Turun yliopisto
University of Turku

# CHALLENGES IN MANAGING LARGE-SCALE AGILE SOFTWARE DEVELOPMENT TRANSFORMATION

Master's Thesis
in International Business

Author:
Jenni Aaltonen
88541

Supervisors:
D.Sc. Birgitta Sandberg
D.Sc. Elina Pelto

8.5.2014
Turku

**CONTENTS**

FIGURES

TABLES

# 1 INTRODUCTION

## 1.1 Achieving agility – a mindset change

*"Whether top management, middle management, or the workers who actually do the work, we are all human so we're like walking misconceptions believing that the way we do things now is the best way". (Ohno 2007, 15)*

The software industry has evolved to become one of the most important industries employing millions of practitioners and creating some of the most essential products used for maintaining and extending our lifestyles. Software has become the embodiment of much of the world's most valuable intellectual property (Leffingwell 2007, 5). As a consequence, the marketplace is being dominated by intensive global competition in which the leaders are often separated by the second-place finishers by their ability to more quickly create and deliver solutions that address real needs of users and customers better. In the prevailing ambiguous and complex world, being relentless in constantly enhancing the provided solutions to assure an ongoing fit to customers' needs, is a mantra sounding simple, but requiring often fundamental changes from organizations to truly master it. (Kotter 1996, 18-19; Leffingwell 2007, 5-6; Pich et al 2002, 1010.)

The experience of large-scale software solutions roots back only a few decades which in technological terms is a relatively short time span. During this period number of methods to control and manage the production of software have been initiated, starting with simple "code - fix - code some more" early practices and progressing through more formal and structured methods, such as sequential and stage-gated waterfall model of development (Leffingwell 2007, 6.) Although there exist success stories achieved by applying these methods, the results are inconsistent (Pich et al 2002, 1008).

The changes occurred in the marketplace during the last decade have created a trend for more agile and incremental methods in the constant search for a better way to create software (Pettersen & Wohlin 2009, 1). Software projects do not ultimately succeed because of development techniques; the key to their success is in ideas, individual responsibility, basic principles shared by team members and collaborative interaction (Highsmith 2000, 185). Thus, the agile methods are characterized by emphasizing speed and flexibility, and by recognizing individuals and their interactions as a critical success factor for achieving agility (Appelo 2011, 22). The benefits showcased by agile methods for those mastering them include faster time to market, better responsiveness to changing customer requirements and higher application quality as well as potential for significant competitive edge (Kettunen & Laanti 2008, 184; Leffingwell 2007, 6;

Schwaber & Beedle 2002, 108-110, Appelo 2001, 22-24, Highsmith 2000, Poppendieck & Poppendieck 2003, 70-71).

The software development industry has seen a major increase in adoption of agile software development methods during the last years. According to the largest annual State of Agile Development survey conducted by VersionOne (2011) over 50 percent of the respondents replied to have personally practiced agile for over two years. The agile methods are not, however, implemented without problems. It has been claimed that organizations adopting agile are not fully aware how large changes are actually required (Schwaber & al. 2007, 5). In fact, others prefer to discuss of *agile transformation* instead of adoption to emphasize the mindset change effort required in every aspect of the plan-driven organization proceeding to change its software development process (Sahota 2012, xi). The results of the VersionOne survey show that greatest barrier for successful agile implementation in organizations appears not to be awareness of the agile methodology itself, but internal company cultures. The respondents of larger organizations (more than 500 employees) said that lack of management support (27%) and general resistance to change (26%) were major barriers for agile implementation.

Hence, agile methods are not a toolkit or a cookbook recipe that can be applied in all circumstances the same way. Process models are only as good as individuals applying them (Pelrine 2011, 27). Thus, organizations need to first develop an extensive understanding of what makes the agile practices work and which factors prevent achieving agility (Laanti 2012, 27) in order to successfully apply agile methods. This understanding creates the basis for sustaining and scaling up the agile intentions (Pelrine 2011, 27).

Scientific studies focusing on only challenges organizations have encountered in their agile transformation journey or studies of mature agile teams with at least one year of experience in agile development are few (Dybå & Dingsøyr 2008, 841). Furthermore, references of organizations which would have realigned their organizational setting in the middle of their agile transformation are lacking. This case study tries to address this gap by investigating the inefficiencies encountered in agile transformation process in the context of large-scale software development within teams with more than one year of agile experience part of an organizational setting with challenging, fast-changing business climate.

## 1.2 Problem setting

The main purpose of this study is to enhance understanding of the challenges of large-scale agile software development transformation management process and bring insight into the underlying factors for such challenges. Furthermore, this paper aims to present suggestions how to address the challenges which the managers have faced in the case

organization after the change to the new mode of operation based on agile principles. This objective is to be achieved by answering the following research problem and sub-questions:

*"Why managing agile software development transformation is experienced challenging in the case organization?"*

Sub-questions:

1. *What characterizes agile software development process compared to plan-driven process?*
2. *What kind of challenges the case organization has faced during the transformation to agile software development process?*
3. *How to mitigate the challenges?*

The motivation for this research has been iterated by a practical need of the case organization to gain insight and understanding of the major underpinning impediments the agile teams have perceived during their one-year journey towards agile in order to mitigate such challenges and eliminate the repetition of ineffective practices to get most out of the benefits. In addition to past learning, this research focuses also on currently existing obstacles in order to create a unique understanding of how the mindset and practices have developed since the beginning of the process.

Although the various benefits associated with agile implementation are the driver for organization to consider such undertaking, this study is limited to focus mainly on different aspects of agile challenges and required mindset change to enable successful agile transformation. Organizations which are still pondering on agile transformation should find the presented findings useful from the perspective of what kind of challenges they should prepare to face in their agile implementation process. These challenges are discussed from the aspects of management, organization, development process and team dynamics. Furthermore, suggestions on how such challenges could be addressed are also considered in the final conclusions.

This study is structured as exhibited in Figure 1. Chapter 2 discusses the different software development methods, plan-driven waterfall method and agile models, as well as their distinct differences. It also seeks theoretical equivalent from social complexity science for understanding why agile methods work. Furthermore, Scrum as an agile method is introduced followed by an overview how to scale it. Subsequently, the impact of organizational culture to agile is discussed followed by a literature review on agile challenges.

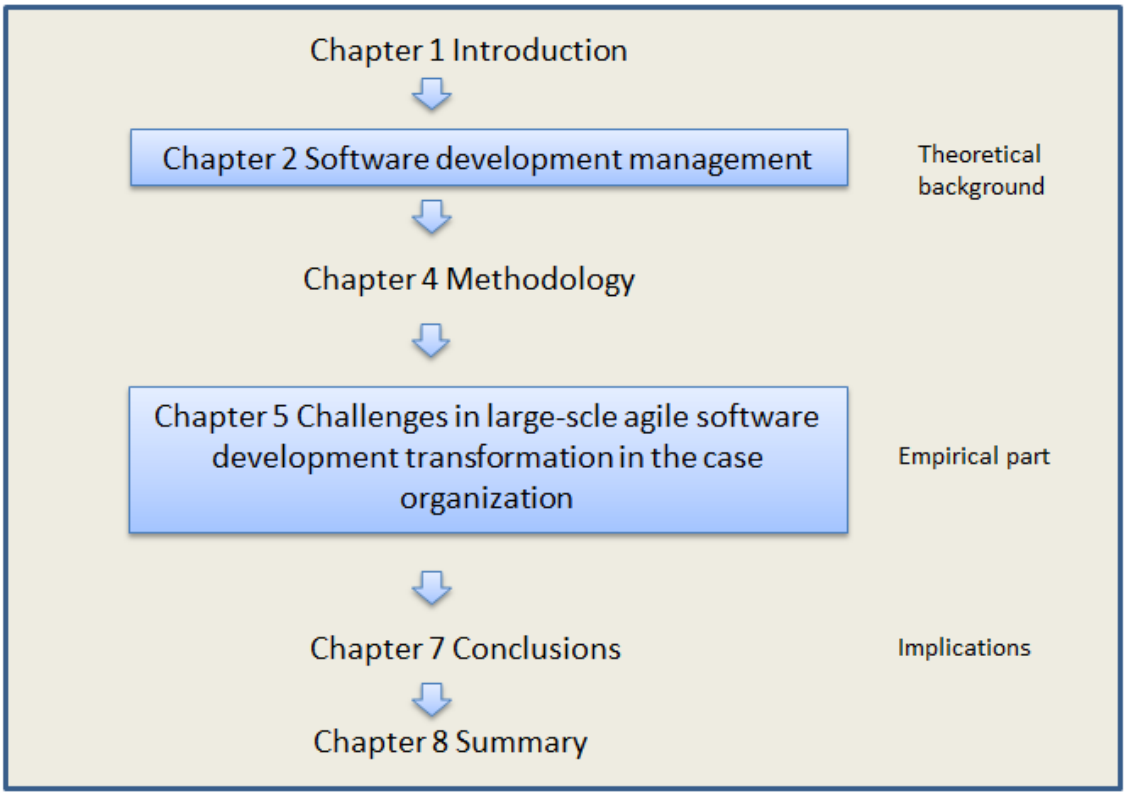


Figure 1 The structure of the thesis

Chapter 4 explains the methodology used in this study. Chapter 5 gives an introduction to case organization and presents its scaled agile software development process model. Furthermore, the results of the collected data are contemplated in a narrative form in Chapter 5. Chapter 7 presents the conclusions and implications of the findings, and Chapter 8 summarizes the work.

# 2 SOFTWARE DEVELOPMENT MANAGEMENT

In order to gain perspective how agile software development produces better results, the following chapter discusses first the characteristics of traditional plan-driven software creation model. This is followed by an overview of characteristics of agile methodologies in general, contemplation on understanding why the agile methods work and how they are different compared to traditional models. Scrum as an agile methodology and its scaling potential to larger organization are introduced in their own sections, which are followed by a contemplation of the impact of organizational culture to agile transformation. Synthesis summarizes the focal themes of this chapter.

