scope agreement. Requirement evaluation is done for candidate requirements, so all the top requirements are prioritized based on cost and business justification. This is needed, as all the requirements don't fit to the scope of the next release (iteration).

Development process consists of release scope agreement, development, demo, testing/bug fixing, uat & go-nogo, go-live and done. Development process is an ongoing iteration, which length is 4 weeks. Release scope agreement is a meeting, where the final release scope is defined, based on the validation of the requirements. Development is the actual creation (coding) phase done in the process.

Demo gives a presentation of all requirements committed to be released. There is also possibility for business owner and stakeholders to give feedback on results, if something needs to be optimized before user acceptance test. Demo's purpose is also keep business owner and stakeholders interested through visibility. Demo can also raise new identified requirements, which are handled as new requirements.

Testing/bug fixing is the time, when final tests are done. Bugs or small enhancements, which are not identified as new requirements, raised up during demo, are fixed before user acceptance test. User acceptance test is the final sanity check from business owner and relevant stakeholders, to make sure that the developed requirement is behaving as it should.

Go-nogo decision is a meeting, where is decided to give go or nogo to the developed release. This is given to whole release or part of it. If nogo is given, the nogo part needs to be reviewed. Decision needs to be also made, what needs to happen to it, so it can achieve go status. Before go decision, it needs to go through user acceptance test again. Go-live is the time, when developed requirements are deployed to production or live environment. Done means in principle that you are done with the release and everything in the scope of the release is deployed to production. It is also a validation of certain period of bug free days after go-live.

8.3 Process environment

Environment, where the studied process is running, is metadata driven environment based on standard semantic web technology. Metadata is structured data, which defines the properties of a resource. Resource defines the benefits, the quantities and often the quality as well. Meta-term itself implies a higher order. Metadata is almost as catalogs, which are based in libraries and archives.

Metadata record consists of many elements of pre-defined resources, which can have many values. Resources must be made visible in a way, so people understand whether they are useful

to them or not. Same applies to the World Wide Web. Metadata is a systematic way to describe resources and help for access them. If the resource supports to make it accessible, you should also define the metadata, which will maximize the means to find it (Taylor 2003.)

Semantic web technology is a World Wide Web service extension, where documentation is designed from machines point of view, so they can process it. This way software uses documents from the user's perspective and help users to find information. It also integrates information to the documents about interrelationship between certain things (Semantic Web 2010.) W3C defines Semantic Web to be web of linked data (W3C 2010.)

In Semantic Web, metadata needs to be presented with human readable tags, which are done with extensible markup language. Extensible markup language includes the information of the webpage author, relevant keywords and the information of the software tools which created the extensible markup language file (Altova 2010.)

The resource description framework is a W3C standard for encoding metadata and other knowledge on the Semantic Web with extensible markup language (Tauberer 2008.) Resource description framework triplet creates resources, resource properties and values for those properties. In English resource is a subject, property is a predicate and value is an object. Example: The thesis writer (subject) is (predicate) Petri (object) (Altova 2010.)

Ontology is a semantic data model for resources, which describes the primitive of information. Primitives are usually classes, attributes or properties and relations or relations between classes. Primitives include information about their meaning, restrictions and logic. In the context with databases, ontology can be seen as an abstraction of data models, but still used for knowledge about individuals and their attributes, and their relationships to other individuals (Gruber 2007.)

8.4 Results of the research

Combined questionnaire and interview results are presented in their own tables, but the results are grouped to relevant sections where they belong. These groups include the process itself, requirements, releases and management of the process. Questionnaire and interview goal was to get as deep insight as possible about the researched process.

Questionnaire included 12 direct questions with yes, no and can't say answers. Interview and questionnaire comments are inserted directly to conclusion. Results are presented here directly without commenting. All no and can't say answers were considered to be errors.

8.4.1 Process

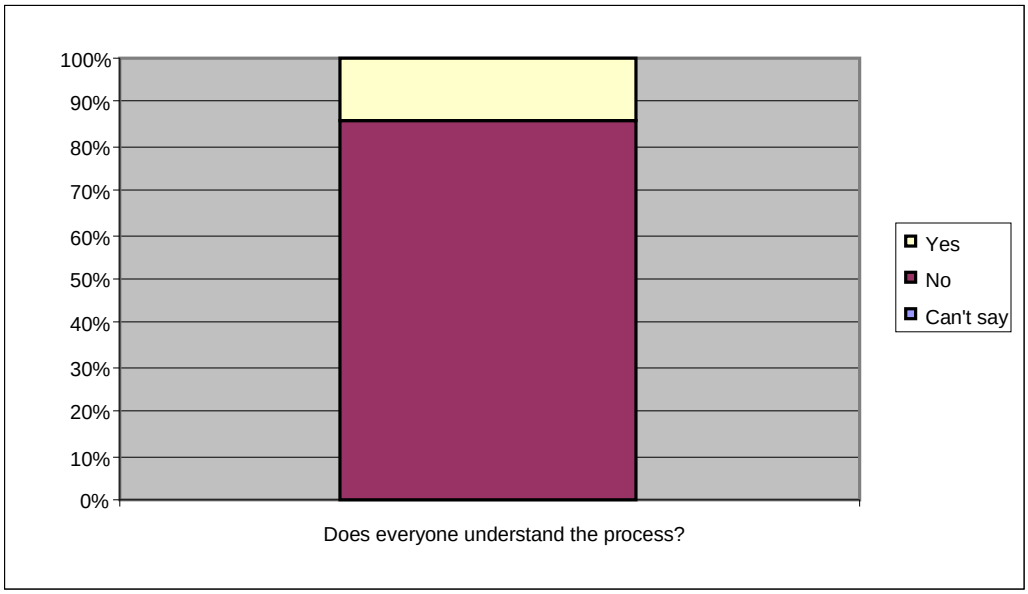

figure 14. Does everyone understand the process? (n=7)

86 percent of the answers (*figure 14*) stated that everyone involved in the process doesn't understand the process used for software development.

figure 15. Are you happy with the process? (n=7)

57 percent of the answers (*figure 15*) stated that they are not happy with the process used in software development.

