Aalto University
School of Economics

Case of three Scrum teams: Agile software development methods at Nokia - The people perspective

Organization and Management

Master's thesis

Minna Räisänen

2012

Department of Management and International Business
Aalto University
School of Economics

Aalto University
School of Economics

Case of three Scrum teams: Agile software development methods at Nokia – The people perspective

Master's program in Management

Master's Thesis

Minna Räisänen

2012

Department of Management and International Business
Aalto University
School of Economics

Aalto University
School of Economics

CASE OF THREE SCRUM TEAMS: AGILE SOFTWARE DEVELOPMENT METHODS AT NOKIA – THE PEOPLE PERSPECTIVE

Master´s Thesis

Minna Räisänen

14th of May, 2012

Organizations and Management

Approved by the head of the Department of Management and International Business

__.__.20__ and awarded the grade ____________________________________

AALTO UNIVERSITY SCHOOL OF ECONOMICS ABSTRACT

Organizations and Management Master's Thesis 14th of May, 2012

Minna Räisänen

CASE OF THREE SCRUM TEAMS: AGILE SOFTWARE DEVELOPMENT METHODS AT NOKIA – THE PEOPLE PERSPECTIVE

Objective of the study

The main objective of the study is to find out how the practitioners of agile software development methods at Nokia experience and account for the new mode of operation. The research on the agile software development is scarce, since it has become a popular practice in organizations only within the last few years. In addition, even though agile methodologies claim to be more people centric compared to plan-driven, traditional methods, little research exists whether the practitioners of agile actually experience them being that. The aim of this study is to address this particular gap in research, by gaining a holistic and thorough understanding on people's perceptions on the given issue.

Research methodology

This research takes a qualitative case study approach to addressing the research question. Semi-structured, thematic interviews were conducted within three Scrum teams across Nokia.

Findings

The main finding of this study is that people are generally very satisfied with working according to agile software development mode and perceive that adopting agile methods into use has been the right strategic decision at Nokia. Another finding was that people perceive the impacts and effects of agile development methods (transparency, sustainable pace, teamwork, etc.) in differing ways. However, a majority of people think the effects are for the most part, positive.

Keywords

Agile software development, Scrum, employee perceptions

AALTO YLIOPISTON KAUPPAKORKEAKOULU TIIVISTELMÄ

Johtaminen ja Organisaatiot Pro Gradu –tutkielma 14.05.2012

Minna Räisänen

CASE OF THREE SCRUM TEAMS: AGILE SOFTWARE DEVELOPMENT METHODS AT NOKIA – THE PEOPLE PERSPECTIVE

Tutkimuksen tavoitteet

Tutkimuksen päätavoitteena on selvittää, minkälaisia kokemuksia ja näkemyksiä ketterien ohjelmistokehitysmenetelmien harjoittajilla on uudesta toimintatavasta. Aikaisempi tutkimus liittyen ketteriin ohjelmistonkehitysmenetelmiin on vähäistä, sillä käytännöt ovat yleistyneet organisaatioissa vasta viime vuosina. Vaikka ketterien ohjelmistonkehitysmenetelmien väitetään olevan ihmiskeskeisempiä kuin perinteisten menetelmien, asiaa on tutkittu vain vähän itse harjoittajan näkökulmasta. Tutkimuksen tavoitteena on lisätä tietoa kyseisellä tutkimusalueella ja saavuttaa kokonaisvaltainen näkökulma ihmisten kokemuksista ketteriin menetelmiin liittyen.

Metodologia

Tutkimuksen lähestymistapa on kvalitatiivinen tapaustutkimus. Aineisto on kerätty haastattelemalla ihmisiä kolmesta eri Scrumtiimistä Nokian sisällä. Haastattelut toteutettiin teemahaastattelun muodossa, puolistrukturoiduilla kysymyksillä.

Tutkimuksen tulokset

Tutkimuksen tulokset viittaavat siihen, että yleisesti ottaen ihmiset ovat hyvin tyytyväisiä ketterään ohjelmistonkehitystapaan ja pitävät niiden tarkoituksenmukaista käyttöönottamista Nokialla muutoksena parempaan suuntaan. Tutkimuksen toinen löydös oli, että ihmisten kokemukset ketterien kehitysmenetelmien vaikutuksista (läpinäkyvyys, tasainen tahti työnteossa, tiimityöskentely, jne.) ovat eriäviä. Suurin osa kokee ne kuitenkin positiivisiksi.

Avainsanat

Ketterä ohjelmistonkehitys, Scrum, työntekijöiden kokemukset

TABLE OF CONTENTS

1 Introduction

"In the field of software development, nothing is a stable, except for change. So why haven't we always developed software in an agile way?"

The previous quote is from one of the interviews I conducted at Nokia, stated by a team members working in an agile development team. I think it sums up in one sentence, why in the recent years organizations have started to adopt agile methods into use in growing numbers, especially in the instable technology and IT industries.

The mobile device market where Nokia is operating, continues to undergo significant changes. Recently, the most notable shift has been the broad convergence of the mobile telecommunications, computing, consumer electronics and Internet industries (Nokia annual report, 2011). Agile methodologies in software development aim at providing solutions to swiftly react to the turbulent market environment.

1.1 Background on agile development

In the field of software development, new methods are introduced quite often, but only a few of them have survived to the mainstream use in organizations (Abrahamsson et al., 2002). As the software industry started advancing quickly since its inception in the 1950s and 1960s, there was a need to better predict and control software projects that grew bigger and more complex. This need lead to the creation of the first software development methodologies, more commonly known as traditional, plan-driven or process-oriented methodologies (Leffingwell, 2011). The most well-known and most extensively adapted of the traditional plan-driven methods in software development history is the Waterfall model created by Winston Royce in 1970. It is also commonly used as a representative of traditional software development models, when the topic is discussed in literature. In the Waterfall model, software development occurs in a series of sequential stages that follow each other in a particular order. In other words, software development starts with agreeing on requirements and design, and the final stages are coding and testing (Leffingwell, 2011). Even though Royce (1970) suggested that there should be iterations of stages, in practice the Waterfall model is often applied so that the next stage doesn't begin until the previous one is completed (Leffingwell, 2011).

The traditional process-oriented software development models have been lately under serious criticism for being too rigid to react to today's turbulent organizational environment. As the requirements in the Waterfall model and other plan-driven models are decided on upfront, it doesn't really give developers a chance to make late changes in the specifications (McCauley 2001, cited in Abrahamsson et al., 2002). According to Boehm (2002, p. 64): "Traditionalists advocate using extensive planning, codified processes, and rigorous reuse to make development an efficient and predictable activity that gradually matures toward perfection". Nandhakumar and Avison (1999, p. 176) argue that traditional software development models are too mechanistic to be used in detail and that they are "treated primarily as a necessary fiction to present an image of control or to provide a symbolic status".

There has been a paradigm shift in the field of software engineering during the past two decades in organizations and in literature towards lighter-weight i.e. agile software development methods (Leffingwell, 2011) which are claimed to be more adaptive and more people-oriented than traditional methodologies (e.g. Nerur et al., 2006). Adaptive approaches are also said to be a better fit for organizations when the requirements uncertain and volatile (Syed-Abdullah et al., 2006). Nowadays, it seems like agile philosophy in software development has reached mainstream in organizations. According to an extensive study conducted by Forrester Research Inc. (2010), based on three surveys conducted in 2009 inside 30 large IT companies, 35% of respondents stated that agile methods most closely reflect their software development process, while only 13% were still applying the Waterfall model (West & Grant, 2010).

A universal definition of agile software development does not exist in literature, which indicates that the concept is complex and multidimensional. In spite of a variety of definitions, usually all of them involve the concepts of speed and flexibility in order to respond to changes in the dynamic market environment (Kettunen & Laanti, 2008). Abrahamsson et al. (2002) argue that there are four characteristics which make a software development method an agile one. This is the case when software development is:

1) Incremental (small releases, with rapid cycles)
2) Cooperative (customer and developers working constantly together with close communication),
3) Straightforward (the method itself is easy to learn and modify, well documented)
4) Adaptive (ability to make last moment changes) (Abrahamsson et al., 2002, p. 17)

1.2 Purpose of the study and research questions

The research related to the new paradigm of agile software development is still scarce, and therefore there is a backlog of research problems to be solved (Rajlich 2006, cited in Dybå & Dingsøyr, 2008). Even though agile methods are claimed to have a human-centered approach to software development (Ceschi et al., 2005) and therefore the conclusion could be drawn that they would enhance employee wellbeing, little research exists to support the claim (Laanti 2012). The previous research on agile methods has indeed reported to bring business value to users, but little research exists on how the people in agile projects feel about the new ways of working, and how agile methods might affect wellbeing at work (Laanti 2012).

Nokia's motivation to research this particular topic is to gain an insight on how the employees that are part of the agile process perceive and experience the new work mode. It is important to study this subject, because research suggests that job design is one of the most significant factors that affect the wellbeing at work (Frenkel et al., 1998; Knight and McCabe, 1998, cited in Syed-Abdullah et al., 2006). The people's perceptions on agile software development methods and their impacts on wellbeing have been researched in the company previously by extensive surveys on people's opinions on agile development (e.g. Laanti, 2012; Laanti et al., 2011). These studies have produced quantitative data and therefore are able to give a general level understanding on people's experiences. However, no studies that provide fully qualitative data on people's experiences about agile development have yet been conducted at Nokia.

The purpose of this research is to fill this gap of research to some extent. The aim is to find out, how people feel about the transformation from a Waterfall-like plan-driven approach into using agile methods in software development, and how they perceive agile methods in relation to previous mode of operation. The aim is to find out, what kind of influences (benefits and shortcomings) have agile methods had on people's everyday work from various viewpoints (workload, stress level, empowerment, team dynamics, collaboration and perceived productivity and quality) as well as on the general wellbeing and contentment in one's work. Also, the aim is to find out, if there are distinctive characteristics (background, personality traits), which might have a linkage to certain kind of perceptions of agile development methods. The research question can be summarized as following:

How do the practitioners of agile software development methods at Nokia experience and account for the new mode of operation?

1.3 Introduction of the case company - Nokia

Nokia's roots go 150 years back, evolving from a riverside paper mill into being a global telecommunications giant. In between these years, Nokia has produced rubber products, cables and consumer electronics. From 1991, Nokia made a strategic decision to sell these business areas and focus on manufacturing mobile phones and telecommunications systems only, and it resulted in the advantage of being one of the first serious players in the emerging market of that time. The mobile phones market was one of the fastest growing in the late 1990's, and the company grew at a phenomenal rate at this stage. During five years, the turnover almost fivefolded, from 6,5 billion euros in 1996 to 31 billion euros in 2001 (The Nokia story, 2011).

Nokia is still one of the leading manufacturer of mobile devices, with net sales worth of 38,7 billion euros in 2011. The company owns NAVTEQ, the leading provider of digital maps and navigation services. Nokia also owns Nokia Siemens Networks together with Siemens, and commenced a strategic partnership with Microsoft in 2011. At the end of 2011, Nokia employed 130 050 people, of which 73 686 were employed by Nokia Siemens Networks (Nokia's annual report, 2011; The Nokia story, 2011).

The new company structure was introduced in April, 2011 which is presented below in Figure 1. Nokia's key areas are divided into separate units for Smart Devices and Mobile Phones, the latter being aimed towards mass markets. Location & Commerce unit develops location products and services, such as Nokia Maps. NAVTEQ is naturally integrated into this unit. The Markets team is responsible for marketing, sales, and developing Nokia's local ecosystem. The company is organized as a matrix organization (Nokia – Our structure, 2011).

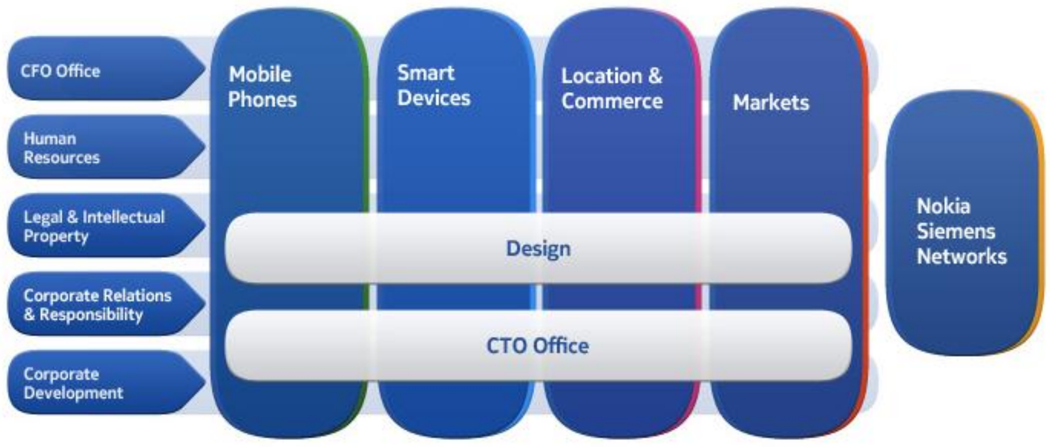

Figure 1: Nokia's company structure. (Nokia – Our structure, 2011)

At the moment, Nokia is facing challenging times due to the intensifying competition in the smartphone market. The whole mobile communications industry is going through substantial changes, because of the growing significance of the Internet communication. Consumer electronics and Internet industries are converging together to form a broader industry, providing with a varying range of different products. As a result, the market for smartphones has shifted from a device-oriented strategy into a platform-oriented strategy. Today, industry participants are competing in providing hardware, software, services and an application environment to create high-quality differentiated smartphones. In order to stay competitive, Nokia started a strategic partnership with Microsoft and decided to develop a new operating system, Windows 7, and to gradually get rid of the Symbian platform. The first Windows phones were launched in October 2011, the Nokia Lumia 800 and the Nokia Lumia 710 (Nokia's annual report, 2011; The Nokia story, 2011).

Nokia states that their success in the mobile device market depends on factors, such as responding successfully to changes, ability to retain, motivate, develop and recruit skilled employees, preventing defects and other quality issues of products and being able to increase the speed of innovation, product development and execution in order to provide products and services to the market in a timely manner (Nokia's annual report, 2011). In Nokia's annual report, it is stated that the employees have to adopt a mind-set of a "challenger", by paying attention on results, speed and accountability. The aim is to accelerate the speed of execution by developing new ways of working. This includes faster decision-making, reducing complexity and improving responsiveness to customer needs and market trends (Nokia's annual report, 2011). The latter discourse is very much in tact with the aims in agile philosophy, such as those of simplicity, speed and responsiveness to customer needs and to market environment (e.g. Abrahamsson et al., 2002)

The fierce competition and Nokia's current position in the markets is one of the main reasons for re-evaluating its processes and as a result, adopting agile practices widely into the whole organization. By adopting agile methods into use, Nokia aims at improving factors such as productivity, product quality, and the morale of personnel. Many organizations in the USA started adopting agile practices shortly after the creation of the Agile Manifesto (see chapter 2.2, p. 17). At Nokia Siemens Networks the adoption process started in 2005, and at Nokia Corporation in 2007 (M. Laanti, e-mail, November 17th, 2011). The motivational factors for adopting agile practices at Nokia are presented in more detail in chapter 2.2.7.

1.4 Methodology

In order to address the research dilemma of understanding people’s experiences thoroughly, I have chosen a qualitative research approach. Qualitative approaches in business research are suitable, when the aim is to interpret and understand a phenomenon in a holistic way (Eriksson & Kovalainen, 2008). The research was conducted as a case study at Nokia, by conducting interviews with open-ended and semi-structured questions on 16 people working in three different agile software development teams, using Scrum as their guiding agile methodology. The informants had a varied experience on working according to agile and traditional development methods. A more comprehensive description of the research methodology used is given in chapter four of this report.

The limitations of this study are linked with the qualitative methods used to address the research question. Since the study was conducted as a case study inside one specific organization, the results cannot be generalized into other organizational settings as such. Another limitation in qualitative research is that the research population is usually small in size, which means that results cannot be straightforwardly generalized into representing the thoughts of a larger population. Only 16 people were interviewed in this study, so the data represents only their subjective experiences on this matter (Eriksson & Kovalainen, 2008).

1.5 Structure of the study

This study begins with presenting the history of software development methods, and how and why companies have in growing numbers, started to switch their mode of operation from a traditional, plan-driven way into using agile methods. I will shortly explain the concepts of agile and traditional, plan-driven software developments models, and describe some of the models in more detail. I chose to present the Waterfall model as a representative of traditional models, and Scrum as a representative of agile models. These models were chosen to be presented, because in most cases, the previous mode of operation in software development at Nokia could somewhat be described as a Waterfall-like process. Similarly, Scrum is applied as the most commonly used agile method at Nokia today (Laanti, 2012), which is why it was chosen to be discussed in more detail of the agile methods. In addition, Scrum is the main agile method that the teams of my sample are applying at the moment. These two models are at the moment also most extensively applied from all methods, traditional or agile, across organizations (VersionOne 2010; West & Grant, 2011). Since Nokia's objective is not to restrict the use of agile methods to software development only, the idea of scaling agility to all levels of the organization in the form of scaled-up Scrum/Scaled Agile Delivery Model is introduced as well. At the end of the following chapter, I will present the main motivational factors for Nokia and organizations in general, to adopt agile practices, based on their alleged benefits.

In chapter number three, I will present the state of research in agile software development, and introduce the main findings that are related to the themes which are the target of my research case study. These are the factors regarding the implementation process of agile, effects or perceived effects that agility has on productivity and product quality, effects on teamwork and collaboration, effects on customer satisfaction and finally the practitioners overall satisfaction on the methodologies and effects on wellbeing and job satisfaction.

After reviewing previous literature, I will describe the research methodology used for this study in chapter four. In chapter five, I will present the main findings of my empirical study, based on my interpretation and analysis of the research data produced in the interviews. In chapter six, I will provide with conclusions of the findings, and try to find support to my findings provided in previous research. At the end, I will suggest some theoretical and managerial implications based on my findings.

2 Software development methodologies

2.1 Plan-driven, traditional methodologies

At the beginning of software development history in the 1960's and 1970's, no explicit or formalized development methods existed yet. Leffingwell (2011) describes this as "cut-and-try" or ad hoc approach, which means that software development process doesn't follow any given plan. The existing hardware at that time was rather limited and malfunctioning, and therefore the emphasis of software development was mostly on programming and solving technical problems. This kind of primitive approach to soft-ware development might have been sufficient at that time, when existing technology was limited and software projects were still small in scope (Avison & Fitzgerald, 2003). However, as the technology advanced and became more complex rapidly, the size of software projects grew bigger as well. When projects are bigger, they are simply harder to control and economically there is usually more at stake. To mitigate the risk of failure, more standardized and disciplined perspective towards software development was need-ed. In order to gain controllability over the process of software development, first soft-ware development methods were created. These first, "traditional" software develop-ment models are also known as process models or plan-driven models. As the name suggest, plan-driven software development focuses on planning everything from the start of the project until the end. The first software development models were character-ized by a lot of planning and dividing the work into a process with certain phases that follow each other (Leffingwell, 2011).

One of the first software development models was known as the Systems Development Life Cycle (SDLC), created by Winston Royce in 1970, more commonly known as the Waterfall model (Avison & Fitzgerald, 2003). It is probably the most well-known and most extensively adapted of the traditional process-models in software development throughout its history. It is also commonly used as a representative of traditional soft-ware development models, when the topic is discussed in literature. There are also a number of other traditional software development models, of which the second most famous one probably being the Spiral Model, which can be seen as the original “response to the Waterfall Model (Abrahamsson et al., 2002). However, only the Waterfall model was chosen to be introduced in more detail, because of its significance. The choice was supported by the fact, that Nokia describes its previous approach to software development as a Waterfall-like process.

2.1.1 Waterfall model

The Waterfall model is used as a typical representative of the plan-based, traditional software development methods. It became very popular and has been applied extensively in organizations since its launch in 1970, and it was also used as a basis for later process models.

The Waterfall model consists of development stages, that are meant to be completed in sequential order, and so that one phase has to be complete before the next one can begin. Altogether there are seven stages in the model, starting from system and software requirements, analysis, and program design phases, and ending at coding, testing, and finally the operation stage, which includes using and maintenance of the software (Royce, 1970).

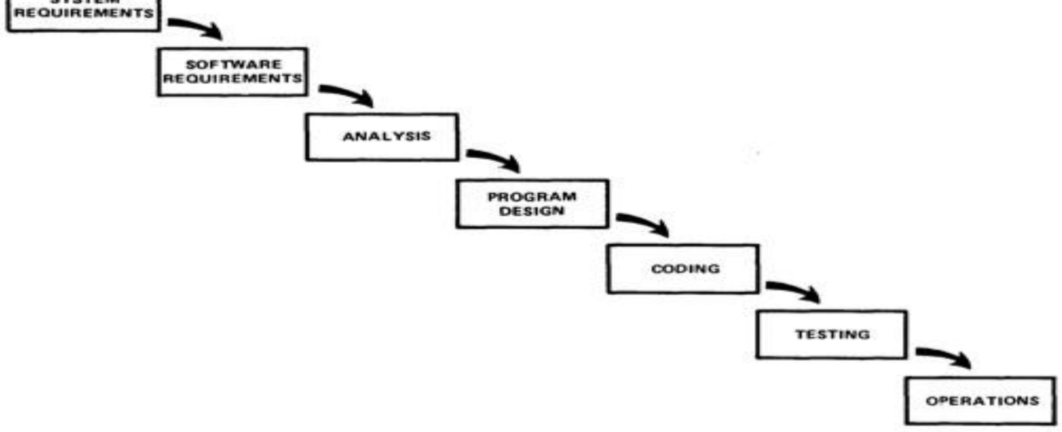

Figure 2: Waterfall model. (Royce, 1970)

The idea in the Waterfall model is to maximize the early work in software development, such as planning and analysis, so that requirements can be as detailed as possible (Royce, 1970.) The Waterfall model has been criticized for too long and too detailed planning and documentation, and thus changes in software are not easy to make after the carefully documented and detailed plans have been made. Also the projects can often take a longer time to be completed, since previous stages should be finished before the next one can begin. For example, plans have to be completed before any coding should begin. However, the Waterfall model indeed included a notion of iteration of phases if problems were encountered or changes required during the process, but in practice this is often ignored. Due to this, bugs in code are often not noticed before the testing phase, and thus the quality of the software and usability are being compromised as well (Avison & Fizgerald, 2003). Royce (1970, p. 335) himself noted, that the simple Waterfall model (such as in Figure 2, p. 10) "has never worked on large software development efforts", but it is still widely adapted in organizations, probably due to its simplicity and seemingly logical approach (Leffingwell, 2011).