## 2.1 Traditional plan-driven methodologies

Traditional software development approach has treated development as a predictable factory production process in which the desired result is produced by adding a specific amount of money and time with a set of programmers and managers. Within this context, the development process is divided into sequential steps where deliverable outcomes are predicted on a given set points (Pelrine 2011, 28). This model of development has been with the software industry since 1970s (Leffingwell 2007, 17). It is believed (e.g. Barlow et al. 2001, 26; Larman & Basili 2003, 48) that traditional plan-driven development, often referred to as waterfall model, has its roots in a scientific article written by Royce ("Managing the Development of Large Software Systems") in 1970s. This model was a substantial improvement to the "code-and-fix" practices which were common in the early days of software industry (Leffingwell 2007, 18).

Plan-driven development includes approaches such as the waterfall model, Rational Unified Process (RUP) and the V-model of which the waterfall development model has been studied extensively (Petersen & Wohlin 2010, 657). This study refers to waterfall development method when describing the approach of traditional or plan-driven software development model. The plan-driven approaches are characterized by a need to have the desired functionalities of the software specified beforehand. Furthermore, a detailed plan is constructed from the start of the project until the end of the project, requirements are specified in high level of detail and rigor change management process is implemented afterwards. Architecture and design specifications need to be completed before proceeding to implementation and programming work is only carried out in the coding phase. Equally, testing is performed only at the end of the project, and quality assurance is addressed in a formal way. (Petersen & Wohlin 2010, 657.)

In the traditional linear process model of project management, as illustrated in Figure 2, the basic development phases are executed in a sequence. For software development

industry this has meant in practice following these steps: first understand and analyze the requirements, then design a system that can address such requirements, after that implement the design and test it, and finally deliver the system to users for operations and maintenance. (Royce 1970, 328.) Each phase is documented and documents are then formally validated before moving to the next phase (Khalifa & Verner 2000, 360).

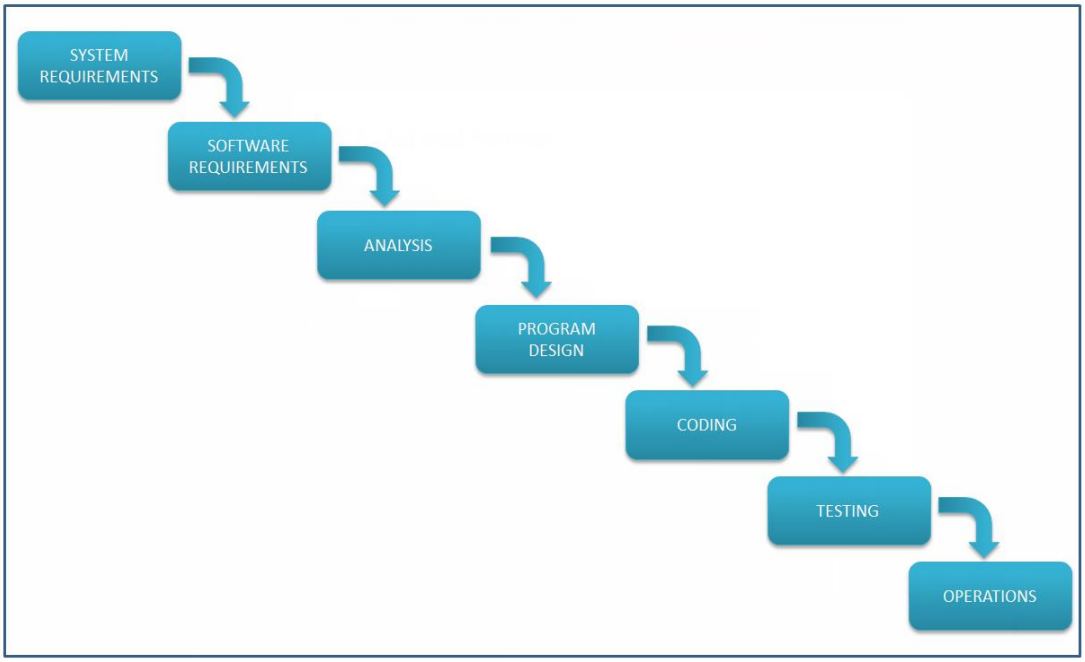


Figure 2 Traditional plan-driven software development model (Royce 1970)

Larman and Basili (2003, 55) argue that the early adoption and promotion of sequential, plan-driven methodologies in the young field of software development relies on the simple-to-recall method: first the requirements, then design and then implementation. IT gives the illusion of an ordered, accountable and measurable process with simple document-driven milestones.

Although the waterfall model of development has been the standard and practiced largely in the software industry, it has also shown to have disadvantages experienced by many. Solutions delivered using the waterfall model are both late and do not fundamentally address the customer's real needs. (Leffingwell 2007, 17-20; Petersen & Wohlin 2010, 657.) In fact, although the waterfall concept was first introduced by Royce, he himself noted that implementation based on this concept is risky and invites failure (Royce 1970, 329). System requirements are frozen and completely specified before design begins (Khalifa & Verner 2000, 360-361). The testing phase which occurs after all the development has been finished is the first event when the developed software functionalities are tested against the requirements. If they do not match, in worst case the development process will return to its origin for requirements or design changes

resulting in schedule and cost overruns (Royce 1970, 329). Managing a large scope of requirements is difficult with consequences that customers' current needs are not addressed at the end of the project and resulting implemented features which are not used (Petersen & Wohlin 2010, 657). The reasons for this are found to be in the changing customer needs and the lack of opportunity to clarify misunderstandings which have the root cause in the inherent nature of the waterfall method: the lack of opportunity for the customer to provide feedback on the system (Ibid 2010, 657). Waterfall method emphasizes also the importance of heavy documentation which is not always practical or even suitable depending on the initiative under development (Khalifa & Verner 2000, 361).

The underlying reasons for the unsuitability of the traditional methods are in the surrounding reality which is ambiguous and complex in nature (Pich & et al. 2002, 1010). New product development, such as software development, is often intangible which sets limits for early phase requirement setting. All the requirements may not be known in the project start-up and they may change along the project. The traditional approach lacks the supporting elements for project dynamism: on-the-fly adaptation to changes as well as management of uncertainty. (Fernandez & Fernandez 2009, 10; Hass 2007; Alleman 2002; Leffingwell 2007, 20-26.)

## 2.2 Agile methodologies

### *2.2.1 Characteristics of agile mindset*

The movement to agile project management methodologies has its roots in the middle of 1990. A number of software development leaders working in parallel with different development languages, different locations and different project contexts started to practice several new and different agile methodologies. Despite their differences, all the methodologies aimed at the solving same problem: creating reliable software more rapidly by eliminating unnecessary waste and unproductive overhead. (Dingsøyr et al. 2012, 1214; Leffingwell 2007, 2.)

In 2001, many of the founders of the agile software development methodologies gathered together with other experienced agile practitioners of the field to discuss what they had in common. As a result of this gathering, an agile manifesto was created summarizing the similarities and common beliefs that underlie the various methods which the group was promoting and practicing. The manifesto boosted substantially the adoption of agile in the field as it constructed and defined all the core beliefs, values and principles of the movements. (Leffigwell 1997, 7-9.) The Agile Manifesto (Manifesto for Agile Software Development 2010) states the following values:

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

While it is noted that there is value in the items on the right side of the manifesto, the items on the left side are valued more. Furthermore, the participants described number of key principles that support the agile philosophy. Having the highest priority on satisfying the customer through early and continuous delivery of valuable software, and at regular intervals reflecting on how to become more effective and adjusting the behaviour accordingly are among of them (Manifesto for Agile Software Development 2010). These principles are fully available in Appendix I.

Agile manifesto provides the philosophical understanding for the agile mindset; however, there does not seem to exist a universal definition for agility itself (Kettunen & Laanti 2008, 184). Agile manufacturing literature has contributed by several possible definitions of agility combining the basic concepts of speed and flexibility for responding to changes in turbulent market environments. Gould (1997, 28) defines agility as "the ability of an enterprise to thrive in an environment of rapid and unpredictable change". Katayama (1999, 44) identifies four underlying principles: delivering value to the customer; being ready for change; valuing human knowledge and skills; and forming virtual partnerships. Yusuf et al. (1999) suggest a broader definition for agility: the successful exploration of competitive bases (speed, flexibility, innovation proactivity, quality and profitability) through the integration of reconfigurable resources and best practices in a knowledge-rich environment to provide customer-driven products and services in fast changing market environments. As the definitions of agility vary considerably, it indicates that the concept is multifaceted and complex (Kettunen & Laanti 2008, 184).

The agile movement emphasizes the relationship of software developers as opposed to the institutionalized processes and development tools. This is manifested by close team relationships, close working environment arrangements and other procedures fostering team spirit (Abrahamsson et al. 2002, 11). Agile calls for small teams were different software development related roles, such as developers, architects, testers, form cross-functional, self-organized units which are preferably co-located (Appelo 2011, 22). Therefore, agile emphasizes the success of the whole team over individual achievements (Schwaber & Beedle 2002, 37) and empowers the teams to organize themselves how the work is carried out with accountability for their results (Appelo 2011, 22). Thus, the management of the teams has been turned upside down from bottom-up rather than top-down.

Second, the vital objective of a software team is the continuous creation of tested working software. New versions are released at frequent intervals, and developers are

urged to keep the code simple and straightforward but technically as advanced as possible in order to lessen the burden of documentation efforts to appropriate level. (Abrahamsson et al. 2002, 11.) Architecture is allowed to emerge from its basic form while developing a product (Appelo 2011, 23). Working software explicitly narrates to the development team and customer what they have in front of them as opposed to the promises what they will have (Highsmith & Cockburn 2001, 120). It also allows the measures of velocity to for the development pace, and provides fast feedback (Ibid 2011, 121).