8.4.2 Requirements

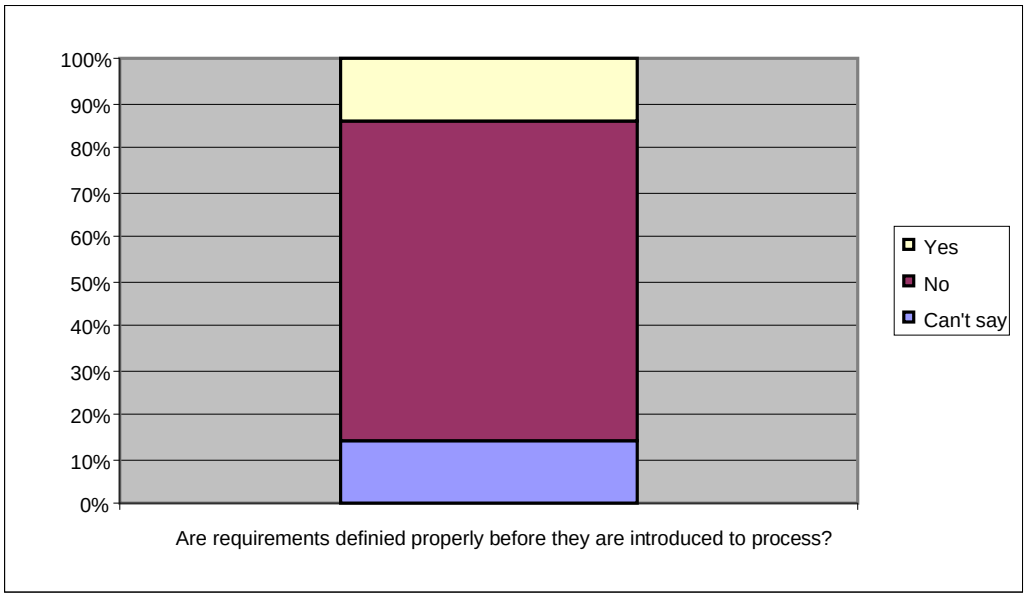

figure 16. Are requirements defined properly before they are introduced to process? (n=7)

71 percent of the answers (*figure 16*) stated that requirements are not properly defined before they are introduced to the process. 14 percent of the answers couldn't say if they were correctly defined or not.

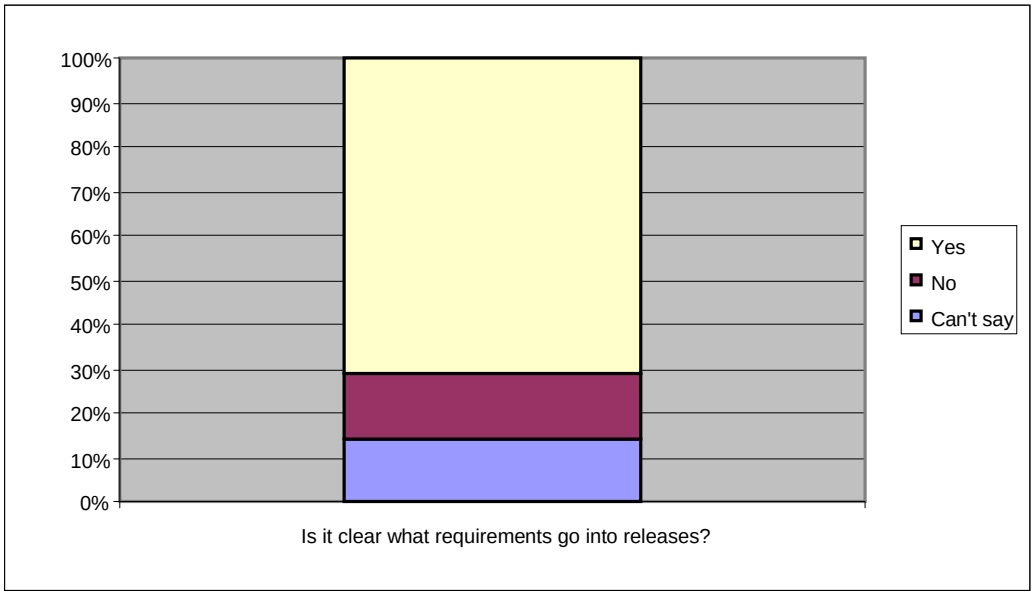

figure 17. Is it clear what requirements go into releases? (n=7)

72 percent of the answers (*figure 17*) stated that they know which requirements go into releases. 14 percent of the answers were confused which requirement go into releases.

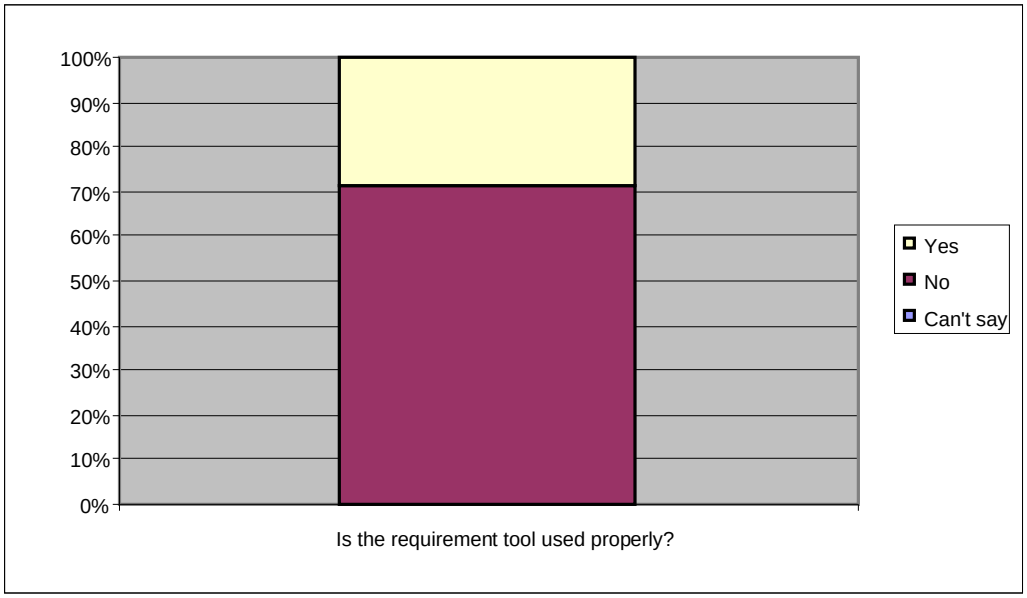

figure 18. Is the requirement tool used properly? (n=7)

71 percent of the answers (*figure 18*) stated that the requirement tool was not properly used.