In the Waterfall model, different roles such as program designer, systems analyst, programmer and tester are defined (Nerur et al., 2005). People are responsible for different stages in the process in accordance to their role. For example, a program designer takes care of the design stages while programmers are in charge of the code and testers handle testing. The communication between team members happens mostly through extensive documentation, which is done during the requirement, specification and design phases (Nerur et al., 2005). The documentation is also meant as a user manual. Royce (1970) stated that without documentation the software can be used only by the people who developed it. Customers play an important role during the early stages of the process when product specifications are defined, but their participation in the later development stages is minimal (Nerur et al., 2005).

2.2 Agile Methodologies

Agility is often associated with concepts such as quickness, alertness, liveliness and nimbleness. In software development, agile methods aim at stripping away as much of the heaviness of processes as possible, which is commonly associated with traditional methods. Agile methods promote quick response to different variables, such as changing environments, changes in user requirements, accelerated project deadlines, and the like (Erickson et al., 2005). In order to reach these goals, the focal aspects of agile i.e. light methods are speed and simplicity; development teams concentrate only on functions that are needed immediately, delivering them fast, collecting feedback efficiently and reacting rapidly to business and technology changes (Abrahamsson et al., 2002).

In 2001, 17 well-known software "gurus" including many of the creators of agile development methods, gathered together in a ski resort in Utah to achieve common ground on their core beliefs on software development (Leffingwell, 2011; agilemanifesto.org). The gathering was an important event, because the outcome of the meeting was a manifestation called the Agile Manifesto, defining value statements and principles, which still work as a basis of defining agility in software development (Leffingwell, 2011). This gathering of the "Agile Alliance" is often regarded as the official starting point of the paradigm shift in the field of software engineering (Abrahamsson et al., 2002).

The Agile Manifesto consists of four value statements, which all have a similar form. According to (Fowler & Highsmith, 2001, p. 29) "The first segment indicates a preference, while the latter segment describes an item that, though important, is of lesser priority". The latter values include stereotypical characteristics of traditional, plan-driven software development methods, which often are criticized in the field of software engineering (plans, documentation, lack of teamwork and lack of customer-orientation.) The Agile Alliance defines the values of agile development as following (http://agilemanifesto.org):

1: Individuals and interactions over processes and tools.

2: Working software over comprehensive documentation.

3: Customer collaboration over contract negotiation.

4: Responding to change over following a plan.

In the first value statement, the alliance doesn’t disregard the importance of processes and tools, but gives even greater importance to interaction between skilled individuals. In relation to this, comprehensive documentation is not necessarily bad, but the primary focus should remain on delivering working software. Continuous collaboration with the customer, throughout the whole development process is important. Contracts and projects charters may provide some boundaries, within which the parties can work, but they are not sufficient on their own in order to get satisfied customers. Strictly following a detailed plan can have dire consequences in today’s turbulent world of business technology. Plans should be made, but more important is for the development team to respond to external changes and therefore be able to alter plans during the process (Fowler & Highsmith, 2001). Following an agile development method makes making changes easier, since all the methods propose the development of software in small, incremental cycles (2-6 weeks in duration). This way the tasks and priorities can be evaluated between these cycles, and also risk is minimized, since schedule slippage will be visible in a few weeks (Abrahamsson et al., 2002).

In the same occasion, the members of the Agile Alliance also agreed on 12 principles which are rules that should be followed while using agile methods, in order to achieve the end goals presented in the latter value statements. The principles of the Agile Manifesto are listed as a whole on Appendix B (p. 103), but in short, they comprise of customer satisfaction, welcoming change, frequent deliveries of working software, working together and with motivated individuals, face-to-face communication, sustainable pace, simplicity, continuous reflection, technical excellence and self-organizing teams. Different kinds of agile development methods have been developed to provide with concrete practices that aim at achieving the value statements and principles defined in the Agile Manifesto. In the following chapter, I will introduce the frameworks of Scrum, and Scaled-up Scrum/Scaled Agile Delivery Model, which are the main agile methods followed at Nokia.

2.2.3 Overview of agile models

There are a number of agile methods, of which probably the most well-known and widely adapted are Dynamic Systems Development Methods (DSDM), Feature-Driven Development (FDD), Adaptive Software Development (ASD), Scrum, Extreme Programming (XP), Open Unified Process (Open UP), Agile RUP, Kanban, Lean and Crystal Methods (Leffingwell 2011).

VersionOne (2010) sponsored a survey called “State of Agile Development Survey” (2010) of 4770 participants from 91 different countries. The respondents were recruited via internet forums dedicated to software development, such as mailing lists and websites. The data generated in the survey was analyzed and summarized by an independent survey consultancy.

According to the survey, 58% of the respondents that apply agile methods describe Scrum as their most closely followed agile methodology (see Figure 3, p. 15). It is also common for companies to lean on Scrum project management practices, and complete them with XP practices in the actual development work; This Scrum/XP hybrid was the second most popular agile methods in use, with 17%. Lean development has also started to gain some popularity as an agile software development methods (2% of participants), and Kanban and Scrum practices combined together create the hybrid of Scrumban (3% of participants).

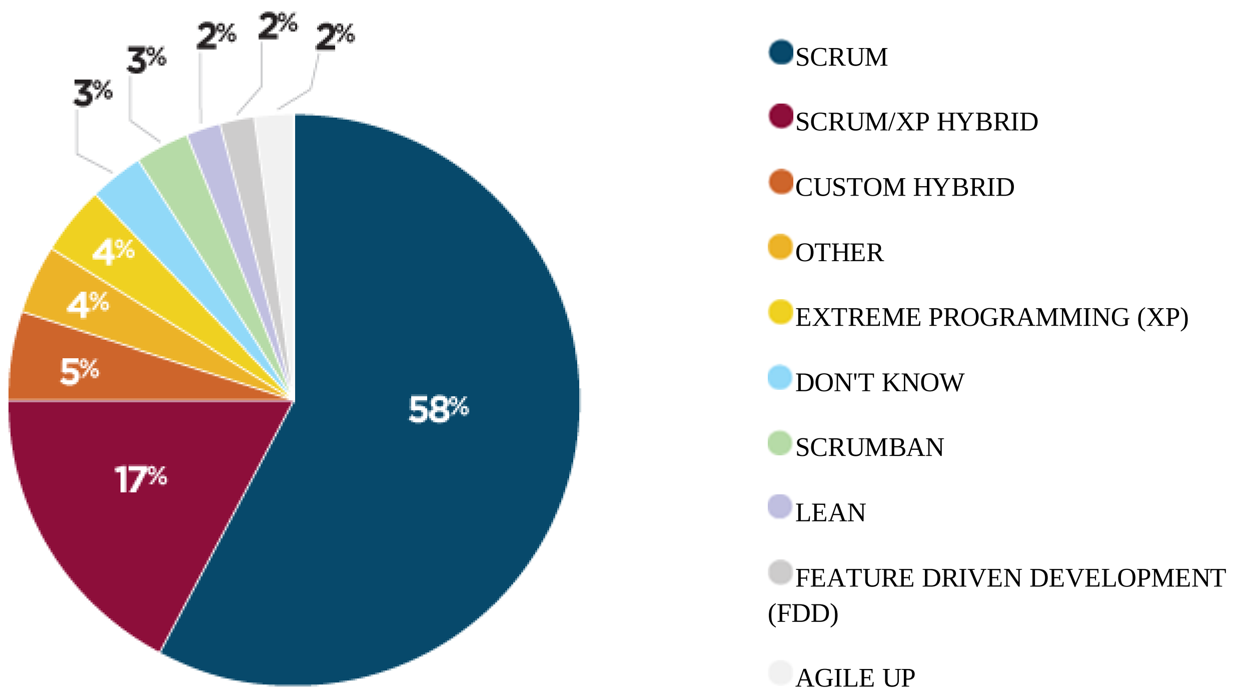

Figure 3: Most closely followed Agile Methodology. (VersionOne: Agile Methodology Survey, 2010.)

Laanti's (2012) quantitative survey study on agile development methods and wellbeing conducted at Nokia seems to indicate that Scrum is the most followed agile methodology among software developers there as well. The survey was sent to 10 000 people semi-randomly to different parts of the software engineering organization. However, the survey tool had technical problems and the final number of people it reached is not known. The total of 466 people answered the survey, and out of these respondents, 39% said that they follow Scrum, and 37% said that they follow ScrumBUT (i.e. should be following Scrum but do not apply all the practices). In addition, 10% stated they were following Scrumban and 5% were following Kanban. Only 4,5% of respondents were still working according to Waterfall, and Extreme Programming was used as the main agile method by only 0,6% of respondents (Laanti, 2012).

In the following chapters, I chose to present the Scrum framework in more detail, because it seems to be the most frequently used agile methodology at Nokia. This choice is supported by the fact that all of the respondents in my study stated that they apply Scrum as their main development methodology. Nokia is also scaling the Scrum process to cover the whole development process from a managerial level, all the way to the software development teams by following Dean Leffingwell's Scaled Agile Delivery Model, created in 2007. I will present the reasoning behind scaling agile processes, and present Leffingwell's model as an example of how agile practices can be scaled-up to the whole organization.

2.2.4 Scrum

The Scrum model in agile software development is based on Ken Schwaber's and Jeff Sutherland's ideas, and was first presented in a conference in 1995. The model became more popular after Ken Scwaber and Mike Beedle released a book in 2002, called "*Agile software development with Scrum*". Since then, Scrum has gained popularity, being the most commonly used agile software development process with XP (Schwaber, 2007). Unlike XP, Scrum isn't concentrated on software development practices, but it is more a project management tool. It doesn't provide any software development techniques for the implementation phase and therefore it is common to complement Scrum with other agile practices, such as those from the XP methodology (Abrahamsson et al., 2002). For example at Nokia, some of the informants of my study applied pair programming as one of the practices taken from XP. Scrum concentrates on how the team members should function together in order to produce the system flexibly in a constantly changing environment (Abrahamsson et al., 2002).

The term "Scrum" originally derives from a strategy in the game of rugby, meaning "getting an out-of-play ball back into the game" by working as a team. The term was used in management literature first in 1986 in an article by Nonaka and Takeuchi, where they used the term to describe "adaptive, quick, self-organizing product development process". The Scrum process has similar goals, since it aims at flexibility, adaptability and productivity (Schwaber & Beedle, 2002, cited in Abrahamsson et al., 2002, p. 29) In Scrum, the assumption is that in systems development, there are several environmental and technical variables (e.g. requirements, time frame, resources and technology), which will most probably be changing during the process. Regarding this notion, the Scrum process enables a system to be produced, when it is useful at that particular time

(Schwaber, 1995, cited in Abrahamsson et al., 2002). Scrum is an efficient project management tool in situations where it is difficult to plan ahead, for example because of a turbulent market environment (Dybå & Dingsøyr, 2008).

Instead of working according to a project mode (such as in Waterfall), work is organized into short development increments called Sprints, which usually last up to 30 days. The customer plays an important role in the Scrum process, and features of the product are chosen and prioritized together with the customer. The core element of Scrum are the feedback loops, which are enabled by constant communication in Daily Scrum meetings as well as the Planning meeting in the beginning of a sprint and a Review and a Retrospective meeting at the end of each sprint (Dybå & Dingsøyr, 2008).

2.2.4.1 Scrum team - roles

There are only three basic roles in the Scrum team, which are the Product Owner, the Scrum Master and the Scrum team member. Each role has a set of specific tasks and responsibilities and all management responsibilities are divided between these roles (Schwaber, 2007). In addition, there might be managerial, customer and other stakeholder roles who give their input into the Scrum process. At Nokia, all the basic Scrum roles are applied, as well as additional roles of a Product Manager, Release Manager and a Lead Developer (Nokia Intranet, training material).

A Scrum team should be small in size, consisting maximum of 9 people. The team is self-organizing and self-managing, which means that the team itself decides on how to allocate tasks among its members, re-organizes itself when necessary and is responsible for managing its own work. The team has therefore a collective responsibility for the success of each iteration and the project as a whole (Schwaber, 2007).

A Scrum Master makes sure that the Scrum team follows the practices, values and rules of Scrum in its work. He/she aims at ensuring that everything is running smoothly in the Scrum process, and tries to remove obstacles on the way. Scrum Master is supposed to be regarded not as a manager, but rather a mentor and coach for the team (Schwaber, 2007).

A Product Owner is responsible for managing, controlling and making visible the Product backlog list, which is a list of all the features and work that are needed in order to complete the product. A Product Owner is basically in charge of the project and he/she represents the voice of the customer and other stakeholders. Every task that is assigned to the Scrum team should be communicated through the Product Owner who updates and makes prioritizations to the Product backlog list (Schwaber & Beedle, 2002, cited in Abrahamsson et al., 2002).

2.2.4.2 Scrum process and practices

As stated earlier, Scrum doesn't provide any practices into the actual development work, but certain managerial practices and tools to the different stages of Scrum are presented in the model. These practices include defining and updating the Product backlog list, and based on the effort estimations, choosing work to be completed to the next Sprint backlog list. All the remaining work is listed on a Burndown chart, which can be made to the Product backlog as well as to the Sprint backlog. In this chapter, I will describe how the Scrum process works and what practices are included in it. The process is also presented in Figure 4 (p. 20), which might provide a more holistic understanding of it.

Work in Scrum process is organized into sprints, which last approximately one month. The outcome of each sprint is an increment added to the product that is decided by selecting variables from the Product backlog list into the Sprint backlog list. In the beginning of each sprint, the Scrum Master organizes a Sprint planning meeting, which comprises of two phases. At first, the customers, users, management, Product Owner, Scrum Master and the Scrum team decide on goals for the next sprint. In the second phase, only the Scrum Master together with the Scrum team, organize their work in order to achieve the goals that were selected into the Sprint backlog. (Schwaber & Beedle, 2002, cited in Abrahamsson et al., 2002)

A Sprint backlog is a set of items (called user stories) selected from the Product backlog, to be completed during the next sprint. Product backlog includes all the features and work, based on current knowledge, which are needed in order to complete the product. There is a constantly updated Product backlog list of Product backlog items, which are business or technical requirements for the product. The Product Owner is responsible for keeping the list up to date, and people from different positions, such as customers and developers, can suggest features to be added on the list (Schwaber & Beedle 2002, cited in Abrahamsson et al., 2002, p. 32).

The team is shielded from changes coming outside the team during the Sprint, and no new user stories are accepted. There has to be a “Definition of Done” (DoD) decided for the items on the Sprint backlog list. It defines when a feature is truly done, with a predetermined list of activities that must be done for the feature. In practice, the list may range from simple test cases to running code analyzers and going through heavier tests. (Schwaber & Beedle 2002, cited in Abrahamsson et al., 2002)

On the last day of the sprint, Scrum Master together with the Scrum team present the outcome of the sprint to customers, managers, users and the Product Owner in the Sprint Review meeting. The Product Owner will approve or reject the sprint result in this review. The participants review the product increment, and determine on upcoming activities and possible changes to the Product backlog. The Review includes a Sprint Demo where the team demonstrates to the customer, what it has built during the Sprint and how the developed code functions. The meeting takes usually from 4 to 8 hours for one month sprints and maximum of 5% of the duration of the sprint for shorter sprints (Schwaber & Beedle, 2002, cited in Abrahamsson et al., 2002).

The purpose for Sprint Retrospective meeting is for the Scrum Master and the Scrum team to raise issues regarding the whole Scrum process, and thus trying to continuously improve it. This meeting is also held at the end of each sprint, and the length is ideally 4 hours. (Schwaber & Beedle, 2002, cited in Abrahamsson et al., 2002)

A Daily Scrum is a 15-minute meeting that every member of the Scrum team should attend. It is advised to keep this meeting standing up, because it helps to keep it short so that people don’t get too comfortable sitting down. The meeting is structured so that every team member answers to three questions at their own turn: : What was done since last meeting, what will be done before the next meeting, and are there any problems that impede the work. (Schwaber & Beedle 2002, cited in Abrahamsson et al., 2002)

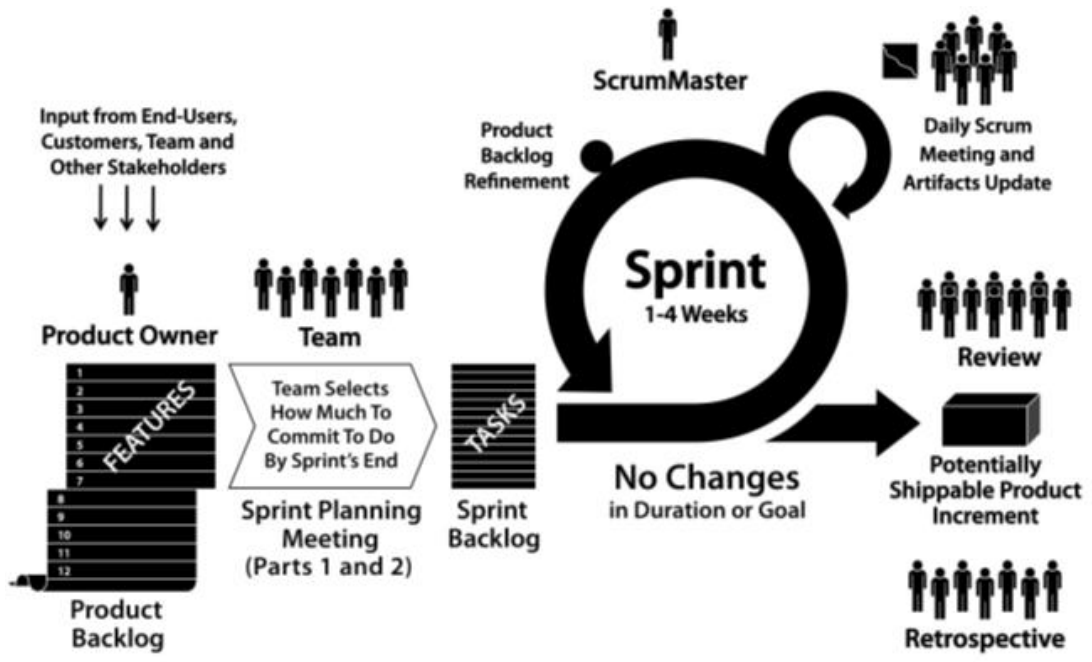
ScrumMaster
Input from End-Users, Customers, Team and Other Stakeholders
Product Backlog Refinement
Daily Scrum Meeting and Artifacts Update
Product Owner
Team
Sprint
1-4 Weeks
Review
FEATURES
Team Selects How Much To Commit To Do By Sprint's End
TASKS
No Changes
in Duration or Goal
Potentially Shippable Product Increment
Sprint Planning Meeting (Parts 1 and 2)
Sprint Backlog
Product Backlog
Retrospective

Figure 4: Scrum process. (Sutherland & Schwaber, 2011)

2.2.5 Scaling agile practices

Originally, agile methods have are developed to fulfill the needs of small and flexible software teams. As more companies have become interested in agile development, they are being applied widely in large organizations and projects as well (Kettunen & Laanti, 2008). There have been many attempts to enlarge agile methodologies, such as those of industrial XP and Scrum of Scrums. Kettunen and Laanti (2008) claim that these attempts to scale agile practices may not have been successful, because the current organizational environment has been too heavy to support agile practices. As Cockburn & Higsmith (2001, p. 132) stated: "an agile team working within a rigid organization has as difficult a time as agile individuals working within a rigid team". Kettunen and Laanti (2008) suggest that, in order for the company to work in an agile way, it is not enough to focus on project- and team level dimensions only as in a typical application of agile software development methods. Successful agile development requires an organizational environment, where agility is understood as a broader concept of an agile enterprise, where obstacles to reach agility are removed and enabling factors are enhanced (Kettunen & Laanti, 2008).

In order to create a suitable environment for the agile development teams to function efficiently, Nokia is deliberately pushing forward an agile enterprise structure. In order to reach this goal, Nokia is adopting Dean Leffingwell's Scaled Agile Delivery Model, where agility is embedded in all organizational practices starting from the managerial level where vision is created, moving all the way to the implementation phase where software development teams create the products. In the following chapter, I will present Leffingwell's (2011) ideas in more detail.

2.2.5.1 Scaled Agile Delivery Model

Dean Leffingwell presents a model for scaling agility into all levels of the organization in his book released in 2007 *"Scaling Software Agility: Best Practices for Large Enterprises"*. He calls the model the Agile Enterprise "Big Picture" or Scaled Agile Delivery Model (see Figure 5, p. 22), and it serves as both the organizational and process model for agile requirements practices (Leffingwell, 2011).

Leffingwell (2011) states that the motivation for larger organizations to adopt agile methods stems from the positive results in productivity, quality and morale achieved within agile teams. Even though agile methods are traditionally thought to work within smaller contexts, Leffingwell (2011) claims that they are applicable in extended and larger enterprises worldwide. Nokia is following Leffingwell's Scaled Agile Delivery Model in order to extend the Scrum process to all levels of the organization.