Third, the relationship and cooperation between the developers and the customer is preferred over strict contracts; all players including customers and users are in the same team where the focus is on delivering business value immediately (Abrahamssom et al. 2002, 11; Highsmith & Cockburn 2001, 121). Thus, the need for respond to change is recognized as a key element and this challenge is addressed by short feedback loops and frequent product releases, delivering new and updated features to the users fast and thereby optimizing the business value (Appelo 2011, 23).

Fourth, the software development team, comprising both developers and customer representatives, should be competent and authorized to make possible adjustment changes emerging during the development process (Abrahamsson 2002, 11). It is believed that best products are created when customers are directly involved with the team creating them: team collaborates with the customer or its representative to maintain and continuously reprioritize the changing backlog of features (Appelo 2011, 22).

The pivotal aspects of agile methods are simplicity and speed. Hence, in software development work the team concentrates only on the functions needed at first hand, delivering them fast, collecting feedback and reacting to the received information. Abrahamsson et al. (2001, 27) identifies following characteristics distinctive for agile methods: software development is incremental (small software releases with rapid cycles), cooperative (customer and developers working constantly together with close communication), straightforward (the method itself is easy to learn and to modify) and adaptive (ability for changes).

Each agile iteration delivers working software, which is demonstrated to the business customer at the end of the iteration. The new feature set should be deployable if decided so by the customer. The phases in each iteration are nearly concurrent; hence, requirements management, design, development and testing activities are being carried out simultaneously. By collapsing the phases, the required activities can be fit into a very short timeframe (Schwaber 2007, 2.)

### *2.2.2 Differences between traditional and agile methodologies*

Agile methodologies aim to address the weaknesses of the traditional waterfall model of creating software (Petersen & Wohlin 2009, 1). The flawed assumptions of the waterfall model – well-defined set of requirements, small and manageable changes, delivering on schedule – are not assumptions of agile methods (Leffingwell, 2007, 26). In fact, agilists believe that such inherent characteristics of software development need to be embraced. Hence, agile based processes address the challenge of an unpredictable world by relying on people and their creativity rather than on rigid processes (Dybå & Dingsøyr 2008, 835).

Agile methodologies focus on pursuing the highest business value to the customer faster at any point of time (Eckstein 2004, 13) by creating working software in every iteration based on the requirements prioritized and ranked by the customer (Leffingwell 2007, 14). Hence, the traditional software development curve assuming the value only at the end of the lifecycle after releasing the software for the first time is changed through the agile planning and practices (Figure 3) enabling constant delivery of working software and, thus, faster return on investment.

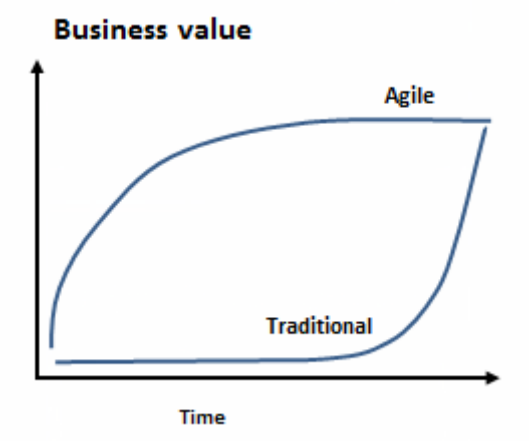


Figure 3 Comparison of traditional and agile development in relation to and business value

Agile principles and practices, such as requirement backlogs and short feedback loops manifested by sprint review and planning meetings facilitate the constant visibility of the planned work and work already done (Abrahamsson et al 2002, 29, 33). In the traditional approach, the lack of feedback loops hinders the visibility to the investment which is often gained back fully once the software development life cycle is proceeding in the last steps as illustrated in the Figure 4.

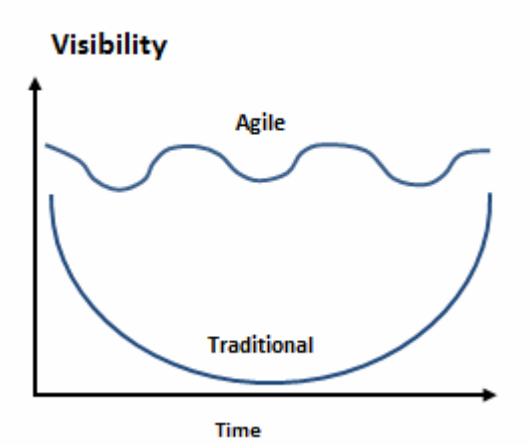


Figure 4 Comparison of traditional and agile development in relation to visibility of the development progress

The traditional notion and experience of the cost of correcting errors, misunderstandings and changing software has been on a substantial growth over time as illustrated in the Figure 5 on the left hand side. Possible reasons for even exponential growth for cost of change are in early design flaws or fatal errors in applications already deployed. agile assumes the traditional cost-change relationship can be modified by specific practices which shape the cost curve accordingly as shown in Figure 5 on the right hand side. Extensive upfront planning and analysis is changed to small increments of working code, continuous testing and short feedback loops in order to minimize the cost of change. (Leffingwell 2007, 31-33.)

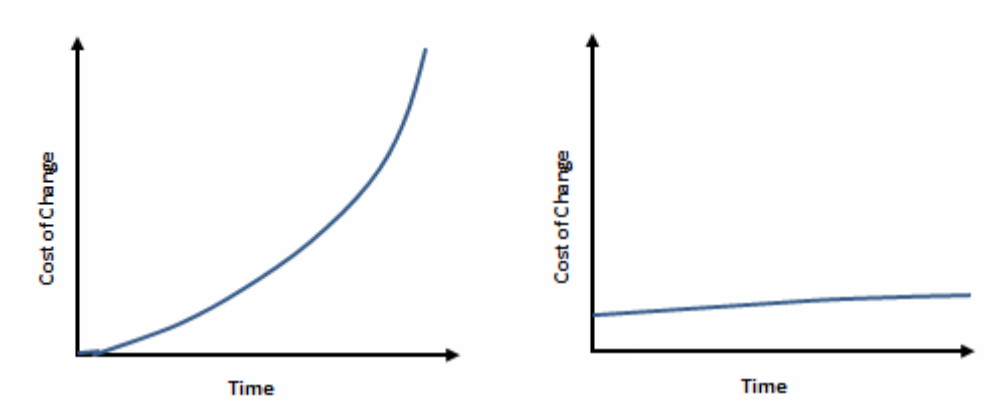


Figure 5 Traditional assumption for cost of change over time vs. agile assumption for cost of change.

The objective of the agile approach is to ensure risks are detected and eliminated early during the software development process (Eckstein 2004, 13). This is to be gained by iterative and incremental development process, providing feedback that allows neces-

sary corrective actions immediately. The development cycles are short to ensure frequent feedback and time-boxed implying that development time frame for each development cycle is fixed. (Eckstein 2004, 13-15.) One view on the risk management is that agile trades off explicit knowledge captured in documentation for tacit interpersonal knowledge in order to achieve more speed while bounding risk (DeMarco & Boehm 2002, 90).

In agile development planning phase it is assumed that schedule and resources as planning parameters are fixed (as presented in Figure 6). The final parameter, requirements (scope), is the only one which is kept variable and can change. Hence, at the end of the development cycle the final produced release may contain more or less functionalities than planned, but the development still ends at the determined time and with agreed resources (Eckstein 2004, 13-15). This agile tenet creates a new mindset compared to traditional approaches as it is not known exactly what will be delivered and when (Leffingwell 2007, 68).

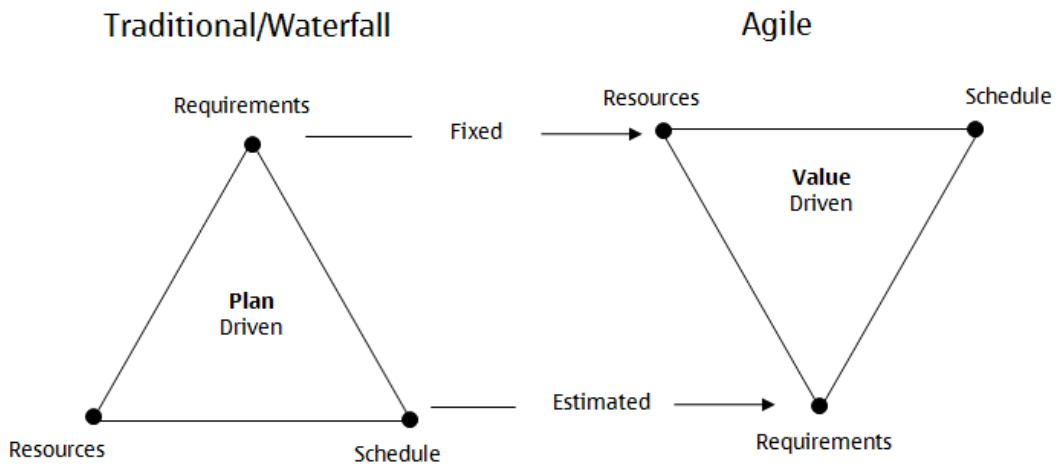


Figure 6 Plan-driven traditional project management versus value-driven agile methods (modified from Leffingwell 2007, 81)

Traditional waterfall development approach emphasizes large up-front planning and analysis with work breakdown structures as a measure of success (Table 1) whereas agile development focuses on delivering value by implementing requirements on a prioritized basis. Procedural and documentation stage gates of traditional approach are replaced with success measures based on working, tested and demonstrated code in agile approach. The plan in agile is flexible and fluid in order to accommodate changes.