8.4.3 Releases

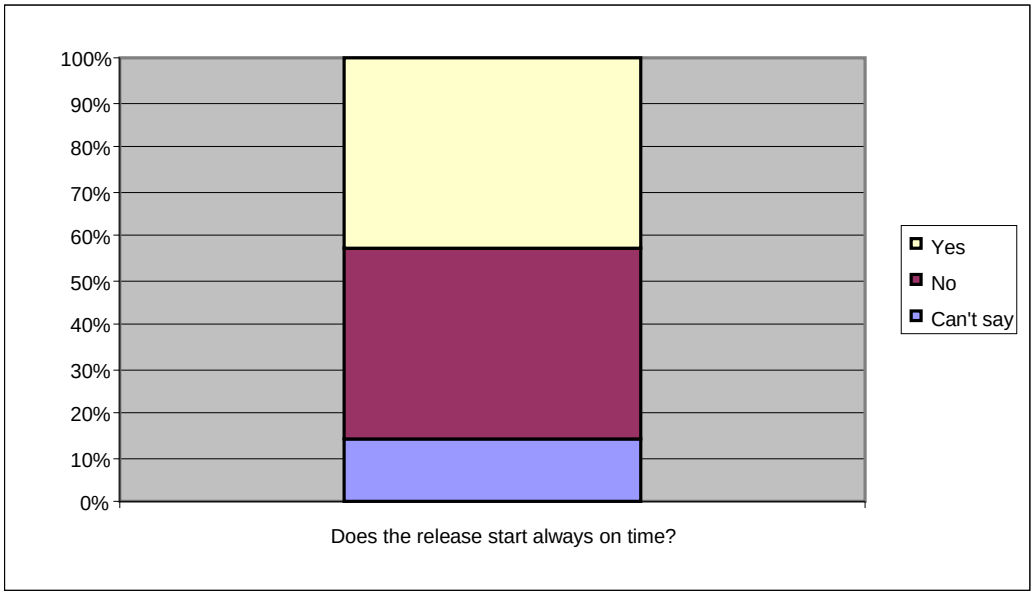

figure 19. Does the release start always on time? (n=7)

43 percent of the answers (*figure 19*) stated that they don't know if the release starts always on time. 14 percent of the answers couldn't specify if the release starts always on time.

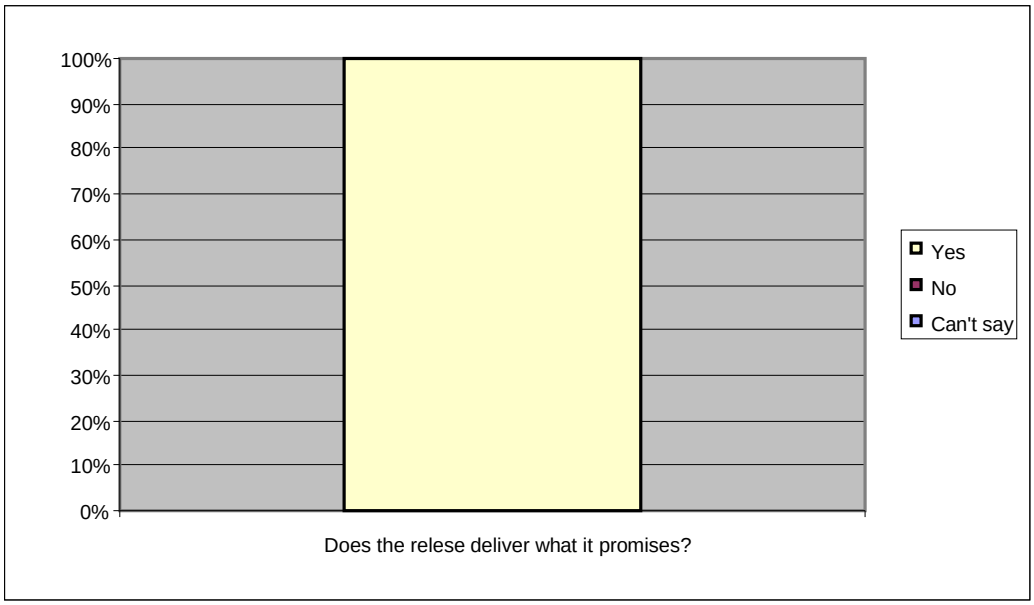

figure 20. Does the release deliver what it promises? (n=7)

100 percent of the answers (*figure 20*) stated that release delivers what it promises.

8.4.4 Management of the process

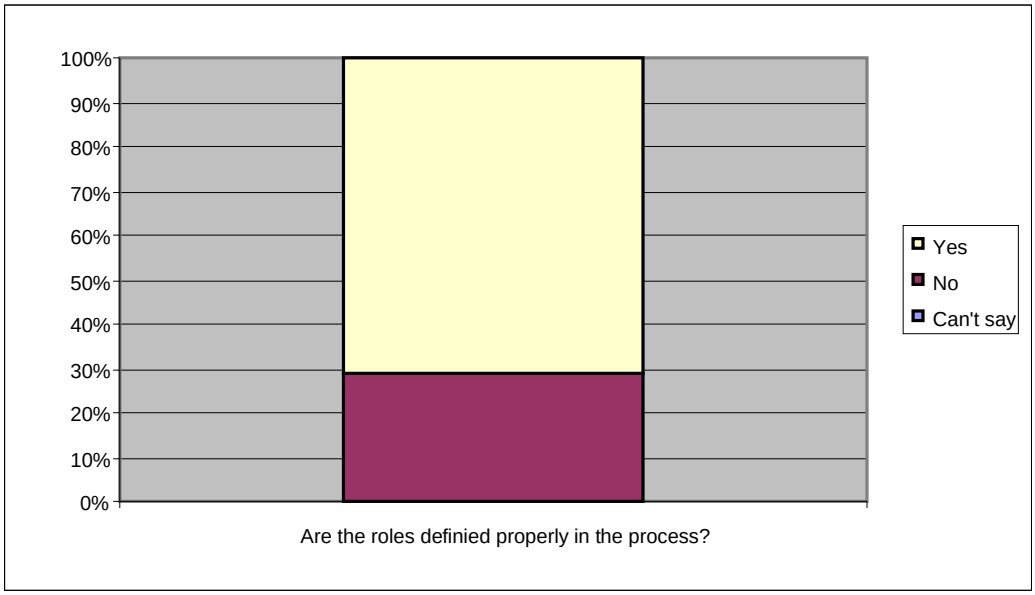

figure 21. Are the roles defined properly in the process? (n=7)

71 percent of the answers (*figure 21*) stated that roles are properly defined in the process.