Leffingwell (2011) divides the organization in the Agile Deliver Model into three different levels according to their product definition responsibilities. The highest level of abstraction in product definition is obviously at Portfolio level, becoming more detailed at Program level, and finally consisting of detailed tasks at the Team level (see Figure 5). Different terms are used to describe these requirement artifacts (user stories, features and epics) at these different levels (Leffingwell & Aalto, 2009).

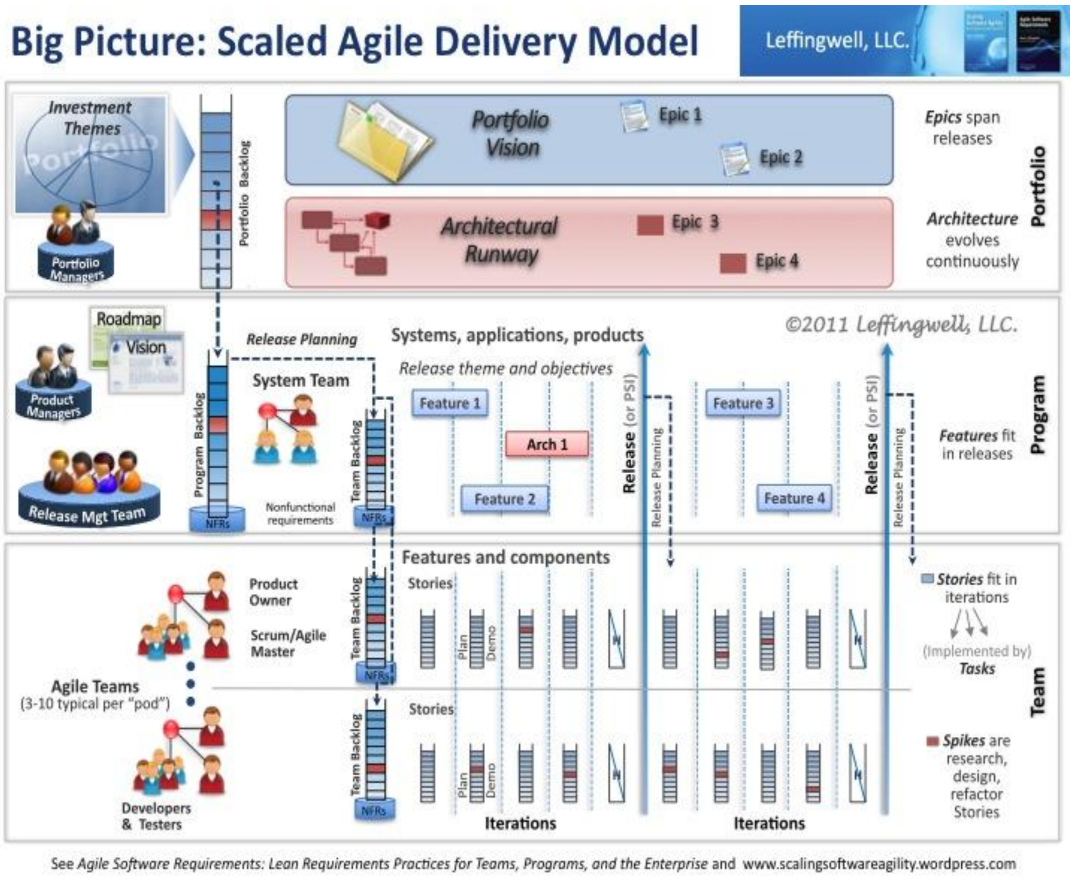

Figure 5: The Agile enterprise Big Picture: Scaled Agile Delivery Model. (Leffingwell & Aalto, 2009)

At the Portfolio level, the Portfolio Vision is derived from a set of investment themes. Investment themes mean the overall investment priorities driving the vision of the enterprise or the business unit. Investment themes should be in accordance with the enterprise business strategy, and they may stay largely unchanged up to a year. The Portfolio manager is responsible for deriving new epics from the investment themes into the Portfolio backlog, that are later broken down into smaller increments of workload (specific features on the program level and finally detailed user stories on the team level). Epic is therefore a term for the artifact used to describe the highest-level expression of a customer need (Leffingwell, 2011).

The dynamics and responsibilities of the agile team defined by Leffingwell are similar to those of a Scrum team (see chapter 2.2.4.1, p. 17), except that more roles are suggested. In addition to Scrum team roles presented by Schwaber and Beedle (2002, cited in Abrahamsson et al., 2002) of Scrum Master, Product Owner, and a small team of developers, Leffingwell (2011) suggests adding testers and test automation experts, and perhaps a tech lead. The team should be supported by architects, external QA resources, documentation specialists, database specialists, source code management (SCM)/build/infrastructure support personnel, internal IT, and whoever else it takes such that the core team is fully capable of producing working and tested software into the system baseline (Leffingwell, 2011).

At Nokia, additional Scrum team roles include those of a Lead developer, Release manager and Product manager. They are all Program level roles, except that of a Lead developer, who operates in both team and program level. A Lead developer is a top level specialist, who leads, and guides the Product team in the system development activities. A Release manager ensures that the product releases are planned according to Portfolio level priorities and has the ownership of the whole product release process, from planning to execution. Product manager's responsibilities lie in managing the delivery of medium or big scale projects by ensuring availability of required resources across Scrum teams. He also monitors that the product is delivered according to schedule and within the estimated budget and in high quality, ensuring customer/end-user satisfaction (Nokia Intranet, training material).

At the program level, there is a product team which the Product manager is responsible of putting together (Leffingwell 2011). At Nokia, the roles at the program level include a Product Owner, a Lead Developer and a Release manager. The Product team participates in Product and Release planning meetings, where they plan and prioritize requirements from epics, and redefine them into product features, and further into user stories which fit into sprints of 2-3 weeks (Nokia Intranet, training material). The Scrum team will decompose user stories into even smaller increments, called tasks, which are concrete steps that must be completed by an individual team member in order to complete the user story (Leffingwell, 2011).

In smaller organizations, there might be just a few of these kinds of agile teams. However, when enterprises get bigger, Leffingwell (2011) suggests that the agile teams would form groups or "pods" of agile teams who work together and build up larger functionalities into complete products, features, and so on (Leffingwell 2011). Leffingwell (2011) calls the process the Agile Release Train (ART), where pods of Scrum teams synchronize their release process into a standard cadence. This procedure was already practiced at in a more mature agile team (team 3), which was the target of my research. The team meets up once a month (after finishing two sprints) with members from a few other Scrum teams, in order to develop bigger entities which need integration with one another across the organizational borders.

The development work in the Scaled Agile Delivery Model is divided into a standard cadence of time-boxed iterations and releases (Leffingwell & Aalto, 2009), just as in Scrum. At Nokia, this means that work is usually organized into sprints of 2 or 3 weeks, product increment releases in every 8 weeks, and finally a customer release in every 1-6 months, depending on the size of the product. Local builds are made multiple times a day (see Figure 6, below).

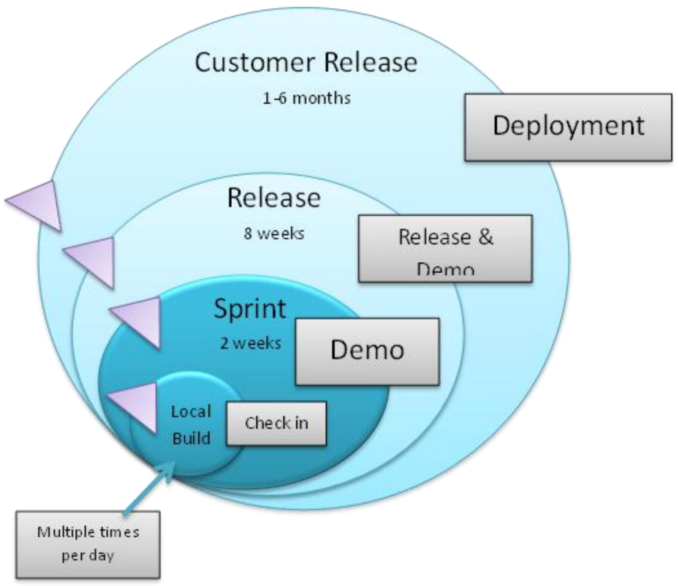

Figure 6: Nokia systems development process (Nokia Intranet, training material, August 28th , 2011)

2.2.6 Plan-driven versus agile software development

Agile methods are often understood as a kind of an antithesis or a counter-reaction towards what is regarded as traditional software development. Waterfall model and other traditional software development models have a rationalized, engineering-based approach to software development, in which it is claimed that problems are fully specifiable and that optimal and predictable solutions exist for every problem (Nerur et al., 2005). Leffingwell (2011) argues, that as software development projects grew larger and larger over time, the methods used to control the development process became heavier and heavier and thus made it slower and more rigid towards changes. As the competitive environment is changing rapidly, it is hard to predict how the products and their requirements are evolving. While the emphasis in the plan-driven, traditional software development was the aim towards predictability, extensive planning and documentation, agile methods, by contrast aim at relying on people and face-to-face communication rather than processes, in facing the unpredictable technology and market environment (Dybå, 2000).

The basic differences between agile and traditional software development models according to Nerur et al. (2005) are presented in Table 1 (below). According to Maarit Laanti (e-mail, November 17th, 2011), the positive traits of agile methods pointed out in this comparison account for many of the motivational factors for companies, including Nokia, for adoption more lightweight methods into use.

	Traditional	**Agile**
Fundamental Assumptions	Systems are fully specifiable, predictable, and can be built through meticulous and extensive planning.	High-quality, adaptive software can be developed by small teams using the principles of continuous design improvement and testing based on rapid feedback and change.
Control	Process-centric	People-centric
Management style	Command-and-control	Leadership-and-collaboration
Knowledge management	Explicit	Tacit
Role assignment	Individual—favors specialization	Self-organizing teams—encourages role interchangeability
Communication	Formal	Informal
Customer's role	Important	Critical
Project cycle	Guided by tasks or activities	Guided by product features
Development model	Life cycle model (Waterfall, Spiral, or some variation)	The evolutionary-delivery model
Desired organizational form/structure	Mechanistic (bureaucratic, with high formalization)	Organic (flexible and participative, encouraging cooperative social action)

Table 1: Traditional versus agile software development (Nerur et al., 2005)

To explain the comparison by Nerur et al. (2005) further, in traditional models, there is a rationalized, engineering-based approach which assumes that all the systems and problems are fully specifiable and predictable. This kind of thinking leads to extensive and detailed upfront planning, as well as the emphasis being on refining and optimizing repeatable processes. The process-centric approach was best served with a management style that involved hierarchy and a “command and control” management style. In traditional software development, the development process is usually following some kind of life cycle model (Waterfall, Spiral, etc.). This means, that there are distinctive phases that follow each other, with each involving specified tasks and outcomes to be performed. The life cycle models also assign distinctive roles to people (such as programmer or tester), so that each person has usually a specialized area of expertise and certain set of tasks they perform. The traditional methods also produce a large amount of documentation of codified processes and product knowledge, and the communication between people happens through reading these documents. Customers do play an important role in the beginning of the process by specifying their needs, but have a small part otherwise throughout the process (Nerur et al., 2005).

Agile methods are more people-centric, by relying on people’s creativity in problem solving, rather than processes and detailed planning. In agile development mode, work is organized in small and iterative cycles, which are characterized by work flowing according to adding product features, periods of reflecting on one’s own and the team’s way of working, collaborative decision making, rapid feedback and change as well as continuous integration of code into the system under development. Instead of organizing the work into one big project, it is broken down into small subprojects, with each one involving planning, development, integration, testing and delivery. Developers work in small teams, and tasks are rotated between team members, which ensures that knowledge is not monopolized by a few individuals. Communication with the team members and with the customer is frequent, and decisions are made collaboratively within the team and with the customer (Nerur et al., 2005). Leffingwell (2011) argues that traditional models weren’t able to fulfill the close co-operation with the customer needed during the entire development process, which is a crucial point in software development efforts. Project Manager’s role in an agile team is that of a facilitator and coordinator, instead of a commander. Agile development does not encourage extensive documentation beyond the code, and therefore product knowledge can be seen as tacit (Nerur et al., 2005).

2.2.7 Motivation for adopting agile methodologies

As discussed in the previous chapter, Nerur et al.'s (2005) grid (see Table 1, pp. 25-26) of traditional versus agile development, shows a lot of benefits of agile practices compared to the plan-driven approach. This comparison points out a lot of motivational factors for companies to change their development mode into more lightweight i.e. agile (M. Laanti, e-mail, November 17th, 2011).

According to Maarit Laanti (e-mail, November 17th,2011), the additional reasons for Nokia to adopt agile practices are very similar to the following alleged benefits of adopting agile practices, presented by Schwaber et al. (2007).

Schwaber et al. (2007) argue that companies adopt agile methods for many different reasons, and might sometimes be surprised about the actual outcomes these processes have to offer. For example, a company might adopt agile development methods for trying to increase productivity, when actually they will find out that the most impressive benefits come from quality issues, as well as visibility into the progress of the project. According to Schwaber et al. (2007), the main benefits of agility can be categorized in five different effects; reduced time-to-market, increased quality, reduced waste, better predictability and better morale. The authors, however, remind that although these benefits are widely recognized, some of them have not been empirically proven. This is mainly because not a lot of companies that have switched to the agile development mode, have gathered comparable "before and after" data. The previous research on agile development will be presented in chapter 3, of which some studies support the following statements. However, the following are the claimed benefits of agility, and serve as the main motivational drivers for Nokia to adopt agile practices. My study does not aim at testing if these hypotheses are true or not, mostly because many of these factors are best to be evaluated by quantitative measurements. Some of my interviewees did however perceive the following traits as being part of the main benefits of agile. The findings and results of the empirical part will be presented in chapters 5 and 6.

2.2.7.1 Reduced time-to-market

Schwaber et al. (2007) claim that the pace of development dramatically increases after adopting agile practices due to effective feedback, team collaboration, communication with the business and different kinds of quality practices. Because the idea is to produce working software at the end of each iteration, the business stakeholders can make the decision to deploy the software whenever they want to, if they evaluate the faster deployment over the benefits of new functionalities. Therefore, when the time-to-market decreases, it is possible to capture the value of the investment earlier as well. At Nokia, one of the biggest problems related to traditional software development was releasing large entities at a time, in slow release cycles (Nokia Intranet, training material). Figure 7 (see below) depicts the situation, where in the traditional development team (picture on the left side), the work-in-progress (WIP) and cycle times are bigger, which results in slower time-to-market and therefore slower return on investment (ROI) than in an agile development team (picture on the right side).

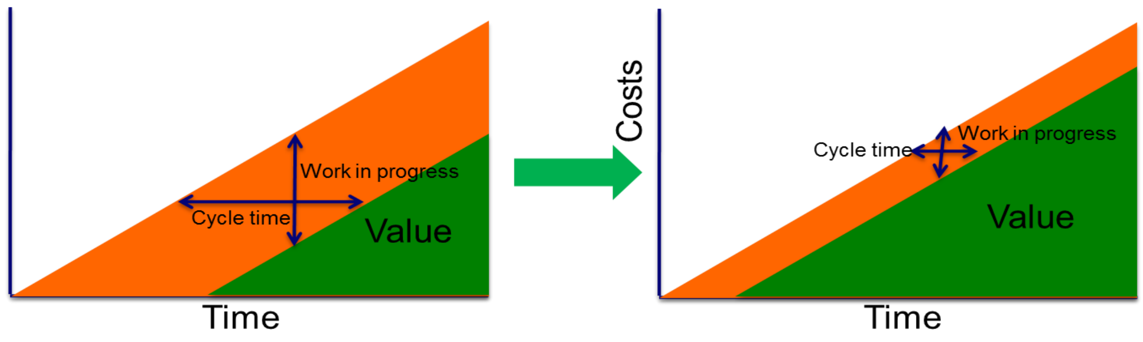

Figure 7: Smaller and more frequent releases. (Nokia Intranet, training material, August 28th , 2011)

At Nokia, the problems in many of the traditionally working teams are caused by planning the content ahead and keeping it fixed for a long time before implementation, late or no feedback and large amounts of work under development (WIP). This leads to long development cycles, and therefore releasing the products more slowly. The aim at Nokia is to react to these problems by applying agile philosophies of continuous and fast feedback loop, prioritizing and updating the contents of the products and features that are taken under development, and working in faster, frequent and smaller release cycles (Nokia Intranet, training material).

2.2.7.2 Increased quality

Schwaber et al. (2007) claim that agile processes lead to better quality in many ways. By applying some of the XP techniques, such as pair programming and test-driven development, the quality of the code should improve. At Nokia, some of my interviewees across the three teams on participating in my research, said they were applying the latter techniques as well as some other agile methods in their development work, alongside to Scrum.

Scwaber et al. (2007) also propose that agile teams produce software that behaves better at the system and code level, by starting testing and integration early in the development process. Further, close and frequent collaboration with the customer that is part of the agile philosophy, ensures meeting the needs of the business better and making changes to the product in a flexible way.

2.2.7.3 Reduced waste

The idea of reducing waste originates itself from Lean development, which draws from the research of Toyota's production system from the 1980s. In addition to gaining popularity in the field of manufacturing and production, lean philosophies have had an influence on agile software development practices as well. Reducing waste is one of the cornerstones of lean thinking, with the term “waste” meaning all the moments or actions that do not add value but consume resource. Waste is, for example, overburdened workers, bottlenecks, waiting, handoff, wishful thinking and information scatter, among others (Larman & Vodde, 2008). In software development, Poppendieck and Poppendieck (2003) claim that the main sources of waste come from partially done work, extra processes and features, task switching, waiting, motion and defects.

Schwaber et al. (2007) claim that agile teams are constantly looking for ways to improve their efficiency, i.e. reducing waste, because they continually gather productivity and quality metrics. That makes the team understand the effect that any changes on the development process have on these factors, and the assumption is that they want to improve them all the time. Also, close collaboration with the customer may sometimes prevent adding extra features to the product, and instead releasing the product at its “right size” (Schwaber et al., 2007). This is an opposing philosophy to traditional development, where the focus is on a high-quality development process, without considering the time or effort it takes to complete the given process. Whereas in agile and lean development, the aim is to optimize the effort and time used to complete each step in the process in relation to the value gained from the end product. Therefore, lean philosophy defines excess time and effort (in relation to the value of the end product) as a source of waste as well (M. Laanti, e-mail, May 3rd, 2012).

2.2.7.4 Better predictability

Schwaber et al. (2007) point out, that delivering working software in short iterations provides visibility into the progress of the project. The team keeps track on remaining work to finish the product on Burndown charts. The Burndown chart consists of user stories needed to finish the product, as well as estimated time to complete each one of them based on their complexity (user story point), which helps to keep track on how many story points is possible to deliver in each iteration.

In addition, agile teams are able to avoid surprises at the end of projects with testing, that Waterfall projects often face. Agile team makes sure that each functionality they deliver to the product is production-ready and tested. They also aim at tackling high-risk requirements in early iterations (Schwaber et al., 2007).

2.2.7.5 Better morale

Schwaber et al. (2007) claim that the teams that have successfully adopted agile processes report to have dramatically improved their morale. Schwaber et al. (2007) says that this stems primarily from the pride of delivering more and better software, but also from changing their roles into more collaborative in nature; Being in direct and close contact with the customer results in increased ownership of the team's work. Schwaber et al. (2007) also claim that working in a cross-functional team doing everything together, rather than working alone on a single part of a highly phased development process, is appreciated by software engineers. Also, making progress every day and delivering working software in each iteration motivates the team. (Schwaber et al., 2007).

3 Previous research on agility

The experience reports from the use of agile methods, often written by practitioners or consultants, have been predominantly positive. However, academic research on the subject is still scarce (Abrahamsson et al., 2002). Therefore there is little scientific proof that would support many of the claims made by the agile community (Mcbreen 2003, cited in Dybå & Dingsøyr, 2008). Laanti et al. (2011) suggest, that the lack of research on the subject stems from companies rarely having comparable data to explain the impacts of agile methods before and after their adoption. Also, the majority of existing research consists of qualitative case studies of individual projects, and therefore the results are not very applicable to wider contexts, and they also lack a broader scope (Laanti et al., 2011).

Until 2008, Dybå and Dingsøyr had identified altogether 1996 studies on the subject of agile software development. Only 36 of these studies were empirical research studies that they measured to be acceptable enough in terms of relevance, rigour and credibility. It seems as one of the research gaps in agile software development research is also, that majority of empirical studies of sufficient quality focus on evaluating a single process model, in most cases XP. According to Dybå & Dingsøyr's (2008) review, 25 (76%) of the studies were done in organizations using XP as their agile methdology. Studies on agility in general seem to be the second popular theme in literature with 15% of the studies, at least in academic context (Dybå & Dingsøyr, 2008). Despite the popularity of Scrum, the authors found only one case study prior to 2006 researching that method. After reviewing the existing literature on agile software development, Dybå and Dingsøyr (2008) consider that the state of research on agile software development being currently on a nascent state, which suggests a need for exploratory qualitative studies. My research will aim at contributing to this need in research, to some extent.

The results on the studies on XP might not be fully applicable to my study, since the target of this research is an organization which is officially applying Scrum as the main agile methodology. Even though XP and Scrum have some similarities, XP is more concentrated on practices while Scrum is primarily a project management tool (Abrahamsson et al., 2002). However, some of the software developers that were interviewed for this research in fact apply some of the XP practices voluntarily at their work, in addition to applying Scrum project management techniques. In this chapter, I will describe some of the existing studies related to agile development methods, and present the main findings of these studies based mostly on Dybå and Dingsøyr's (2008) review. I will also present some later studies, such as the quantitative research studies on agile development at Nokia of Laanti (2012) and Laanti et al. (2011), which both aim to understand the employees' perceptions of agile methods.

I chose to present only research findings, which were somehow related to the themes that my interviewees brought up during the interviews. The chapter begins with discussing the dilemma of introducing agile practices into large organizations (such as Nokia). After that, I will present the previous research aiming at finding out the impacts of agile software development on employee wellbeing and contentment, productivity, product quality, team dynamics and collaboration and customer satisfaction. Lastly, I will summarize the key findings based on my literature review.