Table 1 Differences in waterfall and agile development (adapted from Leffingwell 2007,77 and Pelrine 2011, 28).

| Process | Traditional (Waterfall) development | Agile development |
|---|---|---|
| **Measure of Success** | Plan-driven | Results-driven (working code) |
| **Management Culture** | Command and control | Leadership/collaborative (self-organized teams) |
| **Requirements & Design** | Well-defined, up-front | Continuous/emergent/just-in-time |
| **Planning & Scheduling** | Detailed/fixed scope/time and resource estimates | Two-level plans (release/iteration), estimate scope |
| **Coding and Implementation** | Code all features in parallel, test at the end | Code and unit test, deliver serially |
| **Test and Quality Assurance** | Big-bang/planned/test late | Continous/concurrent/test early |

Command and control type of management peculiar to traditional, plan-driven development approach is seen insufficient to manage complex ecosystems (Highsmith 2000, 199). Sense-making, that is, how one perceives the world, is affecting the approach to management. If the outside world is perceived as turbulent and unpredictable, the approach to management is said to be leading to different set of management practices than if the world is sensed as stable and predictable. The practices have been described as participative and human-centered where leadership replaces command and collaboration replaces control. (Ibid 2000, 205; Appelo 2011, 58-59; Schwaber & Beedle 2001; 68-69.) In order to battle complexity, leaders aim to create collaborative environments in which solutions emerge, team is self-organized and every member is accountable, ideas flourish, risks are taken and mistakes viewed as learning opportunities (Highsmith 2000, 210; Leffingwell 2007, 77). Hence, in agile team roles are not any more fixed, but members are allowed to self-organize (Schwaber & Beedle 2002, 113).

Approaches towards requirements and design are also different in agile compared to traditional methodologies. Instead of detailed requirements specifications or architectural models, which often take long time to create, agile focus on delivering fast value-added requirements into an integrated baseline. Early delivery of working software serves to test the requirements and architectural assumptions. Until the integrity is achieved, refactoring combined with constant user feedback and visibility is carried out. (Leffingwell 2007, 78).

Coding and implementation in agile based processes differs from traditional approaches. Instead of releasing all the functionality as a big bang only at the end of the development cycle, in agile the highest priority items are worked on first to fully functional tested, working, integrated software. The releasable software is demonstrated to the customer at the end of each iteration for immediate and continuous feedback. (Ibid 2007, 79.)

Traditional methodologies assume testing as an individual life cycle phase of the development. In agile it is a continuous activity. Systems with new code that have passed unit tests are given to the testers who often increase their skill levels as they also participate in design decisions via testing. Due to the continuous nature of the testing, development of test automation is common. (Leffingwell 2007, 79-80.) In agile development process planning and scheduling take place in two levels and happens more frequently than in traditional methodologies. Upfront planning takes place in two levels: for each release and more fine-grained for each iteration. Hence, the planning in agile is continuous and more intense compared to one-time planning in traditional approach. (Leffingwell 2007, 80.) For example, in Scrum methodology the daily status meetings and demonstrations for each iteration facilitate the tracking, and the meetings act also as micro planning sessions (Schwaber & Beedle 2001, 42-43). The features of different agile methodologies are discussed further in the subchapter 2.2.4.

### *2.2.3 Complex adaptive systems explaining agile*

In order to contemplate the possible challenges in agile transformation, it is useful to first create understanding of the underlying reasons why agile processes are claimed to work. Complexity science, which describes a set of interdisciplinary studies (Holland 1998, 8), offers a theoretical perspective for this.

Today's software development is performed by teams of motivated individuals (Pelrine 2011, 28). Several agile experts and authors have claimed that complexity theory represents software development more realistically than the traditional engineering models (Highsmith 2000, 8; Schwaber 2002, 89-90; Pelrine 2011, 31; Appelo 2011, 46). In dynamic and unpredictable world adaptation of the organization, in other words the ability to rapidly adjust behavior to the dynamics of the environment, is seen more important than stability (Rzevski 2004, 1). Approaching software development as an adaptive process and viewing the development team itself as a living organism is argued to provide better model for managing software creation initiatives (Highsmith 2002, 11). Hence, complex adaptive systems (CAS) theory can provide a theoretical lens for understanding how agile development can be used in volatile business environments (Meso & Jain 2006, 19).

A software development team or an organization is considered as a complex adaptive system as it is formed of multiple interacting parts within a boundary with a capacity to change and learn from experience (Highsmith 2008, 11; Appelo 2011, 33). According to Rzevski (2009, 1) and Stacey et al. (2000, 106), the characteristics of complex adaptive systems include large number of different agents engaging in rich interaction with each other. The agents are largely autonomous but they are subject to certain rules or norms.

The behavior of complex system emerges from the interaction of agents and is therefore unpredictable. The systems are far from equilibrium because of frequent occurrences of disruptive events which do not allow returning to equilibrium, and the systems are capable to self-organize in response to those disruptive events and evolve over time. Hence, the key components of complex adaptive systems are agents, environments and emergent outcomes (Highsmith 2000, 15). From the perspective of software development and social complexity science team members represent the agents, environments the market economies, and emergent outcome the innovation, for instance, new software, created from the interaction of the system parts (Figure 7) that is, of the self-organized agents (Highsmits 2000, 29-31; Rzevski 2009, 2). Emergence refers to the notion that the whole is greater than sum of its parts; rules that are almost absurdly simple can generate coherent, emergent phenomena (Holland 1998, 2; Schwaber & Beedle 2000, 115). For example, an agile rule of not to change the product backlog during an iteration has deep repercussions. The team is undisturbed since its goal is respected; hence, its credibility is easier to maintain, which results to higher level of trust from the customer side. Therefore, this rule results in the emergence of a more stable customer - team relationship. (Schwaber & Beedle 2000, 115.)

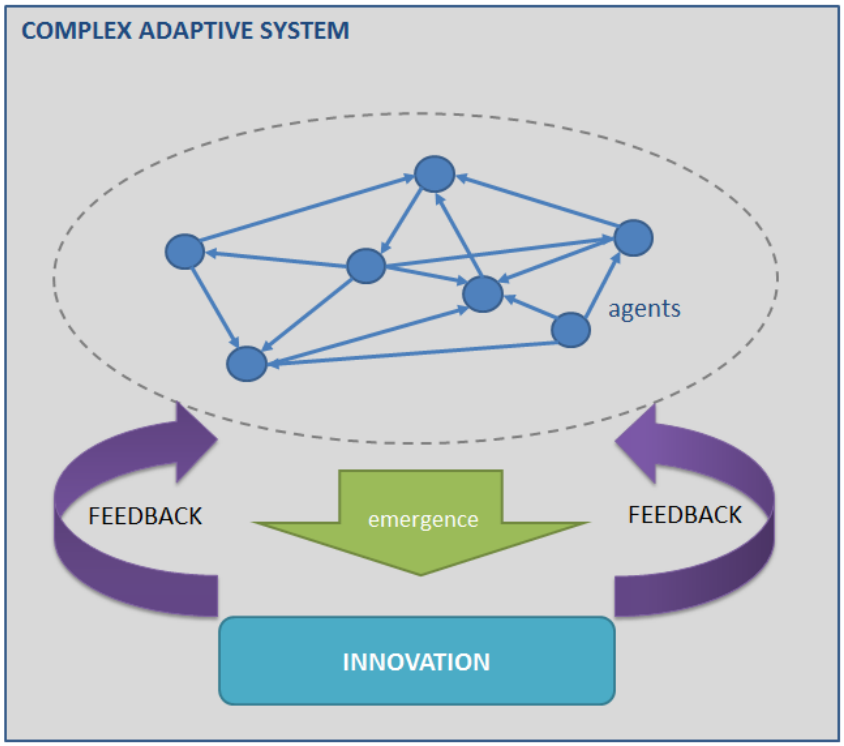


Figure 7 Complex adaptive systems (modified from Appelo, 22)

Dave Snowden's Cynefin sense-making framework, an approach to social complexity science, aims to clarify the issue of solving complex problem, such as software development, further by offering approaches to decision making under uncertainty (Kurtz &

Snowden 2003, 468; Snowden 2007, 72). Cynefin is a Welsh word that signifies the multiple factors in the surrounding environment and the experience that influence individuals in ways that cannot ever be understood (Snowden 2007, 72). An aspect what makes complex systems complex is causality. Cynefin model identifies five domains of system behaviors that require different actions as illustrated in Figure 8.

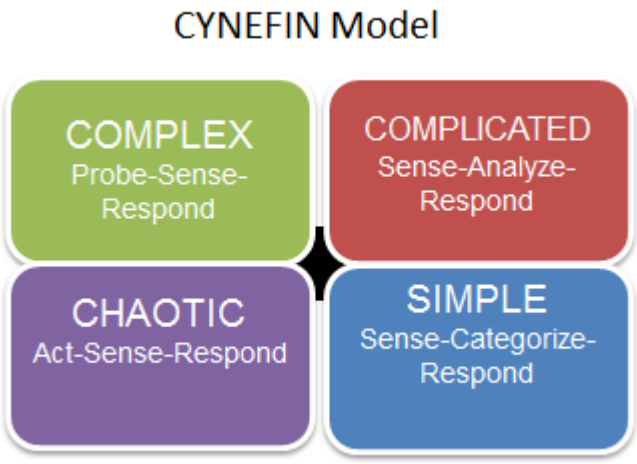


Figure 8 CYNEFIN sense-making model (Snowden & Boone 2007, 72)

Four of the domains are named; fifth domain is in the central area and signifies the domain of disorder. In simple and complicated contexts, forming the ordered domains, cause and effect relationships are predictable and generally linear, and right answers can be determined based on facts. The decision model in the complicated context is to sense incoming data, analyze the data and then respond in accordance with expert advice or interpretation of that analysis. (Kurtz & Snowden 2003, 468; Snowden 2007, 72-74.)