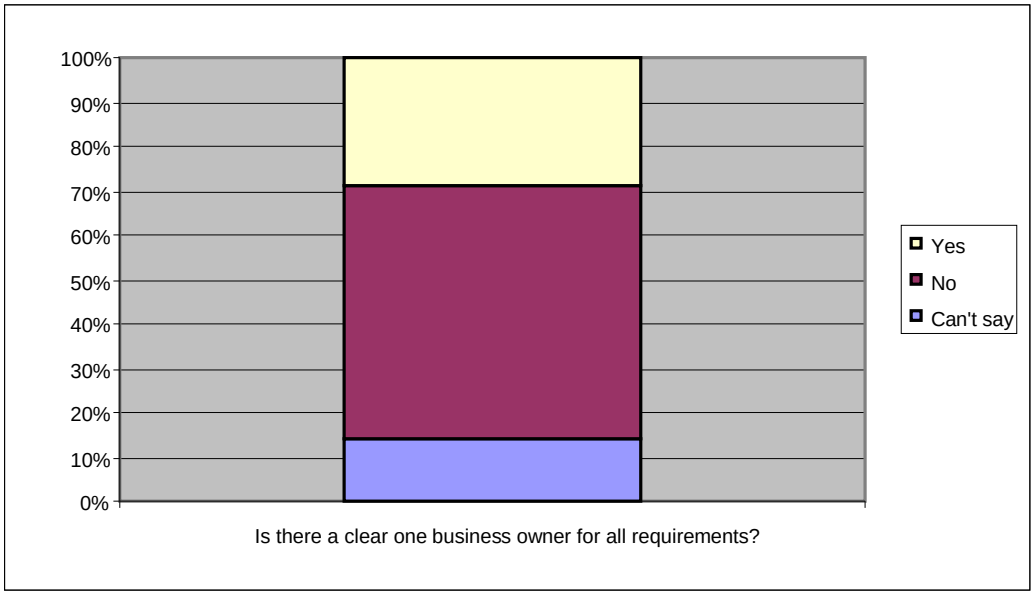

figure 22. Is there a clear one business owner for all the requirements? (n=7)

57 percent of the answers (*figure 22*) stated that there is no clear one business owner for all requirements. 14 percent of the answers couldn't say if there was a clear one business owner.

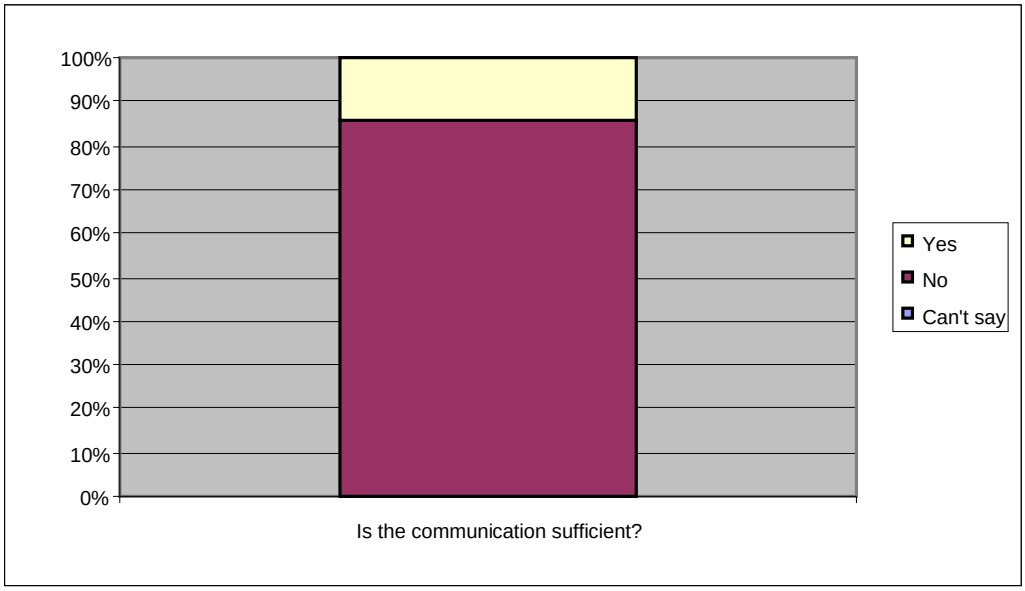

figure 23. Is the communication sufficient? (n=7)

86 percent of the answers (*figure 23*) stated that during the process communication was not sufficient.