3.1. Introducing agile into large organizations and teams

A common argument is that agile methodologies are not suitable for large organizational settings and in large teams (Cohen et al., 2004, cited in Dybå & Dingsøyr, 2008). However, some studies have suggested that agile can be very successful in large organizational settings as well, as long as the organizational environment supports agile adoption. Lindvall et al. (2004) claim that failure to adopt agile successfully in large organizations is related to their complexity and rigid organizational structure. Laanti et al. (2011) note that agile adopters are not often aware of what agility actually means, and how broad of a change is actually required. Also, in large organizational settings where complex software is produced, a holistic view on agility may be needed; it is not enough to focus on team and project-level dimensions only, as in a typical application of agile software methods (Kettunen & Laanti, 2008). Lindvall et al. (2004) note that the challenge for large organizations doesn't necessarily lie in applying agile practices into a project, but in integrating the agile project into its environment. There is a possibility for conflict and double work, when agile practices interact with traditional ones (Lindvall et al., 2004). Agile does not usually thrive in large teams, mostly because it makes face-to-face communication challenging (Lindvall et al., 2004).

The main conclusion of Svensson and Höst's (2005, cited in Dybå & Dingsøyr, 2008) study was that the introduction process of agile development was difficult due to the complexity of the organization. The authors studied a large software development company that introduced the XP development process to a pilot team, during a period of eight months. As a result, they advise companies that are introducing the XP process into the organization to set clear goals on what to introduce, and communicate this clearly with the rest of the company. Also, the companies should bear in mind, that the adoption process should not be underestimated, because it takes time and effort (Svensson & Höst, 2005, cited in Dybå & Dingsøyr, 2008).

Lindvall et al. (2004) studied the experiences of agile software development methods in large organizations. The authors conclude, that based on the experiences in large organizations, agile methods can be successful and especially in small, collocated teams (Lindvall et al., 2004). The authors based their analysis on the experiences shared by Software Experience Center (SEC) member companies, including ABB, Daimler-Chrysler, Motorola, and Nokia. The representatives of Nokia noted that small software development teams are more productive than large ones (Lindvall et al., 2004). The notion of a small size of the team is also present in the agile philosophy. For example in Scrum, the maximum suggested team size is 9 individuals. This is also in order to ensure effective face-to-face communication, which is also part of agile philosophy, defined in the Agile Manifesto (see Appendix B, p. 104). According to Boehm (2002), the scalability of the agile process seems to be one of the limiting factors in the use of agile methods. As the team size gets bigger, communication through documents becomes easier than explaining everything, most probably various times, to a large number of people by face-to-face communication.

Bahli and Zeid (2005) studied a Canadian organization shifting from a Waterfall process to XP, and got quite positive results in their study. The developers found XP easy to use already after one week of training. However, a development manager stated that the adoption process itself wasn't easy, since none of the developers had prior experience on XP and it will take more work to master it compeletely. The developers in this study also stated that they prefer XP over Waterfall, calling the latter an “unpleasant experience”, while XP was described to be “beneficial and a good move from the management”. However, the size of the organization is unknown, since the company wanted to remain anonymous (Bahli & Zeid, 2005).

A study by Robinson and Sharp (2004, cited in Dybå & Dingsøyr, 2008) found that XP has the ability to thrive in very different kinds of organizational settings. The study consisted of three case companies, and factors of organizational type, size and structure as well as physical and temporal settings varied significantly between these companies. The authors found out that despite the diversities in these factors, XP was working well (Robinson & Sharp, 2004, cited in Dybå & Dingsøyr, 2008).

Dybå and Dingsøyr (2008) conclude, that according to research they analyzed in their review, XP seems to be able to thrive in very different organizational settings, for example in organizations that varied from having a hierarchical structure to those that there was little or no central control. Also customer involvement and physical settings varied between the successful XP teams studied. In terms of the adoption process, XP is found to be difficult to introduce in a complex organization, but seemingly easy in other types of organizations. Also, many findings suggest that agile development methods are more suitable for small teams than in larger ones (Dybå & Dingsøyr, 2008).

3.2 Wellbeing and contentment

The aspiration in applying agile methodologies is to be more people-centric than in plan-driven models. The value statement in the Agile Manifesto *individuals over processes and tools* and the principle to organize work among *motivated individuals* indicate this goal (see Appendix A, p. 103). In addition, for example Schwaber et al. (2007) claim that teams that have applied agile practices successfully, have and increased morale.

Laanti et al. (2011) conducted a quantitative study on developer perceptions about agile methodologies at Nokia. The data was collected using a questionnaire, and more than a 1000 respondents from three different continents took part in the survey. Nine statements on agile development were presented (see Table 2, below), and a scale from 1 to 7 was used to collect the responses: (1 = totally disagree, 4 = neutral and 7 = totally agree). According to Laanti et al.'s (2011) research, responses to only one of the nine statements were below neutral (making work less hectic), while all the other statements received positive responses on average. The results revealed that most of the respondents agreed on all accounts with the generally claimed benefits of agile, which include higher satisfaction, feeling of effectiveness, increased quality and transparency, increased autonomy and happiness and earlier detection of defects (Laanti et al., 2011).

Statements	**Mean**	**Standard deviation**
1. Agile development increases the effectiveness of development	4.97	1.285
2. Agile development increases the quality of the product	4.70	1.362
3. Agile development increases the transparency of development	5.13	1.385
4. Agile development increases collaboration	5.04	1.330
5. Agile development makes work more fun	4.61	1.503
6. Agile development makes work more organized/planned	4.50	1.530
7. Agile development increases the autonomy of development teams	4.86	1.379
8. Agile development enables the earlier detection of bugs/errors/defects	4.77	1.416
9. Agile development makes work less hectic	3.64	1.522

Table 2: Opinions on the impact of agile development. (Laanti et al., 2011)

Laanti et al. (2011) also addressed the question of how much a person's background, i.e. the length of experience on agile and traditional methods, affects the perceptions on agile methodologies and overall satisfaction of them. The second motivation for the study was to figure out, how useful of a software development method agile is from the practitioner's point of view. Laanti et al. (2011) identified respondents falling into groups for and against agility. 65% of respondents would like to stay in the agile development mode, as 6% would like to go back to a traditional way of working instead. The rest of the respondents (27%) were neutral, mostly because some of the population answering the survey did not yet have experience on working in agile mode. Laanti (2012) also studied the wellbeing in agile teams applying mainly Scrum as their development methodology. The research was conducted at Nokia, where 466 employees responded to an online survey sent to 10 000 employees, which was analyzed by using quantitative methods. According to this study, 55% of respondents were happy to work in an agile way, and only 12% would like to go back to traditional way of working. Further, the study indicates that agile practices might relieve people's stress. 27% of respondents stated that their stress level is better because of agile, while 15% said that they feel worse. Rest of the respondents did not notice a direct link between work mode and stress level, or felt that it has not changed after the adoption of agile methodologies. Mann and Maurer's (2005) study on introducing Scrum into an organization (presented in more detail on p. 45) detected that developer perceptions on agile development were very positive as well. They perceived the Scrum process being very beneficial, and in a questionnaire, every developer inside a Scrum team would recommend using Scrum in the future. The closer relationship with the customer, as well as increased communication with them was seen as a very positive thing and giving confidence on being able to meet the customer's needs better. In addition, the developers were more satisfied with the quality of the product created than in previous projects.

There has been some research on whether a person's background or personality traits have an impact on perceptions about agile. Laanti et al. (2011) found out, that more than three years of experience on traditional methods affected somewhat negatively towards perceiving agile methods, and those people also found more difficult seeing the benefits of it. Korhonen (2010) also found a positive correlation between the engagement and contentment to agility and being able to perceive improved quality of the products created. Melnik and Maurer (2006) got similar results finding moderate positive correlation between the level of experience with agile methods and the overall job satisfaction. The study was a comparative survey of job satisfaction among 448 IT employees, using both agile and traditional development methods. The data was analyzed using quantitative methods and respondents were recruited via active newsgroups, mailing lists and wikis specialized in software engineering. The results also indicated that in there are twice as many members working in agile teams who are satisfied with their jobs, compared to people in non-agile teams. Statistically significant job satisfiers were ability to influence decisions that affect you, the opportunity to work on interesting projects and the relationships with the users (customers).

Young et al. (2005, cited in Dybå & Dingsøyr, 2008) studied the role of personality traits in agile software development teams. They used a technique called "repertory grid analysis" to identify good (and bad) characteristics for members in different roles in an XP development team. They defined the characteristics of a "good" XP team member as "analytical, with good interpersonal skills and a passion for extending his knowledge base and passing this on to others" (Young et al., 2005, cited in Dybå & Dingsøyr, 2008).

In regards to job satisfaction and wellbeing in agile development teams, studies have yielded mainly positive results. Mannaro et al. (2004) studied the job satisfaction among employees in software companies, which used XP as an agile methodology to those who didn't apply agile development methods. The research was conducted with a web-based questionnaire of 122 participants. The results were very complementary towards XP and not that much towards non-agile development. 95% of employees using XP were satisfied with the current development methods and wanted to keep using XP. In comparison, the satisfaction rate among employees that were not applying agile methodologies was only 40%. Mannaro et al. (2004) found out, that the employees who apply XP practices have greater job satisfaction in a way that they feel the job environment is more comfortable and that their productivity is higher compared to those using non-agile development process.

Syed-Abdullah et al. (2006) studied, if agile methods have any distinct effects on well-being amongst people in agile software development teams. The subjects of research were software engineering students, with a total population of 75 people, forming altogether 17 teams. Half of the teams developed software with an agile XP methodology, and the other half with a plan-driven methodology. The research methods used in the study were quantitative and qualitative in nature, consisting of participative observation, focus group interviews, close-ended questionnaires and simple statistical tests (Syed-Abdullah et al., 2006). Wellbeing was conceptualized in this study by examining factors of job related anxiety, depression, contentment and enthusiasm. These factors were measured by using a 12-items anxiety-contentment and depression-enthusiasm scales developed by Warr (1990, cited in Syed-Abdullah et al., 2006). The results of the study showed statistically no significant difference between agile and plan-driven development teams, except with the level of enthusiasm. The agile methods (XP) had a positive effect on the level of enthusiasm among developers, meaning the feelings of enthusiasm, optimism and cheerfulness towards the project being developed (Syed-Abdullah et al., 2006).

After doing a comprehensive research of existing research on agile development, Dybå and Dingsøyr (2008) conclude that developers are prevalently satisfied with agile methods, and the software developers in companies that use XP as an agile method have reported to be more satisfied with their job and with the products developed as well (Dybå & Dingsøyr, 2008). The authors continue, that whilst the effect on work practices and job satisfaction from the use of agile or traditional methods is not thorough, some studies suggest that the standardized work practices in agile development, lead to greater job satisfaction (Dybå & Dingsøyr, 2008). Person's background seems to have an impact on the perceptions and contentment on agile methodologies (Laanti et al., 2011; Melnik & Maurer, 2006). Contentment in agile seems to have an impact on being able to see the benefits of the methodologies as well (Korhonen 2010; Laanti et al., 2011).

3.3 Productivity and performance

The research on agile teams and productivity is still scarce, and therefore it is difficult to draw conclusions, whether agile increases productivity or not. Dybå and Dingsøyr (2008) could identify only four studies in their review, in which productivity of agile teams was measured in quantitative methods. However, these studies had all an inappropriate recruitment strategy, so unbiased comparison is not ensured (Dybå & Dingsøyr 2008).

Ilieva et al. (2004, cited in Dybå & Dingsøyr, 2008) compared the productivity of two similar projects, of which one used traditional software development methods and the other XP. They measured the productivity of three iterations in each project and the results indicated a 42% increase in productivity for the agile team. A case study by Layman et al. (2004) compared an old product release developed with traditional methods to a new release developed with agile methods. They discovered a 46% increase in productivity for the new agile release compared to the old one. However, in this case, the agile team members had more expertise in software engineering and project management experience than the traditional project team members (Dybå & Dingsøyr, 2008).

There are also a number of studies that suggest that the subjects of the research themselves believe that agile methods lead to increasing productivity (Dybå & Dingsøyr, 2008). According to Melnik and Maurer’s (2005, cited in Dybå & Dingsøyr 2008) study of student perceptions on agile, 78% of respondents believed strongly that using XP improves productivity in small teams. This was also the case in Laanti’s (2012) research, where 64% out of 466 respondents felt that their performance had increased after taking agile methods into use. The main finding of the study was, that developing software at a sustainable pace leads to better performance. Sustainable pace in development is one of the main principles of agile: *Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely* (see Appendix B, p. 104). In practice, this means working in a constant flow and focusing on top priorities. This principle is opposing the Waterfall-like development process where working overtime near deadline and resting after it is over, is usual (Laanti, 2012).

Contrary to these findings, a study by Wellington et al. (2005) analyzing XP in the use of students, reported a 44% decrease in productivity compared to a plan-driven team. Furthermore, Svensson and Höst (2005, cited in Dybå & Dingsøyr, 2008) found no evidence of change in productivity after agile processes were introduced to a company. Some studies indicate that some agile practices require skilled individuals, in order for them to increase productivity. Melnik and Maurer (2002, cited in Dybå & Dingsøyr, 2008) studied student perceptions on XP. While most of the students found pair programming to be helpful, one of the students noted that the skill differences between him and his pair were quite significant, which resulted in decreased productivity. Also, test-driven development was found to be difficult by many students. The authors believe that this is because writing the tests before designing, forces the students to make design decisions early (Melnik and Maurer, 2002 cited in Dybå & Dingsøyr, 2008). A study by Tessem (2003, cited in Dybå & Dingsøyr, 2008) of the XP process, indicated that it takes time to learn correct estimations. At the beginning of the project, only one third of the user story estimations were correct. Also, several study participants felt that there was not enough discussion on design and architecture throughout the project (Tessem 2003, cited in Dybå & Dingsøyr, 2008).

Dybå and Dingsøyr (2008) conclude these studies related to the productivity of agile versus traditional teams, that three out of four show that using XP results in increased productivity in terms of LOC/h (lines of code per hour). Also developer's perceptions on the impacts agile has on productivity and performance seem to be mainly positive.

3.4 Product quality

A common statement made by the advocates of agile development is, that agile processes lead to better quality in many ways. The arguments supporting this are often related to the continuous testing and integration of the code, instead of doing all of that at the end of the project like in plan-driven methodologies (Nerur et al., 2005; Schwaber et al., 2007; Leffingwell, 2011). A comparative study by Huo et al. (2004) of quality assurance in a Waterfall process compared to and agile process suggests, that the frequency of quality assurance (QA) practices are higher in agile processes. The conclusion could be drawn, therefore, that as code is tested more often, it would lead to better quality as well.

Layman et al. (2004) conducted a longitudinal case study at an airline company inside one software development team, and measured the product quality before and after the XP methodology was adopted. After the adoption process, a 65% improvement in prerelease quality and a 35% improvement in post-release quality were reported. In a comparison of two similar projects, Ilieva et al. (2004, cited in Dybå & Dingsøyr, 2008) found 13% fewer defects in the product (reported by the customer or by the quality assurance team) in an XP project that in a non-XP project. Wellington et al. (2005) conducted a case study among university students, majoring in software engineering. They offered two courses, one based on plan-driven methodology and one on agile development, XP in particular. They found out, that compared to a traditional team, the XP team's code scored consistently better on the quality metrics and that the quality of the code delivered was significantly greater than that of a traditional team. However, in the same study, both teams perceived that the plan-driven team had created a better user interface in the product. Macias et al. (2003, cited in Dybå & Dingsøyr, 2008) measured the quality of products developed by 10 plan-driven and 10 XP teams and they found no significant difference in results.

In addition to proving the increased quality in quantitative metrics, according to some studies, developers perceive that quality has improved after the adoption of agile methods. Laanti et al., (2011) conducted an extensive quantitative survey study across Nokia, with over 1000 respondents and a response rate of 33%. The aim of the study was to get an overview on practitioner's perceptions on agile development. The study reports positive results on many accounts (see Table 2, p. 36), including increased product quality (Laanti et al., 2011).

Mannaro et al. (2004) conducted a web-based questionnaire of 122 respondents, among software companies that used XP method and in companies that did not apply agile methods. The study reports that 76% of the people, who had applied XP in their work, believed that XP has improved the quality of code (Mannaro et al., 2004).

Korhonen (2010) studied, if people's perceptions impacts of the agile practices having on code quality were realistic, and the results were quite surprising. The survey was conducted in a large, globally distributed telecommunications organization, migrating from traditional (Waterfall) development mode into agile. A survey was conducted after 6 months of agile adoption, and it received 78 responses. Only 25% of agile adopters believed that quality had increased, in reality, the quality had improved and was visible in defect data. Further, the study revealed that a realistic perception of the positive changes in the defect data coincided with positive emotional engagement in agile transformation.

To summarize the main findings, most studies report increased code quality when agile methods are used. In general, people also perceive that agile has increased quality, at least to some extent. Whether their perceptions reflect the reality, is questionable. Also, the size of the end product does not seem to have an effect on the results, big and small entities have reported to be better in quality after agile was adopted.

3.5 Team dynamics and collaboration

As stated earlier, the principles of the Agile Manifesto encourage working in teams and having frequent feedback and other types of (face-to-face) communication within the team and with customers. (See Appendix B, the principles of Agile Manifesto 4 and 6, p. 104). Successful agile teams should also be self-organizing and empowered (Schwaber, 2007; Leffingwell, 2011) Schwaber et al. (2007) claim that the collaborative nature of agile teams and sharing responsibilities increases people's morale and job satisfaction.

Robinson and Sharp's (2004) study seems strengthens the proof for the previous statements. The authors drew a conclusion from their study of three case organizations using XP, that agile development teams have faith in their own abilities, show respect and responsibility, establish trust and preserve the quality of working life. Further, Laanti's (2012) quantitative research study at Nokia indicated that the majority (71%) of respondents working in agile teams feel their team is empowered.

Whitworth & Biddle's (2007) study of the motivation and cohesion in agile teams brought up positive results as well. The research was conducted by analyzing qualitative data, based on semi-structured interviews with 22 participants working in different agile teams. All but two had experience in working in non-agile teams before agile. The participants were recruited through networking with members of the agile software development community. They operated in a variety of roles, including developers, designers, project managers, coaches and specialists. The researchers examined the results by trying to point out characteristics that are related to team cohesion. They found out, that the main value of agile methods in supporting team cohesion and motivation was a collective team culture. As a result of the agile planning in the beginning of a sprint, as well as the iterative nature of agile software development, the process was seen as stabile but at the same time complex, creative and social problem-solving activity (Whitworth & Biddle 2007). This kind of agile development process was seen to support and even require the development of collective culture and team-wide communication, including effective feedback mechanisms in order for it to work properly (Whitworth & Biddle 2007). The results also showed that self-efficacy was experienced highly by agile team members. Self-efficacy means things such as effort or skill put into the work and controllability or modifiability of one's environment (Eby & Dobbins 1997, cited in Whitworth & Biddle 2007). Agile practices, such as daily meetings, feedback, negotiation of a flexible plan, continuous integration and testing were seen to increase perceptions of self-efficacy and control within the team (Whitworth & Biddle 2007). Being involved and aware of the project activities supported the feelings of self-efficacy, whilst the team members that were not clued to day-to-day activities experienced discomfort, dissatisfaction and the absence of self-efficacy (Whitworth & Biddle 2007). Therefore, although a team can be seemingly agile, by failing to involve people in close communication can remove the cohesion of agile teams.

Despite these positive results, Wellington et al., (2005) study of team cohesion in XP and plan-driven teams yielded equal or higher scores for every aspect of cohesion for the non-agile teams. The study was conducted among students of two engineering courses, one teaching XP and the other a plan-driven methodology. Both of the courses formed a team of 14 to 16 students and they were given the same problem statement to solve with a given methods. All in all, this study of team cohesion did not find any improvement of cohesion in the XP team (Wellington et al., 2005).

From their case study of three large companies, Karlström and Runeson (2005, cited in Dybå & Dingsøyr, 2008) found out that XP teams experienced improved communication within the team, but that they were perceived by other teams as more isolated. Bahli and Zeid (2005) examined knowledge sharing in XP team and a traditional team, and found out that the creation of tacit knowledge was improved as a result of frequent contacts and communication.. Lindvall et al., (2004) suggest that adding support for cross-team communication presents an important need for improvement in large organizations, particularly prominent at Nokia. Large organizations often distribute teams across several physical locations, which can bring up challenges in close and effective communication, which is a crucial part of a successful agile team (Lindvall et al,. 2004).

Studies of agile team dynamics, characteristics and communication indicate that the successful agile teams are able to balance a high level of individual autonomy with a high level of team autonomy and corporate responsibility. Team members in successful agile teams have faith in their abilities and preserve the quality of their working lives. Also good interpersonal skills and trust were found to be crucial factors for a well-functioning agile team (Dybå & Dingsøyr, 2008). An agile teams usually experiences improved communication, but might be isolated from other development teams.

3.6 Customer satisfaction

One of the main purposes of agile development is being able to meet customer needs better, as stated on the Agile Manifesto: *Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.* Also the value statement *Customer collaboration over contract negotiation* in the Agile Manifesto encourages involving the customer as an active part in the development process (see Appendices A & B, pp. 103-104).

Ceschi et al. (2005) conducted a survey study for 20 project managers in software companies that were using plan-driven and agile methods. They found out, that agile methods improved the management of the development process as well as relationships with the customer. The results indicate, that agility creates increased customer contact, which makes a high quality link between the development team and the customer. Agile companies organize their work in more releases and pay more attention to activity planning by prioritizing essential work in each iteration. Managers in agile companies are more satisfied with the way their projects are organized than those in plan-driven companies (Ceschi et al., 2005).