Complex and chaotic contexts of the Cynefin model refer to unordered behaviour which is unpredictable, that is, there is no immediately apparent relationship between cause and effect, to a small or large degree. The way forward is determined based on emerging patterns. In complex domain patterns emerge through the interaction of several agents. Thus, it is insufficient here to rely on expert opinions based on historically stable patterns of meaning. (Kurtz & Snowden 2003, 469; Snowden 2007, 72; Appelo 2012, 43.) Therefore, in a complex system, causality emerges as the system emerges resulting that only in the end one can say how the end result was achieved, but following exactly the same steps again, the result may not be the same. Thus, in complex systems causality is retrospectively coherent. (Pelrine 2011, 35.) A classic example of retrospective coherence is task estimation. Only afterwards completing the task one can say how long it took and what where the reasons for it.

To maximize flexibility in the face of uncertainty of complex issues, the Cynefin model suggests a probe-sense-respond technique as a model for decision-making. Creating first numerous probes inside the boundaries set for the system to emerge will provide feedback on what works and what does not. After that, applying sense-making to the results of the feedback – continue or amplify those probes that worked best and change or disrupt those that did not. (Pelrine 2011, 36; Kurtz & Snowden 2003, 469.) This decision model is in fact similar to the apply-inspect-adapt continuous improvement (see Chapter 2.2.5) model of agile development (Schwaber & Beedle 2002, 117-118), used also as a mechanism of learning in agile (Meso & Jain 2006, 27). Hence, by identifying that software development domain is complex by its nature having too many uncertain aspects, one is not able to predict the behavior of a system in advance, which creates the challenge to envision in advance via pre-determined feature lists how to create a successful outcome. Therefore, different approaches, such as probe-sense-respond technique, can help to continuously improve and adapt to the changing environment as one learns what works and what needs a change.

### *2.2.4 Overview of different agile methodologies*

Several different agile methods are contributing to the iterative, evolutionary and incremental approach of agility (Larman & Basili 2003, 47). Scrum and Extreme Programming (XP), or a hybrid of these two, are the most commonly used agile methods (Schwaber 2007, 2; VersionOne 2011). Scrum is more focused on project management techniques (Abrahamsson et al. 2002, 31) whereas XP emphasizes engineering practices, such as test-driven development and continuous integration (Beck 1991, 70). Thus, these methods can form an integrated package for software development where one can complement the other (Abrahamsson et al. 2002, 36). Scrum is discussed in detailed in Chapter 2.2.5.

XP has evolved from issues caused by long development cycles of traditional development methods (Abrahamsson et al. 2002, 18; Beck 1999, 70). It first started simply as an opportunity to get the job done with practices that had been already found effective in software development processes during the preceding decades. After number of successful trials key principles and practices were collected together. Although the practices themselves were not new, however, the way they have been collected and lined up to function with each other was novel. The term extreme refers to taking the common-sense development principles and practices to extreme levels. (Abrahamsson et al. 2002, 19.) The key practices of XP include following: a small team of programmers work at one location with customer representation on site, and development occurs in frequent

iterations which can be released even daily and which deliver incremental functionality. Refactoring means that the system is evolving through transformations of existing design that keeps all tests running. In pair programming two people sit side by side and write the code. With continuous integration new code is integrated with current system on a daily basis and all the automated tests must be passed. Collective ownership refers to a responsibility that each programmer should be able to improve any code anywhere in the system at any time if they see the opportunity. (Beck 1999, 71; Leffingwell 2007, 13.) Rather than planning, analyzing and designing for the future, XP does these activities a little at a time, throughout the software development (Beck 1999, 71).

Dynamic Systems Development Method (DSDM) is a framework focused on delivering a quality solution quickly (Leffingwell 2007, 65). The fundamental tenet of DSDM is the belief that it is not possible to fix the amount of functionality in a product. Instead, it is preferred to fix time and resources and then adjust the amount of functionality accordingly (Abrahamsson et al. 2002, 61; Leffingwell 2007, 67). DSDM has its roots in rapid application development (RAD), therefore, the distinctive characteristic of DSDM is that it supports the use of extensive prototyping (Abrahamsson et al. 2002, 89; Leffingwell 2007, 69). Other key practices of DSDM include modeling for architectural underpinnings, strong focus on testing throughout the development lifecycle and configuration management to support the rapid pace of coding (Leffingwell 2007, 67-68).

Feature Driven Development (FDD) shares some of the best practices affiliated to agile. Compared to XP and Scrum, FDD was developed and refine to be suitable for systems of large scale and scope. Domain object model is the first step in FDD which serves as the high-level architectural concept for all following development. Development is organized around the implementation of a small-size features or a use case. Smallest implementation unit, a class, should have one owner who is ultimately responsibility for its development and maintenance. Basic team organizational structure in FDD is around feature teams. Code quality assurance relies heavily on inspection, builds are created regularly and configuration and version control is recommended for all system related artifacts, also documents. Project status monitoring per iteration takes place via assessing the state of each feature via six step milestones covering design and build activities. (Leffingwell 2007 71-73.)

Agile processes have much in common with the lean processes which originated in the field of manufacturing. The core ideas of lean development, which has its roots in Toyota Production System created in 1950s (Ohno 1998, xiv), emphasized waste elimination, achieving quality first time and focus on problem solving (Dybå & Dingsøyr 2008, 836). Similarly agile software development methods are characterized by strong focus on eliminating waste. Thus, there is a growing interest in identifying ways to combine principles of lean development with software development and therefore, enlarge the previously dominant view of lean mentality with a view to minimizing unnec-

essary work, especially in creation of wasteful documentation (Dingsøyr et al 2012, 1213 – 1218).

Poppendieck & Poppendieck (2003, 179) have described the lean thinking adapted to agile software development and formulated seven lean principles which can be translated to agile practices. First, focus on eliminating waste that does not add any value to a product. The waste in software development can be for instance partially done work, extra processes adding efforts for extensive documentation, extra features added to the system just in case, task switching especially if one person is assigned to multiple projects, waiting time, defects, or motion efforts required for solving problems (Ibid 2003, 8). Second, amplify learning in order to support the development effort which is often a journey of discovery instead of a production process. Third, deciding as late as possible enables decision-making based on facts instead of speculation. Key strategy for delaying decisions in complex software development is to build a capacity for change into the system. (Poppendieck & Poppendieck 2003, xxv-xxvi.)

Fourth, delivering as fast as possible enables delaying the decision-making and continuous reliable feedback. The shorter the development cycle - in other words a compressed value stream - the more can be learned also concerning requirements of the customer. Fifth, empowering the team enables successful execution: late decisions and fast execution do not leave room for activity orchestration by a central authority. Sixth, building integrity in to the system enables its usefulness over time. This is to be achieved via leadership, relevant expertise, effective communication and discipline. Lastly, as seventh principle, seeing the whole means a deep expertise and understanding in diverse areas in order to build the integrity in and avoid sub-optimization especially in large organizations where multiple organizations are developing the same system. (Poppendieck & Poppendieck 2003, xxvi-xxvii.)

Kanban, a software development framework adapted from lean principles (Kniberg & Skarin 2010, vii), has recently increased its popularity in the field among the practitioners (VersionOne 2011). In addition to Kanban, also hybrid models of Kanban and Scrum, so called Scrumban, have increased their popularity (Ibid 2011). The underlying conception of Kanban is to limit requirement items that are in the state of work in progress. New work should be started only when an existing piece of work is delivered or pulled by a downstream function. Kanban (or a signal card in lean) implies that a visual signal is created to indicate that new work can be pulled as current work does not equal to the agreed limit. Kanban is an approach to introduce change management to an existing software development lifecycle or project management methodology. A visual control mechanism, such as whiteboard with sticky notes or electronic equivalents, is used to track work as it flows through the various stages of the value stream. As tasks, which are in the state of work in progress, are limited, Kanban exposes bottlenecks, queues, variability and waste in the process. Thus, Kanban provides input for performance im-

provements and also visibility into the effects of actions taken to improve the performance. (Kniberg & Skarin 2010, viii.)

It is worth noting that organizations can create different agile practices or tailor existing agile practices to address specific organizational or team needs. Furthermore, teams may also need to be creative and come up with new agile practices to achieve agility while complying with the organizational constraints. (Smith & Sidky 2009, 9.) Translating agile principles into practices that match the organizational setting is essential (Poppendieck & Poppendieck 2003, 180).

### *2.2.5 Scrum as agile software development framework*

Fundamentally, Scrum methodology is a project management framework to manage complex projects. The roots of Scrum date back over two decades. The term Scrum was first used in a development context to describe an adaptive, quick, self-organizing product development process which was developed in Japan due to in response to the required competitiveness brought on by globalization. (Schwaber & Beedle 2002, 1.) The term 'Scrum' refers to a rugby play for getting an out-of-play ball back to the play. Thus, there is a similarity between the rugby game, which is highly coordinated and reactive team sport, and the new type of product development process described by Scrum. (Leffingwell 2007, 42.) Many of the primary Scrum practices were originally developed by Ken Schwaber and Jeff Sutherland in 1994. As Scrum was thereafter officially formalized, many others have extended and enhanced it in several companies and organizations. (Leffingwell 2007, 41; Schwaber & Beedle 2002, 20.)

The Scrum approach has been developed for managing the systems development process (Abrahamsson 2002, 27). Scrum applies the ideas of industrial process control theory to systems development which results in an approach that reintroduces the ideas of flexibility, adaptability and productivity: regularly inspecting activities to see what is occuring and empirically adapting to produce desired and predictable outcomes (Schwaber & Beedle 2002, 100).