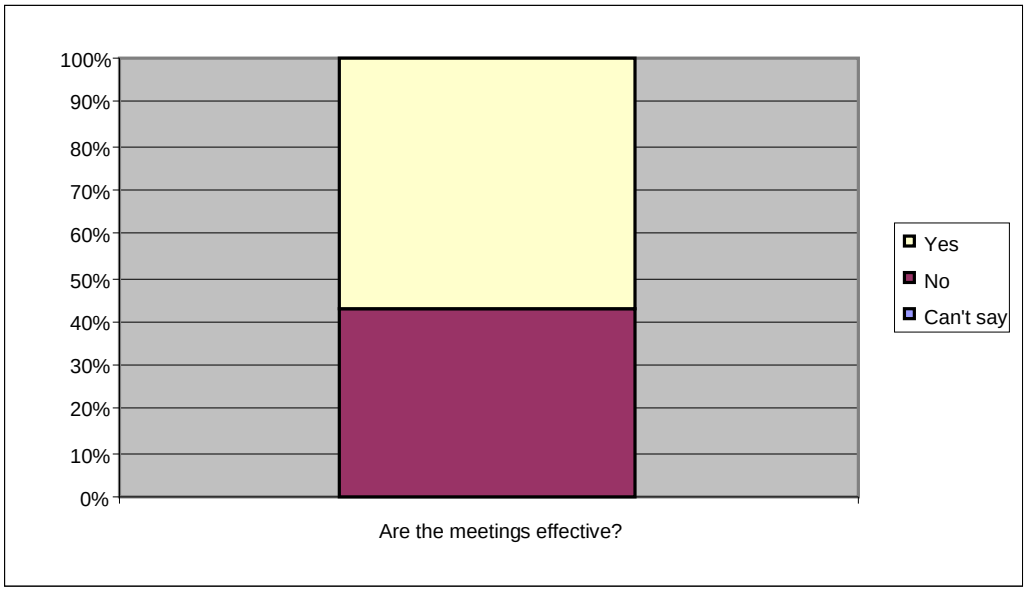

figure 24. Are the meetings effective? (n=7)

57 percent of the answers (*figure 24*) stated that the process meetings were effective.

figure 25. Is the process managed properly? (n=7)

57 percent of the answers (*figure 25*) stated that the process was properly managed.

9 Conclusion

Based on the collected information, following conclusions can be made as answers to the research problem. Following conclusions back up as well the hypothesis made in the introduction. Hypothesis was what errors may appear in agile development processes apart from the already found ones.

Errors, which were found during the research, were categorized so that, if the combined no and can't say percentage was under 40, it was categorized as low error. If the percentage was between 40 and 60, it was categorized as medium error. If the percentage was over 60, it was categorized as high error. Research found ten (10) errors. Five (5) are categorized as high, three (3) are categorized as medium and two (2) are categorized as low. Errors are introduced individually in categories from high to low. Introduced errors are also compared to previously found errors to be identified, are they new or existing errors. There is also a recommendation, in section 10, how to fix the new or existing errors found.

9.1 Errors categorized as high

Research found five (5) errors which are categorized as high: Everyone doesn't understand how the process works, requirements are not defined properly before they are introduced to the process, requirement tool is not used properly, there is no one clear business owner for all requirements and no sufficient communication in development process. Below is the description about the errors in more detail.

Everyone doesn't understand how the process works. There seem to be too many process steps, which everybody doesn't understand. There is a feeling that anyone outside the requirements team doesn't understand what requirements are and how to submit them. This is an error as process should bring value to the customer. Where is the value, if the customer doesn't know what to expect. This is an error as well for the fact that, process should keep things simple and to make sure communication and feedback is constant. This error relates as well to the communication and management of the process and this may also influence the scheduling a release, which because of this, may start late.

Requirements are not defined properly before they are introduced to the process. There is a feeling that some stakeholders get always in to the process with incomplete requirements. This is an error due to the fact that there needs to be a proper description including the intent and business justification in the requirement, before it is entered in to the process. This brings out another error that documentation of the requirements is not handled properly. Process lets through requirements which are not ready for the process and they are not documented correctly in the defined place. This error is related directly to the error, requirements tool is not used and managed properly.

Requirement tool is not used properly. This is a process management error. Requirement tool needs to be always up to date in prioritized order.

There is no one clear business owner for all requirements, as here might be more than one business owner for a requirement. This is a process management error as well as methodology error. This is an error due to the fact that, agile defines that there can be only one business owner, the product owner.

No sufficient communication in development process. There was uncertainty how the communication was organized in the process. There was also unclarity who is communicating to business owners, and feeling that all requirement estimates were not shared every time. This is a severe error as agility should fix communication problems, not bring more problems to it. Agile defines that there should be day to day communication to fix communication problems face to face or in a telephone. Especially this needs to be taken into account, when there is an integrated service involved and they don't realize what is expected from them.

9.2 Errors categorized as medium

Research found three (3) errors which are categorized as medium: Confusion when release starts and what the starting time is, meetings are not effective and process management is not effective. Below is the description about the errors in more detail.

Confusion when release starts and what the starting time is. Is it when the development starts or when the definition of the scope starts? This is an error due to the fact that release should every time start and end on a specific date. In this case, exact date should be defined and followed up through defined measurement. There was also a feeling that requirement analysis

and estimations are done in a rush to get the requirement into a release as the start time seemed to be flexible. This error might relate to a new error, where exception of the start date change becomes a rule and management of the process becomes more and more difficult, which ends up being management of the process error.

Meetings are not effective. Error in this is that all the agile meeting should be short and frequent. Now the meetings are not prepared good enough and lot of time is wasted paying attention to the wrong details or the details do not exist at all. Originator of the requirement is not always present, or people come in unprepared to the meeting. It seems as well that, lots of people are involved in the meetings, who offer no additional value to the whole process.

Process management is not effective. There is a feeling that the process does not have a role owning the overall process, to ensure that it is proceeding according to schedule. As a conclusion, all the errors found are tied with this error, as the success of the process is centralized in how the process is managed. This is an error due to the fact that manager of the process should always stay on top of the problems and solve them immediately or guide the problem solving to the right direction.

9.3 Errors categorized as low

Research found two (2) errors which are categorized as low: It is not clear what requirements go into releases and roles are not defined properly in the process. Below is the description about the errors in more detail.

It is not clear what requirements go into releases. This is a process management error. It should be clearly communicated by the manager of the process, which requirements go where. It is not the job for the requestor to guess should I submit this as a requirement or not. All requirements should be submitted to the same process.

Roles are not defined properly in the process. This is a methodology error as people involved don’t understand the basic terms needed for the process. This is an error also for the reason that agile process is built around motivated individuals who make the process successful.

9.4 Error comparison to previously found

As all the found errors are introduced, there needs to be comparison to the previously found errors, to see if there exist any new errors. Errors are compared in same order as they were introduced.

Everyone doesn't understand how the process works, can be compared to error in section 7.9. There is an error about development team is not briefed how the process works. Conclusion is that this is a previously found error.