Mann and Maurer (2005) did a case study, where they assessed the variables of customer satisfaction and working overtime before and after Scrum was introduced into a development team. The quantitative results of measuring working overtime indicated that after the adoption of Scrum, working overtime had decreased significantly. This means that the team was able to work fewer hours and most probably at a sustainable pace. At the same time, customer satisfaction had increased as well. All of the three customers said they would recommend using Scrum in the future, and were happy to be part of the development process. The customers liked the fact that they were involved in Scrum meetings and found them to be beneficial. They also felt that their respect towards the developers raised as well. Before Scrum, they did not have an active role throughout the process, and sometimes were not satisfied with what was produced.

Dagnino et al. (2004, cited in Dybå & Dingsøyr, 2008) compared the use of an agile approach to a traditional one in two different development projects. They noticed a higher customer satisfaction with the agile team, because it was able to demonstrate business value more quickly and more often than a traditional team. Also, the customer was giving feedback throughout the agile development process, and the agile team was able to incorporate changes more easily because of the incremental development in short cycles.

Sillitti et al. (2005) also found similar results regarding customer collaboration and relationships in agile companies. They interviewed eight project managers from document-driven companies and eight working in agile ones. Sillitti et al. (2005) draw conclusions of the results, that agile companies are more customer-centric and flexible than document-driven ones. Agile companies also seem to have a better relationship with the customer than document-driven companies (Sillitti et al. 2005).

However, a case study by Martin et al. (2004, cited in Dybå & Dingsøyr, 2008) of three XP projects with on-site customers indicated that in all three cases the customers were under stress. They had to commit working long hours on the project even though they were supported by various technical advisors and other personnel inside the company.

Studies on customer perceptions of agile are positive and customers value the opportunities for feedback and responding to changes (Dybå & Dingsøyr, 2008). According to studies, customers appreciate the fact that the development team is able to lay out the value of the product in early development, by making it visible (Dagnino et al., 2004, cited in Dybå & Dingsøyr, 2008). Furthermore, several studies indicate that agile teams are more customer-centric and have better relationships to their customers than traditional ones. This has obviously positive impact on customer satisfaction (Ceschi et al., 2005; Sillitti et al., 2005). However, involving the customer actively into the development process require commitment and resources to the project from the customer's side as well. This might increase stress among the customers involved in the process (Martin et al., 2004, cited in Dybå & Dingsøyr, 2008)

3.7 Summary of research – benefits and shortcomings of agile development

Based on the literature discussed in this chapter, I will summarize the main findings related to agile development. Most of the studies were presented in Dybå and Dingsøyr's (2008) systematic review, and in addition I chose some additional studies to be presented, such as (Korhonen 2010; Laanti 2012; Laanti et al., 2011; Syed-Abdullah et al., 2006). Dybå and Dingsøyr (2008) remind that overall, the rigor of the studies presented in the systematic review are quite low, and in many cases, the situation before the introduction on agile has been unclear. Furthermore, most of the studies were case studies and they were predominantly focused on XP as an agile methodology, which makes the results which makes the results hardly applicable to other contexts or comparable to other agile methodologies.

The studies that address the introduction and adoption of agile methods do not provide a unified understanding. In regards to organizational environment, a benefit of XP was that it thrived in radically different environments; in organizations that varied from having a hierarchical structure to little or no central control. However, some studies indicate that XP was found to be difficult to introduce in a complex organization, but rather easy in other types of organizations. Also, some findings that suggest that agile development methods are more suitable for small teams than for larger projects (Lindvall et al., 2004).

Developers are in most cases satisfied with agile methods. Some studies indicate, that companies that apply XP have reported that their employees are more satisfied with their job and with the product developed. The work environment in agile projects is perceived as comfortable, and can be characterized as respectful, trustful, and help preserving quality of working life (Mannaro et al., 2004).

With respect to the productivity and performance of agile teams, Dybå and Dingsøyr (2008) reported that three out of four studies XP teams had an increased productivity in terms of LOC/h, compared to traditional teams. Findings from several studies also indicate that team members believe that their productivity has increased after adopting agile practices. Some studies suggest that agile team members have to be highly qualified, in order for the development work to be effective (e.g. Mann & Maurer, 2002, cited in Dybå & Dingsøyr,2008)

Regarding product quality in agile teams, most studies report increased code quality (Dybå & Dingsøyr, 2008). However, some studies have reported that the user interface of the product has been better as a result from traditional projects. Some developers perceive that the focus on architecture and design is not sufficient in agile development (Tessem 2002, cited in Dybå & Dingsøyr, 2008), which might be related to this.

Studies of XP indicate that successful agile teams are able to balance a high level of individual autonomy with a high level of team autonomy, and taking on notable responsibilities as a team. In addition, good interpersonal skills and trust among team members were seen as crucial factors for successful XP teams (Dybå & Dingsøyr, 2008). Agile methods facilitate better communication and feedback compared to traditional teams, which results in effective knowledge transfer. However, large team size and geographical distribution of teams were challenging effective communication (Lindvall et al., 2004).

Agile methods propose to have the customer on-site or at least taking actively part in the development process. This is perceived as valuable by developers as they can get frequent feedback and feel that they are able to meet customer needs better. The customers appreciate being part of the process as this provides them with control over projects. Frequent releases from the development team provide the customers with an opportunity to able to see actual results and progress to the project within given time intervals. An additional benefit is the regular feedback on development progress provided to customers (Ilieva et al., 2004, cited in Dybå & Dingsøyr, 2008).The downside for on-site customers is that they have to commit for the whole development process which requires their commitment and puts them under stress (Martin et al,. 2004, cited in Dybå & Dingsøyr, 2008)

4 Methodology

In this chapter, I will describe the qualitative methods used for obtaining the research data. The data gathered for this case study consists of 16 thematic interviews among agile team members, informal discussions and interviews with stakeholders to agile projects as well as Nokia's internal training material. The latter was acquired from Nokia's Intranet, in the form of PowerPoint presentations on issues related to traditional, agile project management. First, I will describe the case study approach to conducting a research, since the study was conducted inside one company. Second, I will present the methodology used for gathering primary data, which is thematic interviews. Third, I will describe the sample and informants taking part in this study, as well as the interviewee recruitment techniques used. Lastly, the validity and credibility of thematic interviews as a qualitative research methodology are evaluated.

4.1 Case study approach

According to Eriksson & Kovalainen (2008), case study research should be understood more as a research strategy, rather than a method; despite its qualitative nature, case studies can also produce quantitative data. However, case studies are not meant to produce statistical generalizations, such as experimental, quantitative and deductive research aims (Ghauri & Gronhaug, 2005, cited in Eriksson & Kovalainen 2008).

The main purpose of case studies is to investigate the case in relation to its organizational and social environment. Methodologically classic case studies are connected to the interpretative, ethnographic and field-research studies (Dyer & Wilkins, 1991, cited in Eriksson & Kovalainen, 2008). Case study research approach is chosen as a research strategy, when the purpose is addressing complex organizational, managerial or other business issues, and when the emphasis is trying to achieve a holistic and detailed understanding on the issue in question (Eriksson & Kovalainen, 2008).

Stoecker (1991, cited in Eriksson & Kovalainen, 2008) makes a distinction between intensive (classic) and extensive case study research. The key differences between these two are that intensive research focuses on finding out as much as possible on a small number of cases, while the extensive design aims at finding out common patterns and properties across cases.

Intensive case study draws on the qualitative and ethnographic research traditions, and it aims at interpretation and understanding of the case and elaborating cultural meanings and sense-making processes in a specific context (inside a certain company for example). It aims at developing an understanding of the case "from the inside" by listening to people that are involved in the case. The researcher takes the role of an interpreter of the multifaceted and rich phenomena, and creates a narrative, a story out of the results (Eriksson & Kovalainen, 2008).

Extensive case study research on the other hand, relies more on quantitative and positivist research, by aiming at testing and developing theories. The main aim is therefore not on detailed prescriptions of the "real life" cases, but they are rather seen as instruments in exploring a certain business-related phenomena and developing theoretical propositions that could be tested and generalized to other contexts or to as a part of a theory (Eriksson & Kovalainen, 2008).

My research falls somewhere in between intensive and extensive case study research. While the aim is to understand the experiences of individuals (cases) quite thoroughly and interpreting them (intensive case study approach), the research also aims at comparing the results and mapping out similar or contradicting patterns based on existing research on agile software development, as well as on the claims that the creators of agile methodologies have presented (extensive case study approach).

4.2 Thematic interviews

In order to address the research question of understanding people's perceptions on agile development and agility possibly having an impact on certain themes (workload, stress, empowerment, etc.), I chose to organize conduct thematic interviews.

Typically, the qualitative interviews are categorized under three structural types based on their predetermined levels of organization and construction: structured, semi-structured, and unstructured. With structured interviews the researcher decides what questions will be asked and the order in which the interviewer will ask them (Goodman, 2001), and they are usually conducted in the form of a survey or a questionnaire (Tuomi & Sarajärvi, 2009). The more structured the interview, the closer the method gets to a quantitative study, providing a narrow platform for participants to respond in their own words (York, 1998, cited in Goodman, 2001). When interviews are highly structured, flexibility and spontaneity are compromised to collect systematic data across variables from each respondent (Patton, 1990, cited in Goodman, 2001).

Unstructured (open-ended) interviews are at the other end of the continuum, where the interviewer does not use a pre-established interview schedule and the interviewee usually leads the conversation (Goodman, 2001). Thematic interviews are semi-structured in their nature, which means that the idea is to proceed according to certain predetermined themes, topics and issues, but still having the possibility to vary the wording and order of questions during each interview. The advantage is that the research data produced is somewhat systematic, but at the same time people can express themselves freely (Eriksson & Kovalainen, 2008). Goodman (2001) notes, that people are able to express complex feelings and thoughts through language, which would not be possible to understand through observation or questionnaires. This is an interesting notion regarding my research as well, since I aim at not only getting an insight on people's perceptions and experiences on a general level, but also to understand what are the underlying factors that lead to a certain kind of thinking. The format of my interviews are semi-structured in a way, that I used mostly pre-planned questions during the interview, but altered the wording according to the situation and asked additional and probing questions in order for the interview to become more conversational and to gain more comprehensive data.

The first half of the interview questions are more open and less guided than at the end. The idea was to let the people tell in their own words what they thought was important in their opinion. The second half of questions, starting from question 12, (see Appendix C, pp. 105-106) are more structured and standardized, as I aimed at getting information on certain themes, such as team dynamics, stress level, workload, etc., if they hadn't been covered during the open-ended questions yet. At the end of the interview I asked the interviewees to sum up their feelings on agile development, and to add any essential points regarding their perceptions on agile development.

Silverman (2001, cited in Eriksson & Kovalainen, 2008) divides the types of different interviews into three categories; positivist, emotionalist and constructionist, and interview questions should be formulated according to these typologies. The emotionalist approach, which I use in my research, focuses on understanding participants' authentic experiences on a given issue. In other words, the interview questions would not focus on hard facts, but on people's perceptions, conceptions, understandings, viewpoints and emotions (Eriksson & Kovalainen, 2008). Open-ended interview questions encourage to more speech than closed ones, and they give the participant more control over what is talked about (Eriksson & Kovalainen, 2008). Many open-ended questions begin with "why" or "how" and they need to be worded so that respondents are not able to answer just "yes" or "no". After the first four background questions in my interview (see Appendix C, pp. 105-106), most of the questions are formulated in this way.

Seeking understanding through active listening skills is an important part of interviewing. The interviewer should try to seek clarity on what is being said throughout the interview, and reflect upon what the speaker is saying (Guion et al., 2009). I used this technique during the whole interview by asking additional questions, for example "why do you feel this way?" in order to seek clarity and understand the holistically people's experiences.

When conducting a qualitative interview, it is advisable that the responses of informants are audio-recorded and transcribed (Tuomi & Sarajärvi, 2009). I recorded all the interviews, and transcribed them from word to word, which resulted in more than a hundred pages of data to be analyzed.

4.3 Sample

Thematic interviews in qualitative studies often involve nonprobability sampling as a recruitment strategy for the interviewees, since generalizing findings beyond study participants is not the objective. In probability sampling techniques, a probability for the sample to represent a larger population can be calculated. Nonprobability sampling does not meet this criterion, and the sample may or may not represent a larger population. However, determining the important characteristics of informants is important and the sample should be formed according to these (Goodman, 2001).

In my study, I wanted to get a varied scale of opinions and standpoints, and especially people who have opposing viewpoints on agile development. Originally, I used the accidental sampling strategy, by just sending invitations to interviews by picking out names randomly. However, I did consider having people from various roles while picking out the interviewees. Since the opinions were predominantly positive after the first round of interviews, I acquired one more interview by “snowball sampling strategy”, which means that the researcher finds informants through referrals of other informants (Goodman, 2001). I did this by asking the interviewees, if they know anyone who is against agile within these teams. The informants did not know any of these kinds of people, apart from team 2. The informant in team 2 pointed out two people, who supposedly were a bit resistant towards agile, of which I interviewed one person (Developer 4, team 2). In addition, I wanted to see if peoples’ opinions varied depending on their background on agile, i.e. how much experience they had on applying traditional as well as agile software development methods. Therefore I chose to interview people from teams that had just adopted agile (teams 1 and 2), and from a more mature agile team (team 3).

The sample of my study consisted of 16 employees across three different Scrum teams. Altogether, people in the following roles were interviewed: 1 Scrum Master/Product Owner, 1 Product Owner, 1 Scrum Master, 3 Lead Developers and 10 Developers (Scrum team members).

In teams 1 and 2, Scrum had been adopted only recently, and people had been working in agile mode for less than a year, aside from the test interviewee (Developer 4, team 2), who was interviewed only a few weeks after the agile transformation process was completed. None of the team members in teams 1 and 2, apart from developers no 1 and 2 (team 1) had had any previous experience on working in an agile mode before this. Team 3 consisted of people from varying backgrounds, of which all members had experience of agile varying from 1-5 years, and the average being 3,5 years of experience of working in an agile mode.

The members in my sample from teams 1 and 2 had in general a long history at Nokia, on average a work experience of 10 years in the company, varying from 1,5 to15 years. In team 3 of my sample, the people had generally been working a bit less at Nokia, with the average being 6,5 years, and the experience varying from 2 to12 years inside the company.

The sample consisted of both sexes, minority of females. In order to protect people’s anonymity, every team member will be referred as “he”, and people’s opinions will be implied as their assigned number and role. The core teams of 1 and 2 worked on the same site, and they had the same Product Owner, because the products they created were related to each other. There were also external and part-time workers, who were not taken to the sample. Team core members of team 3 worked on two different sites, both in Finland. Team 3 could be considered as two different teams, as they were developing different functionalities of the same product, depending on where a person was were located.

Here is the sample according to people’s roles and the assigned numbers:

<u>Team 1</u>

Developer 1, Developer 2, Product Owner 1, Scrum Master 1, Lead Developer 1

<u>Team 2</u>

Developer 3, Developer 4 (test interviewee), Developer 5, Developer 6, Product Owner 1

Team 3

Developer 7, Developer 8, Developer 9, Developer 10, Scrum Master/Product Owner 1, Lead Developer 2, Lead Developer 3.

The interviews were conducted in January of 2012, apart from the test interview (Developer 4, team 2), which was conducted already in September of 2011. The interviews lasted between 30-60 minutes, with the average length of an interview being approximately 40 minutes. The interviews were conducted on three different office sites, all located in different cities in Finland. Two of the interviews were conducted via phone. All interviews were recorded on a digital voice recorder, or on the mobile phone's recorder application. Most of the quotations in this report have been translated from Finnish to English, as the majority of interviews were conducted in this language.

In addition to acquiring data from the interviews of software development teams, I had informal interviews and conversations with other stakeholders, who were somehow connected to the projects of pursuing agile development at Nokia or to the teams that were the focus of my thesis. The communication took part via face-to-face, phone, and e-mails. The people I had communication with, and who I have also used as a source in this thesis in explaining the situational background information on agile and Nokia, were Maarit Laanti (Senior Manager, Agile Coach and researcher at Nokia), Software Quality Manager who had an insight on the situation in the organization that teams 1 and 2 operate in, and an External Consultant who was part of executing the change process in teams 1 and 2. The latter two will remain anonymous in order to protect the anonymity of the people within teams 1 and 2.

I also had access to Nokia's intranet, where I attained textual and visual material (PowerPoint presentations) on agile development, particularly concerning the agile development process of teams 1 and 2. I used this material in order to get some background data on the agile development at Nokia, as well as on understanding the motivational factors for initiating the change process.

4.4 Validity and credibility

The objective of thematic interviewing for qualitative research is to bring light on experiences and create hypotheses, and not to test them. Therefore the notion of validity as applied in quantitative research has got little to do with qualitative research practices (Goodman, 2001). The aim is not to create generalizable study findings or external validity, but to transfer the knowledge to similar contexts (Reese et al., 1999, cited in Goodman, 2001). I apply this philosophy in my study, by trying to complement previous research findings on agile software development methods at Nokia in particular.

Reese et al. (1999, cited in Goodman, 2001) note that qualitative study aims for rigorous trustworthiness and credibility. The most obvious threat to achieving this is the methodology used to collect the data. Thematic interviews are subject to distortion for mainly two reasons. The interviewees might be reluctant to reveal information, or the interviewer is not able to be neutral and instead interjects his/her own perspective (Goodman, 2001). The best solution to avoid the latter one is for the interviewer to use self-monitoring activities during the interview. These include open-ended, neutral and clear questions. I have applied this practice while formulating the interview questions so that they don't include "either-or" choices, such as "has your stress level increased or decreased", instead I am asking: "Does working in an agile mode affect your stress level and in what way?" (see Appendix C, question 15, p. 106). It is also important to build a relationship and trust with the informants, in order for them to feel free to reveal personal information about themselves. I tried to build a trusting relationship with the interviewees by assuring them that they were able to remain anonymous, and could feel free to speak their mind.

In addition, the interviewing situation itself is a bit artificial in a way, that the informants might not be able to express themselves naturally because of the situation being intimidating or weird (Vuorela, 2005). There is also a possibility that the informants want to present themselves in a "socially acceptable" manner, which prevents from true feelings being revealed (Nielsen 1993, cited in Vuorela, 2005). The interviewer's voice, body language and facial expressions all have an effect on the formation of the relationship (Hirsijärvi & Hurme, 2001), and it is the responsibility of the interviewer to make the situation as comfortable as possible (Vuorela, 2005).

To summarize, the researcher who conducts the interviews him/herself is more involved in the research process than when using other business research methods (Vuorela, 2005). Therefore the interview data can be seen (more or less) as jointly constructed by the interviewee and the interviewer (Rapley, 2001). In addition, the researcher makes choices on what factors she/he wants to extract from the interview data and present in the report (Tuomi & Sarajärvi, 2009). In this report, I wanted to present all the perspectives that I found to be essential, relating to agile development. Even though the interviewees seemed to have a very positive attitude towards it, I wanted to lay out the controversial and opposing viewpoints and issues in this report as well. The viewpoints I enhance in this report, might not represent the opinions of the majority of the sample. The idea of this choice was to provide some kind of understanding on why certain people might not enjoy working in agile mode, or what are the shortcomings that people perceive in agile development.

5 Findings

In this part of the thesis, I will report the key themes and viewpoints related to agile development and Scrum, which emerged from the interviews. I will start with providing with some background data related to previous mode of operation in my sample teams, as told by stakeholders to the transformation projects and the agile team members (chapter 5.1). After presenting the baseline situation, I will elaborate on people's perceptions and experiences on the agile mode of operation. Many of the statements made by my interviewees were comparisons to the previous plan-driven mode of operation, which will become evident in the quotations I chose to base my findings on. As previous studies have pointed out (e.g. Laanti et al, 2011; Melnik & Maurer, 2006) that a person's attitude towards agile methodologies seems to change for the more positive as time passes, I wanted to take this into consideration as well. Therefore I started presenting the findings by describing people's attitudes in the beginning of the change process, and how they have developed over time (chapter 5.2 and 5.7). After this, I will present the key findings related to themes of performance, quality, workload, stress level, team dynamics, communication, empowerment, trusting, wellbeing and contentment on agile methodologies. In relation to these themes, I have tried to point out the distinctive differences in opinions between the team members which are newbies in agile (teams 1 and 2) to the team members with more experience working in agile mode (team 3). Lastly, I will summarize the main benefits and shortcomings of agile, as perceived by the informants in agile teams.

5.1 Baseline situation

Teams 1 and 2 went through with the transformation process to agile only recently, starting from March 2011 and ending at December 2011. The External Consultant I interviewed for this study, prepared a "current state analysis" regarding teams 1 and 2 in 2010, when Waterfall was still applied. In addition, I interviewed a Software Quality

Manager, who had an insight on the way things were organized inside the organizational unit that teams 1 and 2 are a part of.

Team 3 had gone through the agile transformation process already in 2006, so it was not that simple to gain insight on the baseline situation in that case. However, some of my interviewees had been in the team in question long enough to explain what the situation was before the transformation process. In addition, there were some new team members to team 3, who were able to tell about the transformation process inside another team.

Overall, it did not seem to make a lot of difference on which traditional team the person had been a part of previously. No matter what the background, the same themes arose in the conversations across all the teams, in relation to practicing plan-driven methodologies. The shortcomings in the Waterfall model seemed to be very similar to those often explained in literature, such as belated deadlines, bottlenecks in the development process due to specialized roles. Also, having to complete one stage before the other one can begin, ineffective prioritizing of requirements and estimation, not being able to concentrate on actual development because of additional things such as meetings, and general lack of order and even chaos in work practices was experienced.