Leffingwell (2007, 41) describes Scrum as "a lightweight agile project management method based on small, empowered, self-organizing teams; complete visibility; and rapid adaptation". Further on, Appelo (2011, 47) highlights the perspective to complex adaptive systems: "Scrum is not a methodology, a defined process, or set of procedures. It is an open development framework. The rules are constraints on behavior that cause a complex adaptive system to self-organize in to an intelligent state." Hence, key characteristics of Scrum include small cross-functional teams which work closely together in an open environment to produce incremental releases of products in predefined fixed-time increments called sprints.

Scrum teams are highly self-directed and empowered to meet the objectives of the sprints (Schwaber & Beedle 2002, 28). Scrum does not provide or require any specific software development practices to be used (Abrahamsson et al. 2001, 31). Instead, it requires certain management and leadership practices and tools in the different phases of Scrum to avoid the chaos caused by unpredictability and complexity (Schwaber & Beedle 2002, 31). The management of a Scrum-based process can be defined as intensive, prescriptive and role-based (Leffingwell 2007, 42). There are only three main roles prescribed:

- The *Product Owner* who is responsible of representing the interest of the customer and key stakeholders on the team. This is achieved by managing the product backlog which is a prioritized list of requirements and other work to be done by the team.
- The *Scrum Master* is responsible for helping the team to achieve its goal by removing any impediments, coaching team in the use of Scrum, interacting with the project team and the customer/Product Owner, and ensuring the implementation of Scrum practices.
- The *Scrum Team* is responsible for implementing the functionalities. The team is self-organized and cross-functional, and its members include developers, testers and any other personnel required to fully implement and deliver the functionalities. The team contributes to the process also by participating effort estimations, reviewing the product backlog list, creating the sprint backlog and suggesting impediments to be removed. (Abrahamsson 2002, 31; Leffingwell 2007, 42; Schwaber & Beedle 2002, 31-36.)

Development cycles are divided up into 1-4 week periods called *Sprints*, at the end of which a potentially shippable product will be ready. Figure 9 illustrates the overview of the Scrum iteration process in a high level. At the beginning of each *Sprint*, the development team selects a fixed set of requirements to work on from a prioritized list. This list is called *Product Backlog*. The *Product Backlog* is constantly evolving and prioritized list of business and technical requirements for the system to be built or enhanced. Initially the first backlog can be generated from the product vision, and listed items can include for instance features, functions, bug fixes, defects, requested enhancements, technology upgrades and even issues requiring solutions before other backlog items can be worked on. Multiple actors can contribute in generating *Product Backlog* items, such as customer, *Scrum team*, other business units and customer support, but *Product Owner* has the responsibility for maintaining it. (Abrahamsson 2002, 32; Schwaber & Beedle 2002, 33-34.)

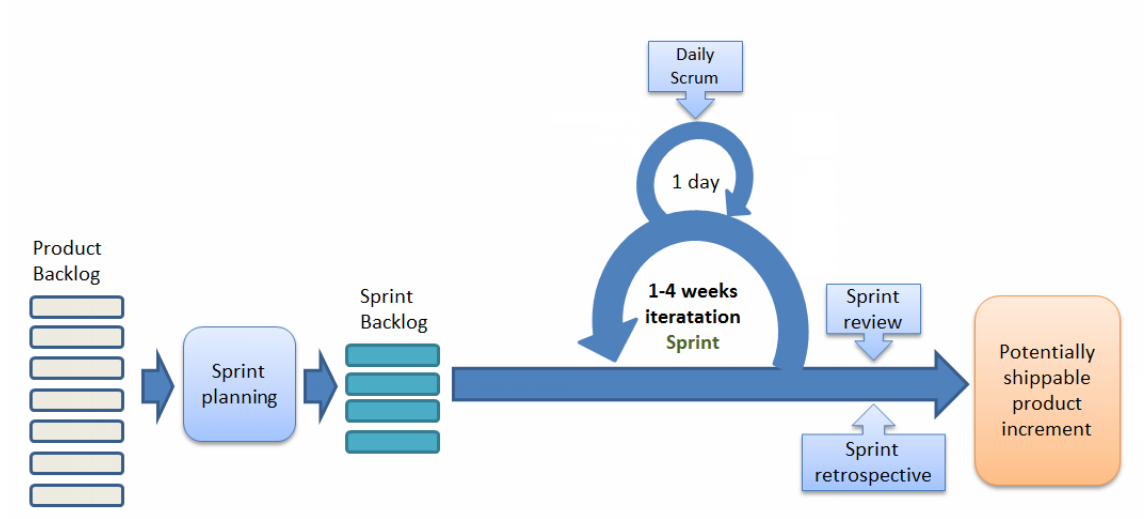


Figure 9 Scrum development process overview (modified from Schwaber & Beedle 2002, 8)

*Sprint Planning* meeting consists of two consecutive meetings organized by *Scrum Master*. In the first phase of the meeting, *Product Owner*, management, *Scrum team* and users decide the goal and the functionalities to build in the next sprint. *Product Owner* presents the top priority items and leads the discussion on identifying what the team can develop during the next iteration. In the second phase of the meeting, *Scrum Master* and the *Scrum teams* focus on planning how the new product increment will be implemented, and thus, how the *Sprint* goal is to be achieved. (Abrahamsson 2002, 33; Schwaber & Beedle 2002, 47-48.)

As a result of the *Sprint Planning* meeting, *Sprint Backlog* list is established of the *Product Backlog* items selected to be implemented in the next *Sprint*. The team self-organizes to assign and undertake the work in the *Sprint Backlog*. Unlike *Product Backlog*, the *Sprint Backlog* remains stable until the *Sprint* is completed (for instance 30 days). *Sprint Backlog* is a visible, real time picture of the work that the team plans to accomplish during the *Sprint*. In order to keep continuously track of the progress of the *Scrum team*, *daily Scrum meetings* are organized by *Scrum Masters*. In addition to communication tools, the meetings serve also as small scale planning opportunities. The agenda of the *daily Scrum meetings* follows always the same format: what has been done since the last meeting, what is going to be done before the next one, and are there any impediments in the development process or in the practices which would require mitigation. The daily meeting is kept short (approximately 15 minutes). Besides the *Scrum team* also others, for example, management can participate in the meeting. (Abrahamsson 2002, 33-34; Schwaber & Beedle 2002, 40-50.)

On the last day of the *Sprint*, the *Scrum team* and the *Scrum Master* demonstrates the results, that is, the potentially shippable product increment of the *Sprint* to the manage-

ment, customers, users and the *Product Owner*. The *Sprint Review* is an informal meeting where the participants assess the product increment and make a decision how to continue the development. The purpose of the meeting is informational and discussion is encouraged. Therefore, the review meeting may bring out new items for the *Product Backlog* and even change the direction of the system being built. (Abrahamsson 2002, 34; Schwaber & Beedle 2002, 54-56.)

Each agile method addresses quality in certain ways. Scrum uses intensive 15-minute *daily Scrum meetings* and comprehensive *Sprint Reviews* at the end of each iteration cycle (Highsmith & Cockburn 2001, 120). The *Sprint Review* session incorporates a *Sprint Retrospective* part which provides key learning opportunity and tactical possibility for empirical process control mechanism (Schwaber & Beedle (2002, 100-101). Quantitative metrics, such as number of defects or progress on test automation during the iteration, can be tracked. Qualitative assessment includes discussion of what went well in the iteration and what should be improved. (Leffingwell 2007, 182-183.) As software development is claimed to be inherently a complex and unpredictable process (Schwaber & Beedle 2002, 100) in those cases frequent adaptation must be made to bring a process back into the range of acceptable (Leffingwell 2007, 47). Based on the evaluation of the results, actions are assigned and decisions made for further improvement (Leffingwell 2007, 183). Hence, Scrum implements constant inspect-adapt cycles by its ceremonies.

### *2.2.6 Scaling Scrum to a large organization*

Laanti (2012, 85-89) notes that successful use of agile methods in large-scale environments requires extensive understanding of agile organization, that is, to have organizational flexibility in an enterprise. Scrum emphasizes cross-functional and self-organizing teams. The processes, the ways of working and the organization itself with its policies define the degree of such flexibility in the organization. The more these agile aspects are stretched, the more the organization can benefit from its flexibility. (Ibid 2012, 89.)

Agile methods were originally developed in small team environments where freedom to explore and innovate prevailed and where resources and possible issues could be managed in a level of a single team (Leffingwell 2007, 88). If an organization implements agile methods only on the team level, the full benefits are restricted to be captured only on a team level which may not improve the overall product development time and thereby, the improvement to time-to-market. Hence, optimizing the speed of delivery only in parts of the system will not result in increase of the output, but can, in fact, lead to increased waste. (Laanti 2012, 89-90.) At scale, specific requirements such

as those that define the performance, reliability and scalability of the system, must be understood early on by all the teams contributing to the development of a solution in order to avoid extensive refactoring. Yet, in order to be agile, this should be done while the investment in early requirements definition is substantially reduced. (Leffingwell 2007, 191.)

According to Eckstein (2004, 43) scaling agile to larger organizations comprising several individual teams even in dispersed locations requires additional focus on people, interactions and communication structure. In the people aspect the ability to take responsibility requires additional focus. Eckstein (2004, 47) highlights the key aspect of whole agile philosophy: the value of team productivity is more important than the individual effort entailing that team's success is the individual's success. Thus, leadership as a management model emphasizing responsibility, trust and teamwork is needed over command and control. Eckstein (2004, 47) notes that trust is the foundation on which such management strategy is built.

Leffingwell (2009) presents a delivery process model for large-scale agile development (Figure 10) in which the full software development lifecycle from project inception to transitioning the system the production or marketplace environment is covered. This model is also referred to as a Scaled agile Framework (Scaled Agile Framework 2014).