Requirements are not defined properly before they are introduced to the process, is related to section 7.8. There is an error about poor requirements gathering and documentation. Conclusion is that this is a previously found error.

Requirement tool is not used properly, is related to section 7.10. There is not directly similar error found that requirement tool is not properly used. There is a proper identified tool which is not up to date. Conclusion is that this is a new error.

There is no one clear business owner for all requirements, is directly related to section 7.9, there are no clear one voice of the product owner. Conclusion is that this is a previously found error.

No sufficient communication in development process is directly related to previously found error in section 7.5, lack of effective communication. Conclusion is that this is a previously found error.

Confusion when release starts or what the starting time is, cannot be found directly from the errors previously found. There is an error that the release time has been extended, but there was not mentioned that, usually releases or iterations start time is not known. Conclusion is that this is a new error.

Meetings are not effective is directly related to previously found error in section 7.9, meetings are too long. Conclusion is that this is a previously found error.

Process not managed properly is a previously found error. Process not managed properly can be a new error as well, so there is no direct relation to previously found errors defined in theory part. Conclusion is that this is a new error.

It is not clear what requirements go into releases has not been mentioned in previously found errors. Conclusion is that this is a new error.

Roles are not defined properly in the process is related to previously found errors in sections 7.6, development team is not performing as it is defined and section 7.9, there are no clear one voice of the product owner. Conclusion is that this is a previously found error.

9.5 Summary

First conclusion is that even tough every release delivers what it promises, 57 percent of the people involved are not happy with the process. This is kind of conflicting as agile states that if your releases are high quality and you deliver your releases on time, you have a happy customer. Conclusion is still that, if your customer is not happy with the process, it needs to be changed.

Second conclusion is that there are four (4) new errors found from the process: Requirement tool is not used properly; confusion when release starts or what the starting time is; process management is not effective and it is unclear what requirements go into process.

Third conclusion is that from these four errors, there is one categorized as high, two categorized as medium and one categorized as low. Requirement tool is not used properly is high; Confusion when release starts or what the starting time is and process management is not effective are medium; It is unclear what requirements go into process is low. In my opinion all of these errors are visible due to the fact that process management is not effective.

Fourth conclusion is that these four errors in the research give answer to the hypothesis and to the research problem and make the research successful. To make the research completely successful, there needs to be still recommendations to all the errors found, how to fix them.

10 Recommendations

Following recommendations to all ten (10) errors found, specify what needs to be done to achieve better customer satisfaction. Recommendations are introduced in same order as they were presented as errors. When the recommendations are properly implemented, there is a process in place, which works and most of the people are happy with it.

As there is a feeling that most of the people involved in the process don't understand how the process works, there needs to be informative sessions about the process. Process needs to have visibility and it cannot be hidden to be effectively used. This can be easily done with face to face or virtual meetings or through teleconferences. There should be also regular checks with the people involved, what are the areas that are not working properly. As a result those check identify the areas, which need to be fixed.

If the requirements are not defined properly or requirement tool is not used properly or there is no one clear business owner for all the requirements, there should be one product owner, or a person who acts on behalf of business owner as a product owner. Product owner handles all requirement specifications on business side. This person works in co-operation with manager of the process, which makes sure that the definition of the requirement is sufficient before it is introduced to the process.

As there needs to be much more detail and thought for specifying the requirements, it is absolutely important that there is always concentration on requirements and in quality of them. It might help to establish a close dialog between requirement originator and final implementer to get the requirement specification correct. This problem should not exist, if the correct person is acting as a product owner.

This clears as well the error that requirement tool is not properly used. Product owner makes sure with manager of the process that all requirements are defined and documented properly before they are introduced to the process. Prioritization of requirements should be done based on return on investment as a number value. Highest value should bring benefit for real end users. This makes sure as well that requirement tool management and keeping it up to date is a constant way of working process.

To clear out the communication not being effective, manager of the process needs to enhance his/her ways of working and start excel the process. He/she needs to increase the visibility and communication about the process and lead the communication as an example to others. This would influence others taking part in the process to enhance their communication as well.

There needs to be weekly or bi-weekly reports, which detail what is in scope in coming period and what has been achieved in previous period. Report should also define, what is expected by whom during the period. This would build the circle of trust inside the process and make sure that everybody is up to date how the process works and what activities are ongoing. This would also help defining the roles in the process, and everybody knows what is expected from him/her.

Confusion when the release starts can be easily solved with, an even more clear distinction about planning and development as separate processes. Planning should be its own sub-process as it is drawn in the picture (*figure 27*) and development it's own. Those should never be mixed up.

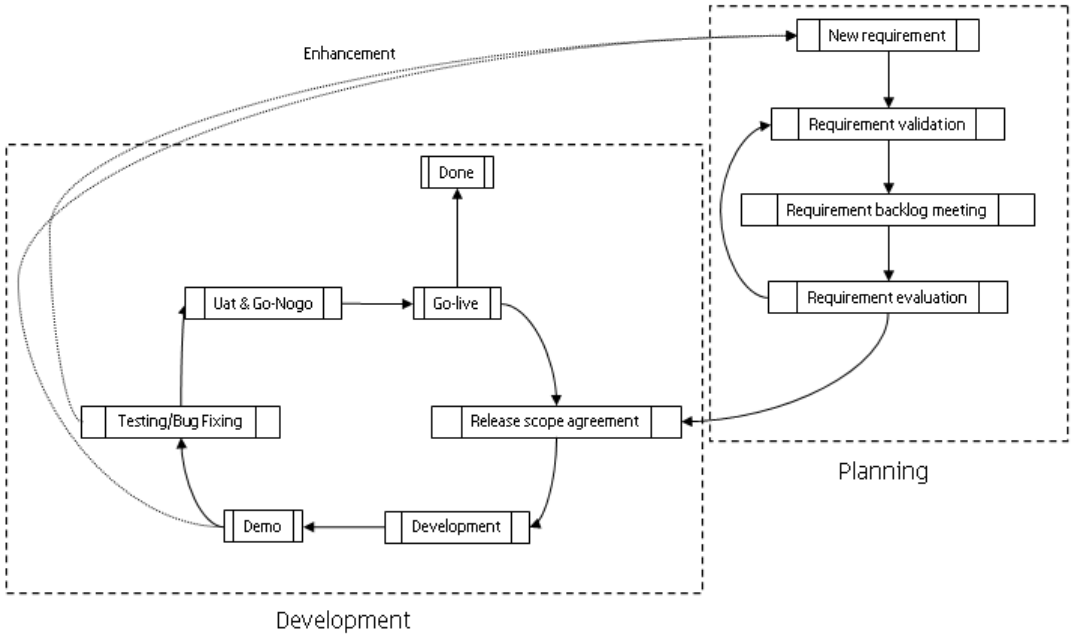

figure 26. Agile process picture (Koivisto, 2010)

Release scope agreement (*figure 26*) should only include requirements and only requirements, which are fully baked in the planning process. This would then be the date, when release always starts.