The previous work mode across Nokia had been predominantly a Waterfall-like process and this was the case also in my sample teams. Across software development teams, work was organized according to projects as opposed to sprints which contain tasks from different entities. This was neither desirable nor effective in terms for an employee having to switch from tasks and meetings several times a day, all concerning a different project. The projects also had a priority order and a budget. The resources for each project were allocated separately according to their perceived importance. Therefore the projects were kind of competing with each other in significance, and often projects that were ranked lower in significance were terminated if priorities changed or there needed to be cutbacks. In addition, prioritization of tasks was made based upon the priority of the project and not the task itself. Also, the priorities evaluation and estimation for given tasks were fallen on a single developers' shoulders and not prioritized strategically by a Product team, such as in Scrum (External Consultant, interview, August 23rd, 2011). All the required changes were responded to very slowly, because first you had to set up a project. Also projects and their milestones were lacking behind schedule (Software Quality Manager, interview, September 30th, 2011).

The previous work mode was also a very Project Manager –driven one. There was one Project Manager in each team, who was in charge of the success of the projects instead of dividing the responsibility between team members. The Project Manager would also assign the tasks to team members. In addition, the work itself was more focused on people conducting their own, individual projects and domains, instead of working on the same entities (Lead Developer 3, team 3).

These types of characteristics or better to say, shortcomings are quite ordinary in organizations that organize their work according to the Waterfall model.

5.2 Early experiences and changing attitudes

The early feelings people had about agile methods, when they heard that they are going to be implemented to their team, were pretty mixed. Employees in team 1 seemed to be quite enthusiastic and curious about it, while team 2 experienced high levels of change resistance. People from team 3 came from different team backgrounds, and had been working in agile for quite some time, so their initial feelings varied as well. I noticed that it seems like the team often forms quite a uniform opinion about agile, either good or bad, and there are a lot of factors that can affect these feelings. If there were some difficulties in the implementation process itself, it seemed to be straightforwardly related to questioning the benefits of agile methodologies as well. Also, senior developers who had been working in traditional way for years seemed to be more resistant towards the change and it seemed to take time and effort to "convince" them about the benefits of working in an agile mode and get used to it.

The people who liked the idea of agile software development methods from the beginning, and were curious about them, had not experienced a change in their attitude since time had passed but still had a very positive attitude. This is how they describe their feelings:

"I have been enthusiastic about this from the beginning; I did not resist the change" (Developer 2, team 1).

“My feelings were pretty good, because my previous workplace was a mess (other than Nokia) and I liked the idea of keeping things organized” (Developer 2, team 3)

Some of the team members had quite a negative or at least skeptical approach to agility in the beginning. All of the interviewees that were skeptic and might have been resistant towards the change in the beginning said that their attitudes had changed for the better since time had passed. This is how some people felt:

”At first I thought, here we go again, yet another way of messing up our patterns. Since then my feelings have changed, this was a step towards the right direction” (Developer 7, team 3).

The only one whose attitudinal change I was not able to question, was the test interviewee, who I interviewed just after agile was wholly implemented in his team (team 2). He resigned shortly after that, so I was not able to get back to him.

The attitudinal change for the better among the skeptics, was often due to learning more about the methodology and participating in training sessions, getting responsibilities and getting involved in the Scrum team (being a substitute Scrum Master etc.), as well as just getting used to the new ways of doing things. This is how people described their attitudinal changes:

”My feelings have changed, but it took some time. At first, I did not like the whole thing of changing my patterns of thought completely, I felt there were so many new meetings and demos and just an overhead of all extra hassle. I would say the turning point for me was substituting our Scrum Master when he was on a holiday, it kind of opened up this whole picture for me and got me an insight on why all of these different Scrum practices are beneficial” (Developer 5, team 2).

”My first reaction about agile was doubtful, but after the excellent Scrum Master training my eyes opened up and I realized what this is all about. My feelings totally went from one end to another and I started to think: why haven’t we always done things this way? Especially in the field of software development, where nothing is stable except change” (Lead Developer 3, team 3).

The process itself was not always perceived as an easy one, since the feelings towards it were mixed and some people resisted the change quite a lot. The negative attitudes were mostly prevailing in team 2, where a member of the team describes his attitudes during the transformation process as following:

"In our team, this was an extremely difficult process in the beginning and of course there are still challenges, but we are heading towards the right direction. I'd say in the beginning 50% of us thought this was a good change, and the other 50% said this is micro management and everything is awful. We had to make a lot of effort for the team spirit to become better" (Developer 6, team 2).

It seemed like there were some distinctive qualities in team 2 that caused a lot of negative feelings in the beginning; The Scrum Master was abroad and did not support the change enough, and the team consisted of more senior people, who had been used to working in a certain way.

"The change process was interesting (laughs). We had two very different kinds of teams; There was team 1, where the Scrum Master is very strong and enthusiastic and in team 2 the Scrum Master was abroad and in a difficult situation in other ways as well. Therefore the change process in team 2 lacked behind, and the difficulties really stood out while the Scrum Master was not to keep the process together. In team 2, there is a lot of young, enthusiastic blood, while team 2 consists of senior experts who experiences Scrum as a threat, taking away their responsibilities and specialist tasks. But the more people learn about this, people realize it is not a threat but an opportunity. The change resistance is not as strong as it was in the beginning" (Product Owner 1, teams 1&2).

A developer in team 3, who went through the change process in another team some years ago, recalls that the process itself was not successful and it affected his feelings towards agility as well:

"We had a bit peculiar situation when we started to pursue agile, our manager was very enthusiastic about the subject, but at no point he provided us with the basic information about agile methods. All of a sudden we started working in sprints of two weeks, and in each of these two week periods we planned everything all over again. We had no goals for longer time spans. At that time I was not excited about it. Now my feelings are better, since I have been working in this team." (Developer 8, team 3).

5.3 Productivity, performance and product quality

When asked, if the employees thought that being in agile mode has improved the team's productivity and performance, the vast majority believed that these factors had improved, or even improved significantly after adopting Scrum.

The most significant factors that were thought to enable performance were shorter iterations and regular checkpoints (Sprint Demos) with the customer, as well as working according to the Sprint backlog. The shorter iterations were thought to enable getting finished products or increments of products out to the customer faster, and the Sprint backlog to enable transparency as well as overall control over things to do. The employees conceptualized better performance and quality in the following ways:

"The productivity has increased significantly, because we are able to make customer releases in faster cycles. Before, there were so many overlapping projects and other duties that small things were buried under the workload, and they were done really late. Now, we are able to finish more things, also little things (but important ones to the customer) faster and also get them into usage quicker" (Developer 6, team 2).

"Before, one could not see any results for months. Now, every two weeks we are combining results into something and are actually releasing things for the business. Even though the entities are small, you can always feel that something is proceeding" (Developer 9, team 3).

"I am pretty sure that controlling and monitoring tasks, as well as guidance increases our effectiveness. If somebody is doing some ineffectual things, it is easy to step in and guide them to do them in a better way" (Developer 8, team 3).

"It is a bit hard to say if productivity has increased, since we haven't measured it, but my gut feeling says that we are doing more right things than before, which has probably improved effectiveness. Now we are actually doing what the customer wants, so in that way we are focusing on the right thing, and on things that really need to be done" (Product Owner 1, teams 1&2).

A Developer (no. 7, team 3) and a Lead Developer (no. 3, team 3) both note that in most cases Scrum does increase effectiveness, but this depends on the current know-how of people. They claim that Scrum will not work in a team which consists mostly of junior level developers. This is because in Scrum, each team member has to independently design the functionality of each story point, whilst in Waterfall, the design is done beforehand by certain people with the needed competencies. On that account, in a Scrum team, each team member must owe the competencies and know-how to complete different story points quite individually, otherwise the team will not perform effectively. Only if pair programming was applied, an agile team consisting of mostly junior level employees was seen to be an option. A Lead Developer (no. 3, team 3) states that he has experience on the team performance dropping when a newcomer enters the team. On the other hand, he states that newcomers learn a lot quicker in a Scrum team than in a traditional one, because they are straight away *"thrown to the deep end of the pool"* by showing them the Product backlog and saying *"here, start taking tasks and ask if you don't know how to do it"*.

The only people who could not see any difference in productivity compared to previous mode or thought it may have even decreased, were the people from team 2 (developers no. 3, 4 and 5). They basically said that the team has always been extremely effective, no matter how things are organized. Developer number 5 even thinks that before agile, they were able to react pretty quickly and perhaps finish the end-product sooner than now, but on the other hand he admitted that the release process itself was more chaotic back then. Developer 3 notes, that previously it was typical that one person agreed on finishing a certain bigger entity, and would work for example 16 hours a day to finish it on time. Now, that bigger entities are split into smaller tasks between individuals, there is no reason for a person to work that long. Therefore he thinks that productivity has in fact decreased, in a way that it takes longer to finish things because people work less overtime. On the other hand, he thinks that product quality has improved. The question of workload and its impacts is discussed further in chapter 5.5.

5.4 Team dynamics and collaboration

All of the respondents felt that agile methods had increased collaboration and openness within the team. In most cases, people felt that because of close collaboration, the relationships between people were closer and better than before. All of the interviewees also thought that working as a team had significantly increased communication, which had resulted in more effective transfer of knowledge and know-how and thus improved learning.

Most of the interviewees enjoyed the close teamwork and collaboration, or at least nobody admitted that they would prefer working alone instead. When asked about how agility had affected team dynamics, comments such as the following came out:

"The relationships between team members have improved a lot, previously people worked alone on their own module, and in the past months we have done a lot of cross-functional tasks and working in pairs and coaching one another. We have deliberately done more knowledge transfer, in order for many people, as opposed to just one, being able to handle a certain area of expertise. (Developer 3, team 1)"

Everyone agreed that Scrum enhances open communication and teamwork within a team. But the way they affect the team dynamics is not a straightforwardly positive, because open communication can cause negative as well as positive feelings across the team. In all of the three sample teams, the current team spirit was regarded as positive. However, the interviewees admitted that the situation could be different, since working closely together evokes a lot of feelings between teammates. One of the interviewees compares Scrum to being in army:

"Since Daily Scrums are an obligatory routine, people will notice immediately that something is wrong if you don't show up there. Scrum also brings about more conflict situations, because of open communication. The discussion in Sprint Retrospectives can sometimes be brutally honest and that doesn't always feel so nice. But in my opinion, open communication boosts team spirit. It can be compared to army, where the people you spent the toughest times in the woods and everyone would get annoyed with each

other, are still the people that you ultimately became the closest with" (Lead Developer 3, team 3).

Some interviewees brought up the notion, that although communication and teamwork had increased, everyone might not enjoy it. Basically they claimed that some people simply would prefer working alone, depending on their different personality traits. People described the differences in people's attitudes in the following way:

"Most of the people like working according to the Scrum mode, often it is the sociable ones who like it. Then there are a lot of people who would prefer working on a project on their own, and of more traditional ways of doing things. I think that those "academic ponderers" prefer working alone. But basically they will adapt to Scrum mode as well, and they will say if they would prefer to work alone on a project or to get a certain area of responsibility for themselves." (Scrum Master/Product Owner 1, team 3)

"It depends on the different personality types within the team, how the collaboration forms. If the attitude towards the openness and transparency is negative and not being able to understand its benefits, people will start hiding things and still not communicate. This kind of problem within a team requires a strong person to guide the team to a better direction". (Developer 7, team 3)

"This mode of operation expects that everyone should open up and bring their own outcomes under certain kind of critiquing is harder for some people than to others, I have noticed that a bit…the pressure of being more open is not that easy for everyone." (Lead Developer 1, team 1).

In addition to differences in personalities, the level of experience was often seen to be an influential variable as well, i.e. differences in attitudes between junior and senior level employees:

"Usually it is the senior level people with a lot of technical know-how, who want to concentrate on their own doings and are not really interested in coaching others. Then there are the newcomers, who want to suck in all the new information, and prefer working in groups, because it accelerates their learning." (Scrum Master/Product Owner 1, team 3)

All the interviewees had a uniform opinion on the definite increase in communication within the team after agile was adopted. The main reason for improved communication was said to be the obligatory meetings that are a vital part of Scrum practices. One developer (Developer 6, team 2) also brought up the concepts of “*tiimiajattelu”* (in English: *“thinking as a team”*) as the reason for better communication.

Daily Scrum meetings were often mentioned as the most obvious link to the improved communication, because they enable the whole team gathering together face-to-face at least once a day, and shortly discuss what everyone is doing and possibly give advice to fellow team members. Some people however note that one shortcoming in Daily Scrum meetings is, that some people work off-site, so they have to participate to the meeting by a telephone or a teleconference device. Transparency, caused by information sharing in the Daily Scum meeting was linked to being able to give more positive feedback to others. One of the developers stated that:

“I get comments, such as “excellent job” from fellow team members these days. I don’t think one could give that kind of feedback, if you didn’t know what the others are doing” (Developer 1, team 1).

It was also perceived that the communication with the customer had increased significantly as well, due to Sprint Demos in each sprint. It enables the team members present their accomplishments to the customer, and the customer gives feedback on the results. Some of the developers also brought up the motivational effects that communication with the customer brings about, such as in the following quotation:

“It’s pretty good that you can try to impress your customers every certain period of time. It’s better than doing the whole project and then get feedback at the end” (Developer 2, team 3).

Sprint Retrospective meetings were also brought up when questioned about feedback and communication. One of the Lead Developers said the following:

“Retrospective in each sprint really helps in communicating what went well or what needs to be improved in the last sprint. In addition, this meeting is not only about talking, but it actually leads to concrete results and improvements as well” (Lead Developer 1, team 1).

Scrum mode changes people's roles in a way, that there should no longer be strict, predetermined areas of responsibility for everyone, but instead in an agile team, everyone should be able to take any task from the Sprint backlog (Schwaber, 2007). Therefore the roles of tester and developer, etc. should no longer apply either, but everyone is called a Scrum team member instead.

Team 3 has been working in agile mode for already four years, and many of the people felt that people did no longer specialize that much in a single area, but everyone was pretty much able to conduct any task, that had a top priority at that moment.

People in teams 1 and 2, which had been in agile mode for only a half a year when the interviews were conducted, felt that they had started to practice a lot of transfer of knowledge and know-how among team members since agile was adopted. Even though they felt the process was not near to the final state, a lot of improvement had already happened. A developer in team 2 describes the situation as following:

"In our team, in the previous mode of operation, there was a prevailing situation where everyone was doing different things and had their own specialized area of expertise. One of the things I am very happy about is, that now we are truly a team because everyone is not just doing their own thing" (Developer 3, team 2).

It became very clear that the close co-operation that was bolstered by using agile methodologies had increased transfer of know-how and skills between individuals. Predominantly this was seen only as a positive improvement, because more individuals are able to conduct the work that is needed at the given moment:

"For years, the situation has been such that only one programmer can handle one single application. So when that programmer is on a holiday, the project is in a halt until he returns back to work. Now that we have done a lot more knowledge transfer, these things are not as much tied to one person as they used to be" (Scrum Master 1, team 1).

However, some people still preferred on working just within their own specialized expertise. Usually these were the senior level employees, who had been used to working in a certain way in the same organization (or even in the same team) for many years, sometimes more than a decade. But even these people admitted that learning new skills and therefore being able to conduct a varied range of tasks was ultimately beneficial for the whole team and organization.

"..It is a two-sided thing, in a way it is nice to work just in your own comfort zone, but on the other hand I guess sometimes I have to challenge myself and do a bit of other people's work as well even though I wouldn't feel like it at that moment"(Developer 5, team 2).

5.5 Workload and stress level

Most of the employees didn't see a direct impact on the workload either increasing or decreasing after the adoption of agile methods. It was seen as a very personal thing, in either of the modes it was seen as depending on the person, how much work he decides to commit to. Most of the people did not feel the amount of work conducted had changed significantly after adopting agile. However, the agile principles of sustainable pace and transparency were seen to have an impact on the fluctuation of workload as well as effective planning of the way things are executed.

Many people thought that working in sprints instead of projects in fact enabled the workload to stay quite steady all the time. In Waterfall, the workload was described to be *"going up and down like a rollercoaster" (Developer 2, team 1).*

Transparency and structure caused by daily meetings and constantly updated and visible backlog was thought to have an impact on workload as well. A developer (developer no. 9, team 3) admits working a lot less in a traditional team, because nobody was there to monitor his daily actions.

"My workload is much bigger, because I'm working every day, not as in traditional team. –How was it then? You didn't have that much work? Or sometimes worked more and sometimes less or..? Usually in a traditional team I've got a task and nobody asked for any results for months. And every other people around are working pretty same in

traditional team. So yes, now I can use my time better I say, I have a real workload" (Developer 9, team 3).

Another developer (Developer 1, team 1) has an opposing view on workload. He states that he no longer has the need to work long hours, but he feels like getting more work done, because the work is organized much better and in a more structured way. He describes the previous work mode as *"putting out fires"* and taking people into numerous projects at the same time, even though they were already buried under a lot of work. Many people noted, that it feels like you can concentrate better on conducting the actual software development, instead of running in meetings all day long. The only meeting in a day is the Daily Scrum, which lasts 15 minutes, and the rest of the day is left for developing software. In addition, the only meetings are those that come at the end, and in the beginning of each sprint.

Some employees brought up the Product backlog, when asked about workload. Some people said that they receive less e-mail and from some random people asking for something to be done, because now all the requirements are on the backlog. This was also said to help concentration. A Product Owner (no. 1, teams 1&2) says, the backlog "*keeps the team disciplined, or actually a better word is, protected, from all the information flow during the sprint."*

Some of the interviewees also felt that because of the Sprint backlog, one could more easily decide on how much work he takes. A Lead Developer (no. 3, team 3) said that previously he had a hard time saying "no" to people who asked him to do something, whilst now the Sprint backlog had made the circulation of tasks among team members quite easy.

Approximately half of the interviewees did not see a particular link between stress level and work mode. This was explained by two differing viewpoints: Some people thought that even though the work mode has changed, the work itself is ultimately quite similar. On the other hand, some people thought that it is simply a personality trait, to either feel stressed out about work or not, and the work mode does not have a significant impact on that characteristic. The ones who did see some kind of a link between work mode and stress level, in most cases felt that working in agile decreases stress.

Many of the respondents thought that the main reason for decreased stress among Scrum team members was the fact that one can simply take one task at a time from the Sprint backlog, and there is no need to think about other stuff while working on it:

"Working in cycles of two weeks decreases stress, because we can focus on smaller entities at a time, and just focus on the next step of the process. Therefore we don't have to chew on big pieces all at once" (Scrum Master/Product Owner 1, team 3).

A Product Owner (no. 1, teams 1&2), however says that while a regular team member nowadays probably has less stress than in a mode where people where working for multiple projects simultaneously, he now has to think for them what needs to be done. On the other hand, this responsibility is split between the whole product team, so one person does not have to take care of operating everything related to the Scrum process. In previous mode, a lot of responsibilities fell solely on the Project managers shoulders.

If too much work is taken to the sprint, it can cause stress to certain individuals. This is basically a question of inaccurate estimation techniques of workload within the team, or the fact that in some cases it is very difficult to estimate the amount of hours it takes to complete a certain thing. The aim in Scrum is to avoid situations, where too little or too much work is assigned to each sprint. A Scrum team member says the following:

"The last week of a sprint is kind of hassled and stressful, because still too much stuff is concentrated there. If we have committed to something, we will do it. In the previous work mode one could not notice the timelines as well as in this mode, projects would quite often go overtime and it wasn't such a big deal" (Developer 6, team 2).

It was difficult for some people to objectively analyze if stress level had increased or decreased since the adoption of agile methods, because the change process from one mode to another often included a new role and changes in job description as well. Also, at the time that agile was adopted to teams 1 and 2, the change process was overlapping with Nokia's new organizational structure implementation, and therefore some people had a difficult time evaluating, how much that had affected their stress level in the equation, and which part was caused by working in agile mode. Those are the main reasons some people found it hard to comment on this question.

There were some interesting contradictions in terms of feeling stressed out. On one hand, working in a fixed cycle of two or three weeks makes the deadlines more strict than in project mode, where the milestones of projects were regarded as more vague targets. Since people are now supposed to be working on a sustainable pace, it does not enable them to take some days more easily, since one is supposed to get work done every day. On the other hand, a person cannot take upon multiple, large scale projects to conduct by themselves, since work is split among team members into smaller tasks. This was seen to decrease stress.

5.6 Empowerment

When asked, if agile methodologies had an effect on empowerment, the opinions were quite mixed. Empowerment in this case is referred to being able to freely make decisions regarding one's work and taking on responsibilities, on individual and team level. It also means that the team feels trusted to being able to conduct its work independently, without anyone trying to interfere with or control the process. All, except for one person had noticed a difference in empowerment, compared to the previous work mode. Most of the interviewees had noticed a positive effect, i.e. felt that the team was more self-reliant than before. However, many people said that individual responsibilities had become smaller since agile was adopted. This was seen either as a negative or a positive thing, depending on the person. There was a noticeable difference in opinions across teams, with the most positive comments coming from teams 1 and 3, whilst in team 2, people felt that agile had a negative effect on being able to take on responsibilities.

Those who felt that the team was more empowered now, than in Waterfall, pointed out the following reasons for this viewpoint:

“The team decides on what is taken on the Sprint backlog, and I can freely pick up things to do from it. Previously someone just suddenly gave me a task to do, and I didn’t have any control over that” (Developer 7, team 3).

”The priorities come from Portfolio level, which is fine, but all the time that is left afterwards, we can use for freely developing the product. In that sense the empowerment has increased” (Product Owner 1, teams 1&2).

“The team decides what it commits to and what is taken to the Sprint. We can even decide on the workload (but it has to be estimated correctly, and not too low)” (Developer 2, team 1).

”Agile philosophy states that the team is responsible for its own actions and what it takes under development, and other stakeholders just have to trust it and not intrude in the process. It is a two-way street; the team has to realize that is has the responsibility of taking the process from start to finish and others have to trust that the team is capable of conducting it and keeping their word” (Product Owner 1, teams 1&2).