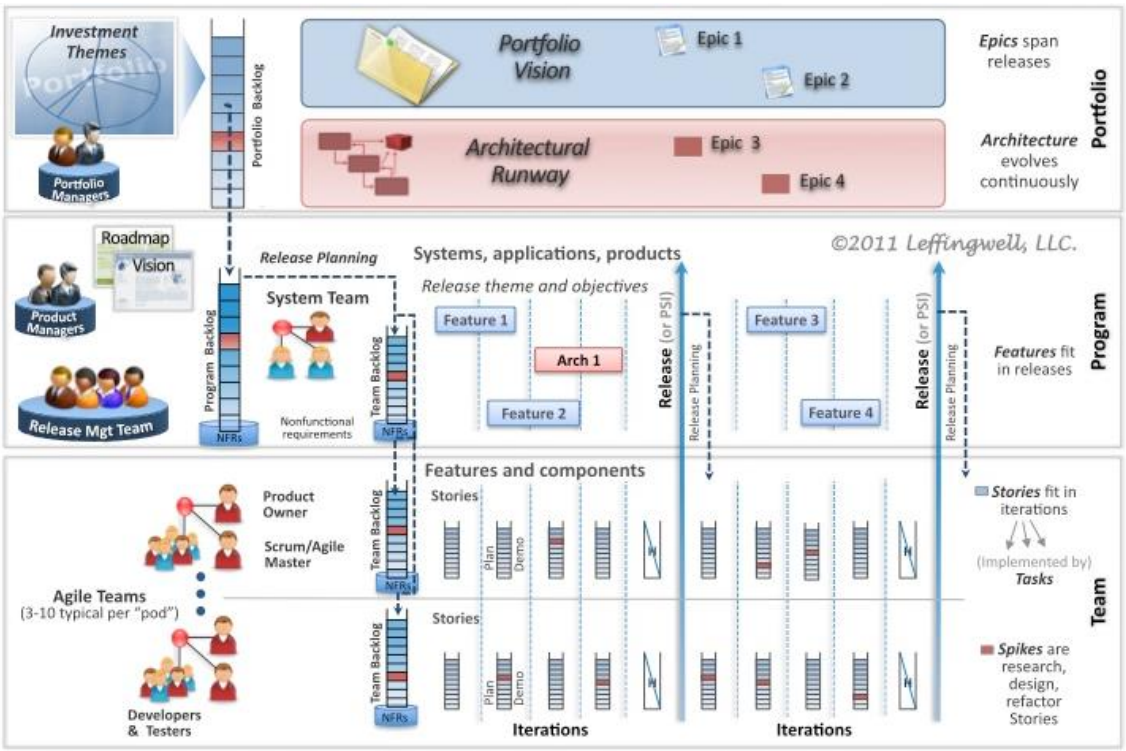


Figure 10 Scaled agile delivery model (Leffingwell 2009, 4)

The model suggests to utilize a lean requirements pattern that has three main elements: a *vision*, a *roadmap* and *just-in-time elaboration* (Leffingwell 2007, 191). At the portfolio management level, Leffingwell (2009, 4) discusses a mix of investment themes, which drive the investment priorities for the organization. In practice this means the required, necessary work to be performed in order to execute and deliver the chosen business strategy. Investment themes drive the portfolio *vision* which is expressed as epics in Figure 9, referring to initiatives or often projects that will be factored into upcoming *releases*. The release roadmap illustrates on a high level that the vision is to be implemented over time in accordance with a prioritized backlog. (Ibid 2007, 191.) At the program level the large scale system development is attained by multiple teams working in synchronized agile release train which means a standard agreed cadence of time-boxed development iterations and releases. The principle of agile release train is to 'leave the station on time': if dates, themes and quality are fixed, then functionality is the only variable. (Ibid 2007, 191.)

The agile teams, typically maximum of 7-9 members, responsible of software development, collaborate on building the larger system. An agile team includes all the roles necessary to define, build and test the software for a particular component or a feature. The roles can include Product Owner, Scrum Master, technical lead and a small team of developers and testers. These teams are typically organized around a software component or a feature. Other already previously noted characteristics of agile apply to these teams also, for instance, ability to self-organize and reorganize continuously based on the work in the release backlog. (Leffingwell 2007, 4, 191.)

Leffingwell (2009, 5) notes a separation of product requirements responsibilities at the portfolio, program and team levels. Typically Product Owner role is responsible for requirements management on a team or project level, program manager for the program level and portfolio manager for the portfolio level. As the requirements move through the value stream, these roles have the overall accountability of managing the requirements. Different requirement artifacts – epics, features and user stories – are used in describing the system at the different levels. Leffingwell argues (2009, 5) that these labels help to control the abstraction level used by the people in the roles limiting early overly detailed specification, aiming to reduce work in progress and freeing teams to implement the intent the way it best suits for the context and their knowledge.

Releases and release increments (ie. potentially shippable working software) are frequent and typically fixed-scheduled for 60-120 day maximum duration boundaries. Leffingwell's scaled agile delivery model (2009, 5) identifies also two additional teams to manage the releasing of a product to the end user which is often rather complex process. A system team and a release management team at the program level are commonly responsible of aforementioned activities: system level testing and release readiness assessment as well as release content management and deployment. Implementing the

aforementioned scaled delivery model relies on the organizational surroundings which impact is discussed next.

## 2.3 Impact of organizational culture to agile transformation

Organizational culture, a perspective of organization theory, has been regarded as a metaphor for understanding organizations (Brown 1998, 9-10). Schein (1991, 26) defines organizational culture by two patterns: the pattern of basic assumptions which are invented, discovered or developed by a given group as it learns to deal with its challenges of external adaptation, and by the pattern of internal integration which has worked well enough to be considered valid and therefore, is to be taught to new members of the culture as a correct way to perceive, think and feel in relation to those challenges. Eldridge and Crombie [1] value also the patterns of behavior as equally important (Brown 1998, 9). Hence, the essence of organization revolves around the development of shared meanings, beliefs, values, and assumptions that guide and are reinforced by organizational behavior (Jaffee 2001, 165). Schein (1991, 31-37) identified three levels of culture in his model of organizational culture: first, artifacts, such as written documents, physical layouts, language, behavioural patterns and rules; second, espoused values meaning beliefs, values and attitudes of the organization; third, basic assumptions concerning taken-for-granted ways of doing, thinking and achieving goals. Through the medium of culture one can make sense of the world and by means of organization culture one can comprehend and associate meaning to its organizational experiences (Brown 1998, 33).

Similar to organizational culture, organizational culture change does not have a definitive widely accepted model; different authors have different notion of culture in mind (Brown 1998, 116). Sathe (1983) has suggested that there is an important distinction between behavioural change and cognitive change. It is argued that when an organization undertakes a change initiative there can be three basic outcomes. First, there can be a cultural or cognitive change on individual level (for example, values and beliefs) without a change in their behavior. This could be a result when individuals agree that, for instance, new ways of working are beneficial, but find it difficult to adopt them because of ingrained habits or lack of relevant skills and knowledge to put them into effect. (Brown 1998, 117; Sathe 1983, 15-17.)

---

[1] Original source: Eldridge, J.E.T and Crombie, A.D. (1974) *A Sociology of Organizations.* Georg Allen & Unwin Ltd, London.

Second, change at the behavioural level may not be complemented by a change at the cultural level. This could be possible, for instance, when compliance with new organizational rules and procedures are enforced by some mean without employee enthusiasm. Third, there can be a change at both the behavioral and cultural levels. This is a change which can lead to permanent change as people both genuinely believe and value their new way of working which makes the new system both self-sustaining and mutually reinforcing. (Brown 1998, 117; Sathe 1983, 15-17.) Hence, change is enduring only when it becomes "the way things are done here" in the work unit. Until new behaviors are rooted in social norms and shared values, they are always prone to degradation as soon as pressures associated with the change effort are removed. (Kotter 1996, 14.)

According to Kotter (1996, 14) particularly consequential in anchoring new approaches into an organizational culture is a deliberate attempt to show people how specific behaviors and attitudes have contributed to improved performance. Kotter (1996, 14-21) notes also that anchoring change requires current management, as well as the next generation of management, to embrace the new approach in its decision making to make sure the transformational efforts are not undermined. His eight stage process for transformational change management (Kotter 1996, 21) focus on mitigating the errors often associated with such efforts as illustrated in Figure 11. Kotter (1996, 20) argues that a successful change tends to be associated with a multistep process that creates sufficient power and motivation to overwhelm the sources of inertia and that key enabler for employing such a process effectively is high-quality leadership instead of excellent management.

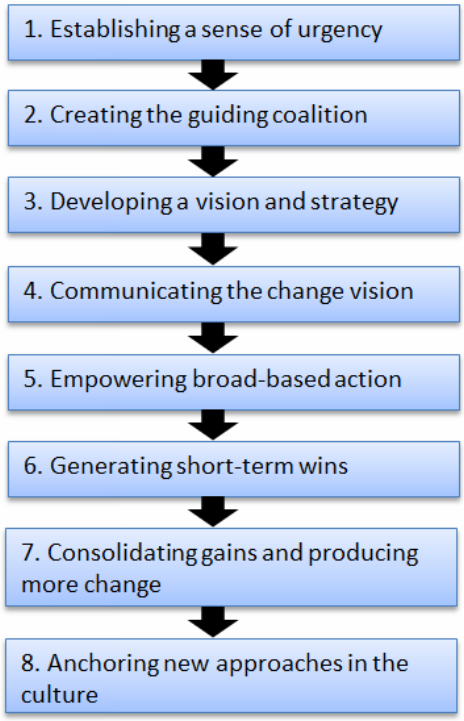


Figure 11 The eight-stage process of creating major change (Kotter 1996, 21)

The first four phases of the process focus on facilitating the path towards the change: establishing a sense of urgency, creating the guiding coalition with enough power to lead the change, developing a vision and strategy to help direct the change effort and achieving the vision, and lastly, communicating the change vision including strategies where the guiding coalition acts as role model for the behavior expected of employees. The phases from five to seven introduce new practices: empowering broad-based action in order to get rid of obstacles, changing structures that undermine the change vision and encouraging non-traditional ideas and actions. Kotter (1996, 20-23.)