To get more effective meetings, there should be monthly workshop or ideas (initiation) sessions, which would help defining both problem and solution in requirements definition (planning) process. This workshop should include all persons necessary to make the workshop successful (business owner, product owner as well as any stakeholder asking for a feature, architect, vendor, etc.). All requirement details should be shared before the meetings, not just the titles of the requirements. If all of the above is happening, all necessary meetings will become effective as meetings will be prepared and meeting time is wisely used. All the meetings should be invited and managed by manager of the process.

There is also a misunderstanding in some parts what requirements should go into releases. All requirements should come to the same process and be deployed in releases. There should not be any separate process for requirement implementation. If there are urgent requirements, the process should be fine tuned so that it would be possible to release (*Go-live in figure 26*) part of the release, or one requirement, every week when needed.

From release capacity planning point of view, release should be a combination of new functionalities (features), enhancements (bug fixes or small changes) and service management requirements (hardware upgrades, non functional requirements). This would make sure that it would be clear what requirements go into releases. This would bring clear value to the whole process.

To get all above in place, process needs to be properly managed. Key to the process to be successful is the manager of the process. He/she needs to be constantly ready to adopt changes to the process, if they are needed to make the process better. Communication is the first customer satisfaction measurement, which needs to be in place as the errors are appearing through communication. This will solve the process management error.

Even if the release delivers what it promises every time, there needs to be key performance indicators established, which measure how successful was the release. It should measure the defined dates, what needs to happen when, and was all the promised requirements implemented. This way, there will be always quality releases and happy customer in the process.

11 Summary

Thesis introduced basics of software development and software engineering. In addition it specified how agile software development broke loose from waterfall model through iterative development and what the different ways of doing agile software development were. Moreover thesis introduced what is a process and what value does it bring when you are using processes. Furthermore thesis specified what an agile process is and how agile process works. As well it defined what roles needed to be in place to make process successful and what rules needs to be followed it to be agile.

Thesis gave an overview what is an error and how it is defined. Scope of the research was to find new errors in defined agile development process and compare, are those similar to what agile development process errors have been previously found. Research was successful as it found new errors from defined agile process. Research found also same errors, which had been previously found. Finding these errors helps studying the defined process in a way that, it can be analyzed and optimized to make it better. New errors were related to following areas: Requirement tool, schedule, process management and requirements. All of the areas had previously found errors. There was also a need to give recommendations how the errors found should be fixed. All errors found can be fixed, when process management is fixed and optimized. Key to all is management of the process and the person doing it.

Research shows that the process works, but it needs to be optimized for the changing customer needs. There needs to be more distinction on defining the requirements and developing them. These two sub-processes need to be kept separate and never mix them. Definition work needs to have a clear one product owner, who knows how to define and prioritize the requirements.

Based on the feedback, even tough there are errors, release always delivers what it promises. In agile this should guarantee a satisfied customer. Based on the feedback tough, all the found errors need to be cleared to make sure that, the satisfied customer exists in the future as well. Follow up research would be needed in the future to see, do the same errors still exist. It would also reveal what kinds of results are visible and what areas in the process needs more optimizing.

References

12 manage, 2010, Customer satisfaction model, http://www.12manage.com, 30.5.2010

Agile, 2010, Agile over rup part4, http://agileconsulting.blogspot.com/2009/02/agile-over-rup-part-4.html, 2.6.2010

Agile manifesto, 2001, Manifesto for agile software development, http://agilemanifesto.org/, 11.10.2010

Agile methods, 2010, Scrum- art of possibilities, Agile, http://www.ketteratkaytannot.fi/fi-FI/Menetelmat/Scrum, 27.5.2010

Altova, 2010, What is the semantic web?, http://www.altova.com/semantic_web.html, 2.7.2010

Ambler Scott, 2007, Strategies for scaling agile software development, https://www.ibm.com/developerworks/mydeveloperworks/blogs/ambler/entry/agile_and_rational_unified_process?lang=en, 9.6.2010

Ambler Scott W, 2009, Roles on agile teams, http://www.ambysoft.com/essays/agileRoles.html, 9.6.2010

Bogue Robert, 2005, Cracking the code: Breaking down the software development roles, http://www.developer.com/mgmt/article.php/3490871/Cracking-the-Code-Breaking-Down-the-Software-Development-Roles.htm, 28.6.2010

Buzzle, Rational unified process methodology, 2010, http://www.buzzle.com/articles/rational-unified-process-rup-methodology.html, 2.6.2010

Coplien James, 2009, Scrum, Co-location, Scrum: It´s all about common sense training material, 1.6.2010

Coplien James, 2009, Certified scrummaster training, March 2009

Defined Consulting, 2010, Agile waterfall picture,
http://www.definedconsulting.com/images/methodology.jpg, 29.05.2010

Gatherspace, 2010, Agile software development,
http://www.gatherspace.com/static/agile_software_development.html, 30.5.2010

Gruber Tom, 2007, Ontology, http://tomgruber.org/writing/ontology-definition-2007.htm, 2.7.2010

Grubb Penny, Takang Armstrong A, 2003, Software Maintenance Concepts and Practise, 2nd edition, World Scientific Publishing Co. Pte Ltd., Singapore, 6.6.2010

Hannus Jouko, 1994, Prosessijohtaminen – ydinprosessien uudistaminen ja yrityksen suorituskyky, HM&V research Oy, Espoo, Suomi, 2.6.2010