The people who had negative comments about responsibility and empowerment, mostly felt that individual responsibility had become smaller. That was because everything had already been readily defined into specific user stories and split into simplified tasks in the Sprint backlog. Some people felt that this took away the freedom on designing the things from start to finish by themselves. It was mostly the senior level employees who felt this way, and those who did not have a long experience on working in agile mode. These quotations explain their feelings:

”I would claim that responsibility has become smaller on my own behalf. Before, it felt like I was in a certain project from the beginning till the end, and had more control over what was done as well as a better insight and focus over the whole project. I do have a lot of specialist skills on certain areas and liked to have a part in the design phase as well. Now it seems like I am stuck as a regular “programmer rat”, because other people do that phase and I just get to choose some simple tasks from the backlog” (Developer 5, team 2).

“My responsibilities have decreased, that is maybe the biggest change compared to the previous mode of operation. We were doing things very freely before, in good and bad” (Developer 6, team 2).

The test interviewee, who was interviewed just after agile was adopted, had pretty harsh words on empowerment:

”They tell us we are more empowered now, but I don’t see it in any way. Maybe in a way that one can plan how to execute a user story given to us….actually no, we are not more empowered in any way”(Developer no 4, team 2).

Some of the people perceived the amount of individual responsibilities decreasing a good thing:

“Maybe there is less responsibility, but I don’t know if that is a problem, I am less stressed out now (laughs)” (Developer 1, team 1).

Many of the employees felt that in agile, a single developer is able to influence their own work less than before, while the team as a whole has more power to decide what to do, than before. The latter was mainly because all of the Scrum team members gather together in the Sprint Planning meeting to decide on what, and how much work is taken to each sprint. Therefore the team level power was seen to have increased.

When asked, if the interviewees felt that Scrum and agile practices had any influence on trust and controlling, many people started talking about the transparency, which is a vital part of the Scrum process. It became obvious that people felt that the control over what was done had improved a lot. The feelings of transparency, control and structure over the development process became through procedures such as the Daily Scrums and artifacts such as the Sprint and Product backlogs, which are visible to everyone in the company. The employees describe the current situation as following:

“Agile has had an effect on trust issues in a way, that in Daily Scrum everyone tells what they have done and what they are going to do. The accountability of one’s actions has been taken to all levels of Nokia now, and it is not only the managers that are held responsible for their actions” (Developer 1, team 1).

“Nowadays we conduct surveillance within the team, previously we had none. But now that we have the Definitions of Done (DoD), we control our own, and other team member’s doings” (Scrum Master 1, team 1).

Majority of people felt that it is good that Scrum provides more control over what is being done, and transparency to other people’s doings is regarded as a good thing. These people did not perceive the transparency of the process as a means for managers to control people’s actions, but as a tool that gives structure to everyday work. Here are some quotations that illustrate the positive feelings:

“We are an autonomous team, so I don’t think the concept of control really applies. Because of controlling each other’s actions, you know what people are doing and what their strengths and weaknesses are. Therefore if you want to allocate some specific tasks, you know who to give them to” (Lead Developer 9, team 3).

“The control has a positive impact. It is easier to see regularly how the work is proceeding: Anyone can take part in the Sprint Demos to see what the team is doing at the moment, and anyone can check the roadmaps and the contents of a sprint to see how far the product and the development process are”(Scrum Master/Product Owner 1, team 3).

While people acknowledged that the control and monitoring eventually benefits the company as well as the team’s goals, people’s personal feelings related to the issue were mixed. Most of the interviewees did not have any problems with the monitoring issue, but some people did have negative things to say.

Some people felt that the control had taken away their personal freedom to work according to their own pace. Also, some people thought that the Daily Scrums can bring about pressure, because the work is brought under everyone’s evaluation, and one has to present results every day. It also brought about the fear of people watching everyone else’s doings and comparing results. However, people said that this was not the mentality inside any of the teams, but that they know teams that have this kind of atmosphere. Also, some of the developers felt that the reason why their work is being monitored is because of mistrust issues and they feel offended because of that.

The following quotations describe the people’s feelings on some issues that may have caused negative feelings:

"I think that Scrum has brought about stricter monitoring on a single developer's actions. Every day, you have to report what it is that you have done, and what you are going to do, and if some changes occur you will be told that instead of that, you will do this today. I was used to planning my own work on a weekly basis...sometimes it feels a bit dumb when you haven't finished anything since the last meeting, and you have to make up stuff" (Developer 8, team 3).

"My initial fear was that this agile brings about a certain mentality of constantly lurking on other people's doings, and you have to keep book on every bloody action you have taken and report them to the team on the next day. I mean, everyone has their good and bad days and I don't want to feel guilty for being lazy on one day. Luckily this kind of mentality has not shown in our team, but I have heard some horror stories, where the Scrum Master just has a bit too much time on his hands and monitors everyone's actions on a minutely basis…all in all, I prefer longer time spans of 2 months or so, I feel more trusted when someone is not trying to figure out what I'm doing every single minute, but that I will deliver the result by the due date" (Developer 5, team 2).

Some of the informants could not see a link between trust issues and the mode of operation. One of the developers stated the following:

"Personally I just can't work in some non-trustful environment, neither Scrum nor traditional one. So for me it doesn't make any difference. –So the mode of operation does not have any impact on these issues? –No" (Developer 9, team 3).

5.7 Wellbeing and contentment

When asked, what are the uppermost feelings related to Scrum and working in an agile mode, the responses were predominantly positive, or very positive. A few people said it was good or ok, and it really did not evoke strong feelings in them, whilst only the test interviewee (Developer 4, team 2) did not like it that much at the time the interview was conducted (in September, 2011.)

The strongest feelings (either negative or positive) were certainly in teams 1 and 2, where the change had taken place only recently. It seemed like in team 3, many of the people were so used to agile, that they regarded it as a standard mode of operation and it did not evoke strong feelings.

I wanted to sum up here all of the respondents statements about this question, because I think it will provide some kind of overview on how satisfied people are in general about working in agile mode. Ultimately this question sums up the primary research question about people's contentment towards agile methods among this sample of three teams. Basically, the strongest feelings about working in agile (either negative or positive) arose from teams 1 and 2, where agile was recently adopted. The members in team 1 were very happy and enthusiastic about agile among the whole team:

"I'm thinking of a strong enough word to describe my feelings: Amazing" (Developer 1, team 1).

"Very positive, I like it" (Developer 2, team 1).

"My overall feeling is good, and I would like to learn more. Sometimes I feel our team is too big in size and working with too many applications to be a truly agile team. But all of us in this team like it still" (Scrum Master 1, team 1).

"Motivated. Previously I had a lot of frustration towards the shortcomings in Waterfall and I often found myself thinking if I should go on with this anymore. Now those bad days come rarely" (Lead Developer 1, team 1).

"It's a good thing, I do not understand why didn't we start to do this earlier" (Product Owner 1, teams 1&2).

Team members in team 2 had a little bit more difficult time adjusting to the new mode of operation, probably because the transformation process was a bit slower and more difficult for them than in team 1. However, everyone admitted that the development process had improved since the switch of work mode.

"I don't see any negative sides in it, I am very content" (Developer 3, team 2).

"The overall feeling is very tired of all this hassle. On the other hand it feels good that we are trying to get some king of control over our development process, and there are a lot of improvements already" (Test interviewee, Developer 4, team 2).

"This is starting to feel like an OK way of doing things, after all the hassle and anxieties" (Developer 5, team 2).

"Very positive, I see a lot of improvements already" (Developer 6, team 2).

For team 3, where people had been working in agile mode for at least one year, it felt like they did not take so strongly "sides" for or against Scrum. Everyone recognized it being a good approach to software development, but that it generally did not evoke such strong feelings.

"When done correctly, damn good. If used in a wrong way, it slows things down" (Developer 7, team 3).

"It's hard to say since this is already feels like the standard way of doing things. It does not evoke any feelings" (Developer 8, team 3).

"Great" (Developer 9, team 3).

"At least coming from the other place where we had no organization at all its better but I would also like to try something else like Kanban and XP. But I'm kind of happy doing this and if I would found my own company I would use Scrum. So it is good to track what others are doing" (Lead Developer 2, team 3).

"It's ok. Difficult to say, because I have never worked in a traditionally operating team" (Developer 10, team 3).

"I think agile methods are more effective than other software development methods" (Scrum Master/Product Owner 1, team 3).

"I like this and think it is an extremely positive thing. When I first heard about Scrum here at Nokia, I was appalled about the fact that nothing related to this was taught during my studies" (Lead Developer 3, team 3).

In the next chapter, I will sum up the main benefits and shortcomings of agile development and Scrum at Nokia, as perceived by the people in my sample.

5.7.1 Main benefits of agile methodologies

When asked, what were the main benefits of applying agile and Scrum, it was noticeable that there was a set of topmost themes that arose among the answers. The themes are already presented in more detail in the previous chapters, but I summed up below some of the points that were brought up, when asked about the main benefits of agility in relation to the previous work mode. These were related to prioritizing and filtering requirements, resourcing and organizing manpower, better focus on actual work, agility in answering to customer and business needs, transparency, teamwork, effective learning and communication. Most of the informants had similar conceptions about the benefits of agility, and they brought up all or most of them during the interviews. Here are some quotations conceptualizing each of the main benefits perceived by agile team members:

"Communication has improved significantly" (Developer 7, team 3).

"It is a very good thing that people can concentrate on one thing at a time, it was very tiring when you had three overlapping projects at a time and five meetings a day" (Test interviewee/Developer 4, team 2).

"The best thing is, that we are able to answer to customer needs rapidly, and it does not take six months like in the previous mode of operation" (Developer 8, team 3).

"We are able to meet the needs of the business better than in the previous work mode. The vision of the business is communicated better due to a Product Owner being the link between the high level management and us conductors. Before, the vision got lost somewhere in the layers of middle management" (Developer 9, team 3).

"The process is controlled now, which is extremely good. The requirements go through one filter before they reach the developers" (Developer 6, team 2).

"I feel more responsibility and more support from the team, because the team is actually motivated to provide a compound result. So I feel that the team members are interested to for example to teach me as a newbie in this team, because I have to learn a lot about the existing system" (Developer 9, team 3).

"One of the main benefits is the transparency of the process, you are able to constantly see where we are at, what is being done and what is not, and can you do something to help others" (Lead Developer 3, team 3).

"Splitting tasks and responsibilities among the team has improved. In the previous mode, things were automatically assigned to certain people who always conducted similar tasks, so that a person would excel in his area of expertise but not on other areas. That resulted in resource bottlenecks and the lack of insight to the whole process and entity" (Lead Developer 1, team 1).

5.7.2 Shortcomings of agile methodologies

When asked about what things are worse now, after agile was adopted than before the transformation process, a small portion of the interviewees could not figure out anything. Most of the people had something to say to this question, but none of these people thought that there was something severely wrong, just rather small details that could be improved. But these things were mostly not related to shortcomings of the Scrum process itself, because when I asked if you would change anything in the Scrum process and agile mode itself, people did not really have a lot to say. The issues were more closely related to things that agile mode and Scrum were causing.

The main themes that arose from the question of what is worse now compared to time before agile, were related to the worsening team spirit (mainly during the transformation process), work becoming more simple, lack of a broader scope and goals, too little concentration on design and architecture and synchronizing work between teams becoming challenging. Also, big team sizes and people working on different sites or from home office seemed to pose challenges for a preferable Scrum mode. In addition, some people believed that an agile team consisting of junior level developers cannot be efficient.

It seemed like many people had a problem with the work becoming more simplified by splitting it into user stories and tasks. It seemed like most of the senior level people, and especially those who had been working in traditional mode for a long time and had only recently adopted agile (mostly in teams 1 and 2), were not very happy about this. Here are some of the quotations, which further might explain their feelings:

"I think my scope is not as broad as it used to be, now I'm just a Scrum team member and it feels like that involves only programming" (Developer 5, team 2).

"When people worked according to projects, they were involved in it from the very beginning till the end, thinking about processes and so on...now the requirements come to us ready-made, and it might feel like being part of an assembly line" (Scrum Master 1, team 1).

Many people said that being in agile mode poses challenges for being able to maintain a good team spirit, because it encourages to honesty and close collaboration. This was prevalent especially in teams 1 and 2 and in particular during the transformation phase, because the team was divided into pro- agile and opposer camps. This is how people described the situation:

"There is nothing worse now compared to time before agile, but at some point our team spirit was very bad. We started to argue with each other on if this is good or not. But it was because everything was so new, and we had done things in a similar way for ages" (Developer 6, team 2).

"At some point we had big problems with interpersonal chemistries because Agile brings about ones' deepest feelings. But luckily we have good supervisors and solving conflicts is not a problem, so we got through that" (Scrum Master 1, team 1).

One of the interviewees perceived that synchronizing work between different Scrum teams posed a challenge:

"Now that our areas of responsibility between teams is quite strictly defined, it is difficult to get resources from other teams when a need arises that needs action from many teams. Earlier, during the project mode, we could get people into projects across differ-

ent teams quite easily. At the moment, that still requires some practicing" (Lead Developer 1, team 1).

One of the developers in team 3, who had a lot of experience working according to different agile and traditional models, said that Scrum is sometimes too strictly applied by the book, and there should be room to maneuver. Some people felt that more focus should be given on the design and planning phase:

"In Scrum, I think we forget the stage where we would describe the designs in more detail, which was a forced stage in Waterfall. On the other hand, documentation and design in Scrum are up to date, whereas in Waterfall we would not update the designs after that phase was finished, and the reality might have been very different from the design" (Lead Developer 3, team 3).

"It should not be so that we force our own doings to fit the Scrum mode, but rather to take all the good things from different models that are best for us, and create a hybrid model. For example, from the Waterfall model I would adopt a bit more of the planning phase in the beginning, in order to gain a long term goal. I don't mean a 1,5 years of requirement planning, but a light one. In agile, someone just decides to test if something would work, instead of using time to plan ahead" (Developer 8, team 3).

Some people felt that, it was not necessarily Scrum and agile that caused problems, but in order for it to work properly, teams had to be small in size, and it would be preferable if people worked close to each other, since successful Scrum requires close communication and collaboration. Here are some things that people said:

"I can't figure out anything wrong with the Scrum practices, but I would like all the people to be on the same site. In project mode, when the work was more solitary, it was not that necessary because you could do your own programming for a week and then just attend the weekly status meeting"(Scrum Master 1, team 1).

"We should have a better site focus, so that on one site there is one Scrum team. We could divide ourselves into three different Scrum teams, in my opinion" (Scrum Master/Product Owner 1, team 3)

"We should start considering into dividing ourselves into two different Scrum teams, since we are 9 internal and 4 externals at the moment" (Lead Developer 1, team 1).

It was also pointed out, that an agile team consisting of only junior level developers cannot function efficiently, because the developer needs to have certain skills in order to figure out how to conduct tasks. This is how one of the developers describes the issue:

"An agile team consisting of only junior developers doesn't work. Only if there is pair programming, juniors can be included. Agile requires having experienced people on the background. In Waterfall, there was someone who would design and take care of functionality and give it to juniors ready-made, so they could just conduct the work needed. In Scrum, the team is in charge of preparing the functionalities altogether, and a group full of juniors do not have the skills to do the design independently" (Developer 7, team 3).

5.8 Summary of findings

In order to get an overview of the main findings based on each of the themes in this chapter, I wanted to summarize them in a simple grid (Table 3, below). One mark (*) indicates the opinions (positive, negative or neutral/mixed) of approximately three people on the issue in question.

Research target	**Positive**	**Negative**	**Neutral or mixed**	**Comments**
Early feelings	**	**	*	People experienced the transformation process to agile being challenging, since people did not have a clear understanding what agility means, and teams experienced change resistance. Half of the people were enthusiastic about agile from the beginning, while others experienced more negative feelings throughout the change process.
Productivity, performance and product quality	****		*	People had almost a uniform opinion about increasing productivity and product quality. They were caused by shorter iterations, concentration, transparency which increased controlling, working on sustainable pace, customer collaboration, and test-driven development.
Team dynamics and collaboration	***		**	All respondents felt that collaboration, communication and openness had increased within the team. Some had mixed opinions how pleasant it is, stating that some people prefer working on their own and do not like their work being brought up under evaluation. Also, openness was seen to result in conflicts more easily. However, none of the informants admitted that they themselves would prefer working alone.
Workload and stress level	**		***	Most of the people thought that there was no direct link between workload and stress level changing because of agile. However, people noted that work proceeded in more sustainable pace. Many people said that their workload had increased because of agile, but perceived this as a positive thing. They felt that they were able to conduct more work in less time.
Empowerment	***	*	*	Most of the people felt that empowerment on team and individual level had increased after agile adoption. However, some people felt that individual responsibilities had decreased, while the team as a whole was empowered.
Wellbeing and contentment	****		*	All informants agreed that agile had improved things in general. Most people enjoyed working in agile, while others did not experience strong feelings. None of the respondents would like to go back to working according to Waterfall.

Table 3: Summary of findings

6 Results

In this chapter, I will summarize the results of this study in regards to the research question presented at the beginning of the report. Further, I will analyze how these results take place related to previous research on agility, particularly to that conducted at Nokia. I will finish the report by implications, and recommendations based on them, that could be considered while adopting agile practices at Nokia.

Here I will summarize the key points of my research, by answering shortly to the ultimate research question that I presented in the beginning of the report.

How do the practitioners of agile software development methods at Nokia experience and account for the new mode of operation?

All in all, people in my focus group are very content with working according to agile practices and Scrum. When asked, if they would go back to the old way of working, none of the informants replied “yes”. Although some people perceived it not as being a perfect way of doing things and it did have some shortcomings, all of the people that were interviewed clearly thought it was a change for the better. This result is intact with the results from the research at Nokia by Laanti et al. (2011) where only 6% would like to go back to a traditional way of working.

Table 2 on page 36 of this report summarizes the opinions on the impact of agile development, as regarded by practitioners at Nokia in 2010. All in all, my research seemed to confirm all of these statements, as people generally thought that work was more effective and organized, quality of the products had increased, transparency of the development process had increased as well as collaboration, and bugs and errors were detected earlier. I did not ask if work had become more “fun” (see statement 5, Table 2, p. 36), but it seemed that people were very pleased with agile in general. The answers to questions 7 and 9 were controversial in my study, since people did think that autonomy on the team level had in fact increased, while some people perceived that at the same time, individual responsibilities had decreased. In regards to agile making work less hectic (question 9), the answers were mixed as well. Generally it was thought that in agile, work was proceeding in more sustainable pace than in a traditional model, but it did not necessarily mean that there was less work or that it was less hectic. However, some people pointed that agile enabled to control the workload better than in a traditional model, but at the end it depends on the person how much work he takes upon.

The aim of my research was not just to get a general view on people's perceptions and gain quantifiable results, as this has already been researched at Nokia. The main contribution of this research was to find out what are the underlying thoughts that lead to having certain kind of attitudes, and how do people perceive about the different impacts of agile (such as increased collaboration and transparency, etc.)

Next, I will present the main results that came across most significantly in the interviews of 16 agile practitioners. Although these results include some negative thoughts as well, I have to remind that these were the opinions of a rather small minority. I still wanted to bring them about more, in order for understanding why some people may not be happy with working in agile development mode. My aim is to present these findings in relation to previous research on agility, to either challenge existing theories or complement them. In addition, I wanted to see if my results are intact with the value statements of agile development, as well as those statements that the creators of Scrum (e.g. Scwaber 2007; Schwaber et al. 2007) have promised about Scrum and agile adding value, from the practitioners' points of view.

Some of the impacts of agile were seen as resulting only in positive feelings, and some of the impacts caused mixed feelings between participants. Here, I will present the most significant thoughts that arose from my study. I decided not to divide them up in different categories, since they are highly related to one another. Quite frankly, there seems to be linkages within all of the characteristics of agile.

When asked, if the agile methods had an effect on empowerment, the opinions were quite mixed. All, except for one person had noticed a difference in this, compared to the previous work mode. Most of the interviewees had noticed a positive effect, i.e. felt that the team was more self-reliant than before. This was the result of the Product team handling all the requirements and being the filter between the Scrum team and the stakeholders to the project. As a result, random people were not able to assign tasks straight to a developer, but the team decided on work to be taken on the backlog as well as prioritizing them. However, some people said that the individual responsibilities had become smaller, due to splitting the tasks between teammates, and working according to simplified user stories and tasks, that were all listed clearly on the backlog. Depending on the person, this was seen either as a positive thing, while some people saw a negative side to it. Many people felt that there was finally an order in things, and it enabled focusing on one task at a time. This was also seen to decrease stress. As said, some people had mixed feelings, by realizing that better organizing of work was ultimately a clear benefit to the organization, while at the same time they missed having challenges and working on bigger entities.

Approximately half of the interviewees in my study did not see a particular link between stress level and work mode. The ones who did see some kind of a link between work mode and stress level, mostly felt that working in agile decreases stress, because one was able to work on one assigned task at a time and had less responsibilities to consider. Previously, people would take you in to multiple projects even though you were already busy, and it seemed to be more difficult to say “no”.

However, most of the employees didn’t see a direct impact on the workload either increasing or decreasing after the adoption of agile methods, but it was seen as a personality characteristic of how much work one want to take upon. On the other hand, the agile principle of working in a sustainable pace was seen to have some sort of impact on smoothing the fluctuation of workload.

According to Laanti’s (2011) research, it seems to be that if a person feels that the team is working in a sustainable pace, they have a smaller workload and less stress. At the same time, the teams that are working according to sustainable pace, also perceive that their performance is better than in those teams that are not working on a sustainable pace. One of the respondents in my study said, that he felt stressed out on the last week of sprint, because they had taken too much work on the sprint, i.e. estimations were not accurate enough. Many of the informants noted that if there was a realistic workload taken to each sprint (not too much, but not too little), workload should be bearable and there should be no stressing out on things.