Hypethot, 2010, Software development methodology,
http://www.hyperthot.com/pm_sdm.htm, 4.6.2010

IBM, 2010, Iterative process picture,
http://www.ibm.com/developerworks/rational/library/content/04March/3070/3070_fig1.jpg, 30.5.10

IBM, 2010, The seven habits of effective iterative development,
http://www.ibm.com/developerworks/rational/library/1742.html, 22.6.2010

IBM, 2010, Unified modelling language, http://www-01.ibm.com/software/rational/uml, 5.6.2010

Keyes Jessica, 2004, Software Configuration Management, Crc Press Llc, Florida, USA, 6.6.2010

Larman Graig, 2007, Agile & Iterative development, 8th edition, The Agile Software Development Series, Pearson Education Inc., Massachusetts, USA, 1.6.2010

Lehtovirta Daniel, March 2009
Nodder Chris, Nielsen Jakob, 2009, Agile Usability: Best Practices for User Experience on Agile Development Projects, 2.6.2010

Nykänen Pirkko, 8.4.2009, Ketterä (agile) tietojärjestelmien suunnittelu / ohjelmistuotanto, http://www.cs.uta.fi/tjsum/TJSUM_08042009_PirkkoNykanen.pdf, 1.6.2010

O'Regan, Gerard, 2002, A practical approach to software quality, Springer-Verlag New York Inc, New York, USA, 7.6.2010

Peng Wendy W, Wallace Dolores R, 1995, Software error analysis, Silicon Press, New Jersey, USA, 15.6.2010

Puurunen Petteri, 2010, Haaga-Helia ICT development ICT4TD021 course material, 4.6.2010

Reaktor, 2010, Scrum, http://www.reaktor.fi/web/en/technology-and-research/scrum, 1.6.2010

Refresh Software, 2010, Development cycle picture, http://www.refreshsoftware.com/SiteObjects/published/031230514FB35BB8010C7E5134182170/CEC877DF9BFF931578E1F74586AB5B56/file/Cycle.jpg, 29.5.2010

Rehn Ralf, 2010, Haaga Helia Business processes ICT2TD006 course material, 25.5.2010

Rossberg Joachim, 2008, Pro visual team system application lifecycle management, Springer-Verlag New York Inc, New York, USA, 28.6.2010

Scacchi Walt, 2001, Process Models in Software Engineering, http://www.ics.uci.edu/~wscacchi/Papers/SE-Encyc/Process-Models-SE-Encyc.pdf, 4.6.2010

Scrum Alliance, 2010, Scrum, http://www.scrumalliance.org/learn_about_scrum, 2.6.10

Semantic Web, 2010, Semantic web, http://semanticweb.org/wiki/Main_Page, 30.6.2010

Senders John W, Moray Neville P, 1991, Human error: Cause, prediction and reduction, Lawrence Erlbaum Associates Inc., New Jersey, USA, 28.6.2010

Shore James, Warden Shane, 2008, The art of agile development, O'reilly Media Inc., California, USA, 11.6.2010

Six sigma, 2010, Value, http://www.sixsigma.fi, 26.5.2010

Stankovic Alma, 2010, Web-projektien aikainen testaus, https://publications.theseus.fi/bitstream/handle/10024/16599/Alma_Stankovic.pdf?sequence=1, 18.10.2010

Syracuse University, 2008, What is a process?, http://its.syr.edu/eps/services/process/what_is.html, 25.5.2010

Swebok, 2004, Guide to the Software Engineering Body of Knowledge (SWEBOK), http://www.computer.org/portal/web/swebok/htmlformat, 4.6.2010

Tauberer Joshua, 2008, Rdf, http://www.rdfabout.com/intro, 2.7.2010

Taft, Darryl K, 2010, Agile development hitting the mainstream, http://www.eweek.com/c/a/Application-Development/Report-Agile-Development-Hitting-the-Mainstream-539452/, 18.10.2010

Taylor Chris, 2003, Metadata, http://www.library.uq.edu.au/iad/ctmeta4.html, 1.7.2010

Taylor John R., 1997, An introduction to error analysis, 2nd edition, University science books, California, USA, 15.6.2010

Testing Brain, 2010, Software testing, http://www.testingbrain.com, 5.6.2010

Thom William, 2009, People, process and performance in project management, http://www.pmhut.com/people-process-and-performance-management-in-project-management, 29.5.2010

Tietoviikko, 2007, Ketteryys on kilpailuetu,
http://www.tietoviikko.fi/taustat/kaikki_jutut/article135334.ece, 14.10.2010

University of Waterloo, 2001, Software engineering, http://www.softeng.uwaterloo.ca, 4.6.2010

Väyrynen Virve, 2009, Onnistunut testaus ja ketterät menetelmät,
https://publications.theseus.fi/bitstream/handle/10024/7221/Vayrynen_Virve.pdf?sequence=1, 18.10.2010

W3C, 2010, Semantic web, http://www.w3.org/standards/semanticweb, 30.6.2010

Wang Yingxu, King Graham, 2000, Software engineering process, Principles and applications, Crc Press Llc, Florida, USA, 7.6.2010

Wells Don, 2009, Extreme programming: A gentle introduction,
http://www.extremeprogramming.org, 1.6.2010

Wells Don, 1999, Iterative development,
http://www.extremeprogramming.org/rules/iterative.html, 31.5.2010

Wisegeek, 2010, What is iterative development?, http://www.wisegeek.com/what-is-iterative-development.htm, 31.5.2010

Appendices

Appendix 1. Questionnaire

Could you please give yes or no as answer to the questions? If you cannot answer the question, please answer can't say. If you answer no or can't say, explain shortly why.

1. Does everyone understand the process?

2. Does the release start always on time?

3. Are requirements defined properly before they are introduced to process?

4. Is it clear what requirements go into releases?

5. Is the requirement tool used properly?

6. Does the release deliver what it promises?

7. Are the roles defined properly in the process?

8. Is there a clear one business owner for all requirements?

9. Is the communication sufficient?

10. Are the meetings effective?

11. Is the process managed properly?

12. Are you happy with the process?

Do you have any other comments?