It was the clear majority who thought that productivity, performance and product quality had increased as well. According to previous research on agility, this has seemed to be the case in the majority other studies as well (see chapters 3.3 and 3.4, pp. 39-43). Better product quality was seen to be achieved by working on smaller entities, taking all the unnecessary features off (or at least prioritizing the most important ones), and test-driven development and continuous integration practices.

The underlying reason for better productivity was better organizing of work which led to less meetings and other additional stuff on top of the actual development work. Also, people thought that progress was made more quickly, since in every three weeks or so, there was a working product or an increment of product released to the customer. However, some interviewees pointed out that an agile team consisting of junior level developers only might decrease productivity, because they do not have the required skills to design the functionalities of tasks individually. Only if pair programming was practiced as a way to balance the know-how within the team, agile could work amongst juniors as well. They said that junior level developers might have it easier in a Waterfall team, where there are people who are specialized in design and take care of that for the juniors. Earlier studies have revealed that agile methodologies might not efficient in a team that consists of people of different levels of expertise and skills (e.g. Melnik & Maurer, 2002 in Dybå & Dingsøyr, 2008). Two of the creators of the Agile Manifesto, Alistair Cockburn and Jim Highsmith, suggest in their article, that the critical people factors that an agile team member should possess are amicability, talent, skill and communication skills (Cockburn & Highsmith, 2001). The previous finding strengthens the evidence of at least for the "talent and skill" –part.

All of the respondents felt that agile methods had increased collaboration, communication and openness within the team. In most cases, people felt that because of close collaboration, the relationships between people were closer and better than before. However, they admitted going through some tough times during the transformation phase, especially when part of the team resists the change while others promote it. To summarize, agile transformation as well as the methods itself evokes the stronger formation of feelings, either negative or positive ones.

Most of the interviewees enjoyed the close teamwork and collaboration, or at least none of my informants admitted that they would prefer working alone instead. However, some of the informants said that it had a lot to do with a certain kind of personality characteristics, and some people noted that among software engineers, there are a lot of a people who might enjoy working and solving problems alone, better. Whitworth and Biddle's (2007) qualitative research on the social nature of agile teams, resulted in similar findings where some of the developers had the tendency for to feel stressed or exhausted after spending the whole day being 'on' or socially active. This can also explain if some individuals feel more stressed out in agile mode, so it is not caused by workload only. One of the informants pointed out a similar thing, saying that the anti-social people might feel stressed to speak up in Daily Scrums etc. All in all, the informants said that even the anti-social ones would usually adjust to a sociable atmosphere, and at the end, they formed a minority.

People felt that close collaboration was resulting in increased knowledge transfer, which was also mostly seen as a positive thing, although some of the senior level developers found some negative characteristics in it as well. They were used to being good at their own area of expertise and were not always eager to learn new things, or were afraid of "losing" their work to somebody else. However, knowledge transfer resulted in everyone being able to conduct any task there was, and this was actually seen as one of the main benefits of agile by many people.

However, working on different sites was seen as challenging the effectiveness of communication within the team. According to the principles defined in the Agile Manifesto, that face-to-face communication is one of the key factors inside an agile team (see statement 6 on Appendix B, p. 104). In order for the team to communicate effectively, it is important that all the team members are physically located close to one another (Maurer, 2002). In addition, some people felt that communication and work allocation between different Scrum teams was challenging. This finding is in line with Karlström and Runeson's (2005, cited in Dybå & Dingsøyr, 2008) research of the cohesion of XP teams, which experienced improved communication within the team, but were perceived more isolated by other teams. Lindvall et al. (2004) found cross-team communication to be a crucial challenge for agile teams in large companies, and in particular at Nokia where teams are spread across several physical locations. Team 3 was already applying a solution for this, by synchronizing work in same cycles with a few other Scrum teams of whose work was co-dependent on one another, and meeting with some of the team members from those teams once a month in an Agile Release Train (ART) meeting (see definition, p. 24). This was found to be beneficial for co-operating with other teams.

One of the clear themes that arose among the interviews was the development process becoming more transparent, and issues related to that. Again, it was seen as one of the main positive things in agile, but also one of the most controversial issues as well. If the transparency was used for positive purposes, such as getting an insight of where the project is at, it was seen to be a positive thing. If it was used for the wrong purposes, such as strict monitoring each other's' actions and managers pushing to conduct more work, it was seen as having negative impacts. According to Chalykoff and Kochan (1989, cited in Syed-Abdullah et al., 2006), high level of monitoring has a negative impact on wellbeing at work.

However, none of the people within these three teams felt that transparency was bringing out negative actions within their own team. Some people also felt, that updating each task to the requirements' tool was a sign from the higher management of not trusting people. This viewpoint was challenged with the majority of informants, who saw only positive sides to transparency and noted that those people who do not understand the benefits of agility, have something bad to say about everything related to it. This supports the finding of Laanti et al. (2011), that those who do not like working in agile mode, find it difficult to see the benefits of it. Further, according to Korhonen's (2010) findings, positive engagement in agile correlated with being able to remark the improvements in code quality after agile was adopted.

7 Implications

The implications of this study have contributed to the research of the practitioner's perceptions and experiences related to agile software development methods, in a large company applying Scrum and scaled-up Scrum as their main agile methodology. Although it has been debated on, whether agile methods can be successful in large companies, this study strengthens the proof for the opposite direction. The limitation of this study is, however, the rather small number of participants (16 in total) across only three teams inside a single company. Therefore, the generalizability of the results is questionable regarding the whole Nokia, and even more so to other company contexts. However, in this case of 16 practitioners of Scrum, everyone thought that the adoption of agile methodologies has been a change for the better. Based on the viewpoints of these informants, there are some managerial implications that could be considered in the future at Nokia regarding the use of agile methodologies.

First of all, I noticed that people's attitudes towards agile practices were improved as they learned more about the benefits of it, especially through training (such as Scrum Master training), and when they felt that they were actively included in the change process. Whitworth and Biddle's (2007) study supports this finding, by stating that failing to involve people in close communication can remove the cohesion of agile teams. Open communication from the beginning is very important, as well as enough training sessions and available support from agile coaches throughout the change process. Team 1 felt that part of the reason their Scrum transformation was so successful, was because one of the team members had a history of working in an agile team previously, and he seemed to have taken the role of a mentor and a coach for that team.

Second, there seems to be teams where agile is not functioning well enough, since some people would like to go back working according to previous mode of operation. It seemed like the underlying reasons why agile is not working in some teams, is not necessarily because there is something wrong with the methods and practices itself, but in the ways they are used. There must be individual difficulties in each of these teams, depending on the situation. However, regarding what the informants of my research pointed out, I noticed two different kinds of patterns that might cause some problems.

First, if Scrum practices are only vaguely followed, they might not provide actual benefits that agile methods are able to bring about. For example, if the Sprint and Product backlogs are not properly updated, the benefit of transparency suffers. Or, if estimation techniques are not accurate and too much work is constantly taken to each sprint, the benefit of working on a sustainable pace suffers. Or if Daily Scrums are held only twice a week, the benefits of communication, work allocation and transparency suffer. For future research at Nokia, it could be measured if there is a correlation pattern between not following Scrum practices thoroughly and with dissatisfaction among teams. Some previous research has already shown, that the benefits of agility are not reached if agile practices are not followed thoroughly enough. For example, Korhonen (2010) refers to a recent survey made at Nokia Siemens Networks, which concluded that several agile engineering practices and working at a sustainable pace need to be in place before significant improvements in code quality can be reached.

Furthermore, Scrum might be used for wrong purposes within the team, such as strictly monitoring each other's actions, which can create conflicts and bad spirit within the team. Some people brought up the significant role of the Scrum Master in this case, since he has to be dedicated enough to offer constant support and at the same time function as a mentor and not as a commander. It seemed like sometimes the Scrum Master was not able to fulfill this criterion, and it had in fact negative effects on the team spirit. In my opinion, this could be improved by rotating the position of Scrum Master within the team. However, if some people are viewed as incapable of filling role of a Scrum Master, a team could consider choosing a Scrum Master amongst them, in order for the right person to be selected for the role. A successful Scrum Master needs to possess the skills of not only knowing the method, but being able to coach people, communicate with them and being and adopt the role of a mentor instead of a typical manager. All in all, whatever issues related to "command-and-control" type of management take place within a team, they should be researched and addressed, because this kind of thinking is not supported by agile philosophies.

It also came apparent, that some people do not like working in the Scrum mode, no matter how well it is applied. A correlation has been proven by previous research between the longer experience in traditional methods resulted in being more dissatisfied with agile methods (Laanti et al., 2011; Melnik & Maurer, 2006). This finding was found to be supported in the based on the interviews I conducted. It seemed like the people who had the most difficult time adjusting to agile, were senior developers who had been working in a traditional team for many years and who had only recently started to practice agile development. In addition to the adjustment process being slower than with newbies, they felt that their expertise was not valued in an agile team, where tasks are distributed quite evenly among Scrum team members and planning is mostly the responsibility of the Product team. This problem has to be addressed, as it is not a good thing that people with a lot of expertise feel that their responsibilities are taken away. The way Scrum is applied in practice needs to have some kind of components that take into consideration the senior developers, without making them feel like the new methodology is forced on them. It is important to include these people into the change process, by making them feel that they are a vital part developing the new mode of operation.

In addition to experience, being dissatisfied with agile might be a question of differences in personalities. Young et al. (2005, cited in Dybå & Dingsøyr, 2008) have already identified the good and bad characteristics of people working in XP teams (see chapter 3.2) and this branch of research could be extended further to figure out if there is a correlation between personality traits and enjoyment of agile development mode. This is important to know, because agile methods involve the characteristic of being socially active and open, which has seen to cause stress among some individuals (Whitworth & Biddle, 2007). The opinions of my informants strengthen the evidence to this direction, since many people believed that the more sociable people would enjoy agile more. However, this conclusion cannot be drawn straightforwardly. Since that agile development encourages also the more introverted individuals to being socially active, they might actually enjoy it. A suggestion for future research would be, if there is in fact a correlation between the personality traits and enjoyment of working in an agile team, or enjoying teamwork in general. In practice, this could be tried to take into account when employing new people. Since agile methods are deliberately pushed to majority of software development teams at Nokia, it is important to ensure that a potential candidate prefers teamwork over working individually.

As previous research indicates, (e.g. Korhonen, 2010; Laanti et al. 2010) the people who are dissatisfied with agile fail to see the benefits of it as well. My research strengthens the evidence, as it seemed like the people who were most resistant towards agile did not see it bringing significant value to the team. As Poppendieck and Poppendieck (2007, cited in Korhonen, 2010) suggest, the sense of progress is one of the motivating factors during agile transformation. Korhonen (2010) suggests, that one way to increase the motivation within agile teams is to provide them with for example, comparable quality metrics before and after the adoption of agile. By being able to see the actual benefits of agile, the motivation towards the methodologies might improve as well.

References

Abrahamsson, P. , Salo, O., Ronkainen, J., & Warsta, J. (2002). *Agile software development methods: review and analysis,* VTT Technical report. Espoo: VTT Publications.

Avison, D., & Fitzgerald, G. (2003). Where now for development methods? *Communications of the ACM, 46*(1), 78-82.

Bahli, B., & Zeid, E. (2005). The role of knowledge creation in adopting extreme programming model: an empirical study. *Information and Communications Technology, Enabling Technologies for the New Knowledge Society: ITI 3rd International Conference,* 75-87.

Boehm, B. (2002). Get ready for the agile methods, with care. *Computer 35*(1), 64-69.

Ceschi, M., Sillitti, A., Succi, G. & De Panfilis, S. (2005). Project Management in plan-based and agile companies. *IEEE Software 22*(3), 21-17.

Cockburn, A. & Highsmith, J. (2001) Agile Software Development: The People Factor. *Computer 34*(11), 131-133.

Dybå, T. (2000). Improvisation in small software organizations. *IEEE Software 17*(5), 82–87.

Dybå, T., & Dingsøyr ,T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology 50*(9-10), 833-859.

Erickson, J., Lyytinen, K., & Siau, K. (2005). Agile modeling, agile software decelopment, and Extreme programming: The state of research. *Journal of database management 16*(4), 88-100.

Eriksson, P., & Kovalainen, A. (2008). *Qualitative methods in business research* (1st ed.). London: Sage Publications.

Fowler, M., & Highsmith, J. (2001, August). The Agile Manifesto. *Software Development*, 28-32.

Goodman, H. (2001). In-depth interviews. In Thyer, B. A. (ed.), *The Handbook of social work research methods* (pp. 308-319) Thousand Oaks: Sage Publications.

Guion, L., Diehl, D., & McDonald, D. (2009). *Conducting an in-depth interview*. University of Florida, IFAS Extension. Retrieved from http://edis.ifas.ufl.edu/pdffiles/FY/FY39300.pdf. Accessed March 28th, 2012.

Hirsjärvi, S., & Hurme H. (2001). *Tutkimushaastattelu – Teemahaastattelun teoria ja käytäntö*. Helsinki: Yliopistopaino.

Huo, M., Verner, J., Zhu, L., & Babar, M. (2004). Software quality and agile methods. *Computer Software and Applications Conference, COMPSAC 2004. Proceedings of the 28th Annual International*, 32-41.

Kettunen, P., & Laanti, M. (2008). Combining agile software projects and large-scale organizational agility. *Software Process Improvement and Practice 13*(2), 183-193.

Korhonen, K. (2010). Exploring defect data, quality and engagement during agile transformation at a large multisite organization. *11th International Conference on Agile Processes and eXtreme Programming in Software Engineering*, 88-102.

Laanti, M. (2012). Agile and wellbeing — Stress, empowerment, and performance in Scrum and Kanban teams. *Submitted for publication process.*

Laanti, M., Salo, O. & Abrahamsson, P. (2011). Agile methods rapidly replacing traditional methods at Nokia: A survey of opinions on agile transformation. *Information and Software Technology 53*(3), 276–290.

Larman, C., & Vodde, B. (2008). *Scaling lean & agile development: Thinking and organizational tools for large-scale Scrum*. Boston, MA: Addison-Wesley.

Layman, L., Williams, L., & Cunningham, L. (2004). Exploring
extreme programming in context: an industrial case study. *Agile Development Conference,* 32-41.

Leffingwell, D. (2011). *Agile Software Requirements : Lean requirements practices for teams, programs, and the enterprise*. Upper Saddle River, NJ: Addison-Wesley

Leffingwell, D., & Aalto, J-M. (2009*). A lean and scalable requirements information model for the agile entreprise.* Whitepaper. Leffingwell,LLC.

Mann, C., & Maurer, F. (2005). A case study on the impact of Scrum on overtime and customer satisfaction. *Proceedings of the Agile Development Conference*, 70-79.

Mannaro, K., Melis,M., & Marchesi, M. (2004). Empirical analysis on the satisfaction of IT employees comparing XP practices with other software development methodologies. *Lecture Notes in Computer Science 3092*, 166–174.

Melnik, G., & Maurer, F. (2006). Comparative analysis of job satisfaction in agile and non-agile software development teams. *Lecture Notes in Computer Science 4044*, 32-42.

Nandhakumar, J., & Avison, J. (1999). The fiction of methodsological development: A field study of information systems development. *Information Technology & People 12*(2), 176-191.

Nerur, S., Mahapatra, R. & Mangalaraj, G. (2005). Challenges of migrating to agile methods. *Communications of the ACM 48*(5), 72-78.

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

Rapley, T. (2001). The art(fulness) of open-ended interviewing: some considerations on analysing interviews. *Qualitative Research 1*(3), 1-25.

Robinson, H., & Sharp, H. (2004). The characteristics of XP teams. *Lecture Notes in Computer Science 3092*, 139–147.

Royce, W. (1970). Managing the development of large software systems. *Proceedings of IEEE WESCON*, 1-9.

Schwaber, K., Laganza, G., & D'Silva, D. (2007). *The truth about agile processes: frank answers to frequently asked questions.* Cambridge, MA: Forrester Research, Inc.

Schwaber, K. (2007). *The Enterprise and Scrum*. Microsoft Press.

Sillitti, A., Ceschi, M., Russo, B. & Succi, G. (2005). Managing uncertainty in requirements: A survey in documentation-driven and agile companies. *Proceedings of Software Metrics 2005, 11th IEEE International Software Metrics Symposium*, 17–27.

Sutherland, J., & Schwaber, K. (2011). The Scrum papers: Nuts, bolts and origins of an agile framework. Retrieved from http://jeffsutherland.com/ScrumPapers.pdf. Accessed April 12th, 2012.

Syed-Abdullah, S., Holcombe, M. & Gheorge, M. (2006). The impact of an agile methodology on the wellbeing of development teams. *Empirical Software Engineering 11*(1), 143–167.

Tuomi, J., & Sarajärvi, A. (2009). *Laadullinen tutkimus ja sisällönanalyysi* (5th ed.). Helsinki: Kustannusosakeyhtiö Tammi.

VersionOne (2010). 5th Annual State of Agile Development Survey Final summary report. VersionOne Inc. Retrieved from http://www.versionone.com/pdf/2010_State_of_Agile_Development_Survey_Results.pdf. Accessed December 12th, 2011.

Vuorela, S. (2005). Haastattelumenetelmät. In Ovaska, S., Aula, A. & Majaranta, P. (eds.) *Käytettävyystutkimuksen menetelmät* (pp. 37-52) Tampereen yliopisto: Tietojen-käsittelytieteiden laitos.

Wellington, C., Briggs, T., & Girard, C. (2005). Comparison of student experiences with plan-driven and agile methodologies. *Proceedings of the 35th ASEE/ IEEE Frontiers in Education Conference 21*(5), T3G-18-T3G-23.

West, D., & Grant, T. (2010). *Agile Development: Mainstream Adoption Has Changed Agility*. Forrester Research.

Whitworth, E. & Biddle R. (2007). Motivation and cohesion in agile teams. *Lecture Notes in Computer Science 4536*, 62-69.

Websites

Annual report – Nokia Corporation (2011). Retrieved from http://i.nokia.com/blob/view/-/966698/data/2/-/form20-f-11-pdf.pdf. Accessed May 7th, 2012

Beck, K.., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., … Thomas, D. (2001). AgileManifesto. Retrieved from http://agilemanifesto.org. Accessed October 10th, 2011.

Nokia Intranet. Training material. PowerPoint presentations. Retrieved throughout fall, 2011.

Nokia lyhyesti. (2010). Retrieved from http://www.nokia.fi/NOKIA_FINLAND_50/Nokia/About_Nokia/pdf/Lyhyesti_09.pdf. Accessed December 12th, 2011.

Nokia – Our structure (2011). Retrieved from http://www.nokia.com/global/about-nokia/company/about-us/structure/our-structure/ Accessed December 12th, 2011.

The Nokia story (2011). Retrieved from http://www.nokia.com/global/about-nokia/company/about-us/story/the-nokia-story/ Accessed December 12th, 2011.

Personal communication

Maarit Laanti, e-mail. November 17th, 2011

Maarit Laanti, e-mail. May 3rd, 2012

Software Quality Manager, interview. September 30th, 2011.

External Consultant, interview. August 23rd, 2011.

Appendices

Appendix A: The value statements of the Agile Manifesto (http://agilemanifesto.org)

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- *Individuals and interactions over processes and tools*
- *Working software over comprehensive documentation*
- *Customer collaboration over contract negotiation*
- *Responding to change over following a plan*

That is, while there is value in the items on the right, we value the items on the left more.

Appendix B: The principles of the Agile Manifesto (http://agilemanifesto.org)

1) *Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.*
2) *Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.*
3) *Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.*
4) *Business people and developers must work together daily throughout the project.*
5) *Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.*
6) *The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.*
7) *Working software is the primary measure of progress.*
8) *Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.*
9) *Continuous attention to technical excellence and good design enhances agility.*
10) *Simplicity - the art of maximizing the amount of work not done - is essential.*
11) *The best architectures, requirements, and designs emerge from self-organizing teams.*
12) *At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.*

Appendix C: Interview questions

1. How long have you worked for Nokia and much experience do you have on applying agile software development methods in your work?

2. Can you describe your work – what is it that you do?

3. Have you ever worked in a team that applies traditional Waterfall-like development process?

4. Does your team use other software development methods besides Scrum? How strictly do you follow Scrum and its practices?

5. Can you compare the time before agile to the present moment? What things are different?

6. What were your initial feelings on agile development, and how have the feelings possibly changed over time?

7. Can you tell me about the agile transformation process and the feelings you and your workmates had during that process? What caused these feelings?

8. Do you feel that working in an agile mode affects your, as well as your teammates' role and tasks? Are the effects good or bad?

9. Can you elaborate on what are the things that you think are worse now than before agile was adopted?

10. What things are better now after agile model has been adopted into use?

11. If you had the right to choose, would you go back to the old way of working and why or why not?

12. What are the factors that you feel that have the most effect on your job satisfaction? How well are these factors realized at the moment, and do agile methods have any effect on them in your opinion?

13. Do you think that working in an agile team is different from a traditional team in terms of team dynamics?

14. Do you think applying agile software methods have had any effect on team dynamics and working in a team? What kinds of effects?

15. Does working in an agile mode affect your stress level and in what way?

16. How is your workload, and do you think that applying agile methods might have an effect on it?

17. How has working in agile mode affected on feelings of getting responsibility and being empowered?

18. Does agile way of working have an effect on trust and control and what kind of effects?

19. Do you think applying agile development methods has an effect on your team's performance and effectiveness and how?

20. Do you feel that communication within your team and between other teams have changed after taking agile into use?

21. Do you feel that working in sprints and releases suits you and why? Do you feel that working according to user stories and tasks is nice?

22. How would you define an agile team, and do you feel that your team is agile at the moment? Why or why not?

23. Do you feel that you are aware on how the whole agile process works and what in fact is agility and agile practices? Would you like to get more training related to agility? Do you thinks there are enough chances to get more training if you want to?

24. If you could decide, what factors would you change in the current agile way of working?

25. What is the overall feeling you have about applying agile software development methods and Scrum in your work?