The process continues by generating short-term wins: planning for visible performance improvements, creating these wins and visibly recognizing and rewarding the ones making the wins possible. The phase seven focus on consolidating gains and producing more change: using increased credibility to change the systems and policies which do not fit the transformation vision, developing and promoting people who can implement the change vision and employing it to all new projects. In the last stage the focus is on grounding the changes to the organizational culture with long-term perspective. Ibid (1996, 20-23.)

Kotter (1996, 158) argues that all the eight stages are required due to the extraordinary difficulty of changing organizational culture. Schein (2001, 131) also states that unlearning the existing attitudes and shared values is often uncomfortable and creates distress which increase change resistance. To support the learning in a transformational change in the organization psychological safety can be achieved by following actions: management to follow the communicated vision and mindset, organizing enough proper training to support the new mindset, using positive role models in order to let others to see the new behavior in action, creating supporting forums in order to openly share the thoughts of the change process and creating uniform systems and structures which support the new ways of working and mindset by for instance, renewing the rewarding system to emphasize team based behavior instead of individualistic success (Schein 2001, 142.)

Major change initiatives are often made up of number of smaller initiatives. If an organization is in crisis, the first change project within a larger change process can be a turn-around effort whereas the followed initiatives might be associated with new strategy or reengineering, followed by structural and cultural change. Hence, multiple simultaneous projects with multiple process steps can result in complex, messy and dynamic end result requiring excellence in change leadership. (Kotter 1996, 24-27.)

## 2.4 Previous research on challenges in agile transformation

Despite the growing interest on agile methodologies (VersionOne 2011), the transformation, especially in large organizations (Kettunen & Laanti 2008, 189), may not be a simple task to accomplish. Managers may face several barriers when agile approaches are brought into traditional plan-driven organizations (Boehm & Turner 2005, 30). Drury et al. (2012) studied obstacles to decision making in agile software development teams by analyzing decisions made during an iteration cycle and identified key six obstacles to these decisions. *Unwillingness to commit to decisions* occurred due to teams not willing to make decisions. Instead, the decisions were left to Scrum Master or managers which then either made the decisions or stood back and allowed the team's confidence to emerge. Architectural decisions were often cited as these types of decisions. It was also noted that uncertainty affects decision making as well as complexity of a problem or internal conflicts. (Drury et al. 2012, 1245)

Furthermore, Drury et al. report *conflicting priorities* causing issues for decision making in regards of scope, content and priorities. *Unstable resource availability* during sprint due to other projects or external tasks such as support issues was also identified as an obstacle. *Lack of implementing decisions* was experienced by cases when decisions were not tracked nor followed up which could have been caused by the nature of agile supporting more informal communication. Similarly, the study noted also negative experiences in team retrospectives in case the positive and negative experiences were just expressed, but *decisions and actions for improvements in future iterations were not agreed* which caused frustration among team members (Ibid 2012, 1246).

Also Eckstein (2004, 43) notes that especially with large team settings the team size provides a special risk for decision-making: quality of the decision-making can suffer as the team size increases. She argues that unclear or postponed decisions can occur as team members may avoid responsibility due to a collective mentality that "someone else will decide". She also notes (Ibid 2004, 45) that especially software developers may find responsibility new to them since they can be accustomed to have always someone higher up in the organizational hierarchy who has the ultimate responsibility. Therefore, in case developers are given responsibility, they may feel uncertain due to their inexperience in taking responsibility and understanding what it implies. Furthermore, additional effort and time may be needed to build a culture of trust and respect among the development teams to facilitate such collaborative decision-making (Nerur et al. 2005, 76).

The practitioners have noticed (Eckstein 2004, 28) that agile can *disclose problems buried in teams* otherwise invisible. For instance, if team members refuse to work in pairs, this could be a sign of communication problems or that they do not respect each other. Leffingwell (2007, 295) argues that Scrum is just a simple framework which identifies

everything in an organization that gets in the way of optimally building software. The work required to manage and move these impediments represents the challenging part of the agile implementation, and it is different for every organization since every organization is unique (Ibid 2007, 295).

Kettunen and Laanti (2008, 189) noted that ideally all the elements of a company should be aligned for agility, but in practice this may be difficult to achieve as other parts of organizations may have different strengths. Hence, it is important to be aware of factors that can limit the organization's agility, noting also a tension between large software organizations and the agile concept of self-organizing teams, Kettunen and Laanti (2008, 189**)** conclude that reason why it is often difficult to apply agile methods to large organizations springs from *a missing linkage between company's business model and software development,* hence, suggesting that agility needs to be strategic choice. Kettunen and Laanti (2008, 189) continue that scaling agility to large software initiatives is challenging especially due to increase in number of external dependences. Thus, successful agile adoption depends more on external factors than issues in software engineering techniques.

Petersen and Wohlin (2009) researched issues and advantages in large scale agile development settings in Ericsson AB and identified several hindrances: *handover from requirements to design takes time* due to complex decision-making processes, *a prioritized backlog is difficult to create and maintain,* and *too much testing documentation is required to be produced* due to the company's processes. Furthermore, it was found that there is *management overhead* due to high number of teams requiring large amount of coordination and communication effort. Additionally, attached to the previous obstacle, it was found that *dependences were difficult to recognize in the project implementation planning* and were not taken into account in the architecture plan. Thus, *the architecture received only little focus* which led to bad design decisions and hence, to conclusion that agile methods do not scale well. (Petersen & Wohlin 2009, 5-9)

Karlstrom and Runeson (2005, 45-48) studied the feasibility of applying agile methods in large software development projects using stage-gate project management methods. They found that developers were motivated by the agile development principles, but that some of the teams *lacked management commitment* as they initially considered the new methodology with empowered developers as a threat and needed to be trained for agile. Furthermore, the study found that agile developers were focused on past and current releases, while the managers focused increasingly on current and future releases, resulting to a potential problem of *raising technical issues to management too early*.

Similar to Karlstrom and Runeson (2005), also Grenning (2001) who studied agile adoption in a company setting with a large formal software development process concludes that *lack of trust between management and developers* can be a hindering factor

for agile implementation: in case the developers would pursue agile, management buy-in is not self-evident.

Adolph et al. (2012), 1276-1277 studied number of agile software development teams in order to understand the variability of productivity in software development and how the teams manage that. They encountered several cases where the balancing of openness and communication was replaced by *leaders either leaving the team exposed or aggressively sheltering the team from the organizational ecosystem*. In some cases, the cut offs were so extreme that the team would be referred to as a black whole from which nothing ever escaped.

Adolph et al. (2012, 1276-1277) also discovered that frequent problem of software development was a perspective mismatch, in other word, *an ability to get everyone on the same page regards of how to achieve the sprint goal*. Furthermore, Adolph et al. 2012, 1283) noted that most software teams in their research consisted of members with a variety of skills and domain knowledge, and once there exists an understanding and consensus of their sprint tasks, team members reduced their interactions with others to work undisturbed. This kind of *bunkering* may lead to cognitive divergence or perspective mismatches between team members, resulting into potential surprises during sprint reviews. (Ibid 2012, 1283.)

Another common failure in agile software development model perceived by Adolph et al. (2012, 1284) was *long cycle time*. Although most of the studied teams claimed to develop software in short sprints, there were several example cases of sprint commitments not being reached, and work that was not completed was simply being rolled over into the next sprint. Additionally, working software demonstrated at the end of the sprint was in fact incomplete, hence, giving misleading impression to product owner about the true state. Thus, developers were acting as if they were operating on short cycle times and assuming the benefits of a short cycles while, in fact, they had long cycle times.

The success of agile methodologies relies on a cooperative social process characterized by communication and collaboration between the members of the community who value and trust each other. For developers accustomed to work independently or with rather homogenous groups of analysts and designers may find shared learning, retrospective workshops, pair programming and collaborative decision-making challenging concepts. Furthermore, iterative test driven development, in which the test cases are written prior to coding aiming to emphasize adaptability, maintainability and early and frequent testing, can be difficult to implement due to its contradictory basis for testing (Nerur et al. 2005, 76-77).

Customers or their representatives have an essential role in the agile development in order to receive constant feedback and keep the focus on priority items. It may be difficult to find such *persons who would actively participate in the development process*

*with high commitment*, knowledge and proximity especially in for complex systems (Nerur et al. 2005, 76)

Change from traditional process-centric, life cycle model to people-centric, feature-based development agile methods is seen to entail major alterations to work procedures, tools and techniques, communication channels, problem-solving strategies and roles of people (Nerur et al. 2005, 77). Thus, changing the practices and mindset to support the understanding that everything is uncertain has been found to be a challenging effort.

A study of agile opinions at Nokia Corporation identified three major challenging areas of agility (Laanti et al. 2010, 284). The major hindrance for agility was the *deployment of agile methods themselves to the organization*. Respondents noted problems for example with rolling the deployment out to all teams simultaneously to people who are not familiar with and to ensure that it is followed correctly. Also finding the right balance and level of agility was found to be difficult. This concludes that if one team plans going for agile, then all the teams must go agile and respect it in order for it to be effective. (Laanti et al. 2010, 284).

The results of Laanti et al. (2010, 284) indicate also that the *requirements management* i.e. managing and planning requirements in more flexible and iterative manner was perceived challenging. Based on the respondents the flexibility meant that software was now created as fast as possible without any prior planning or design effort. Product management, frequent changes, the new way of managing requirements in prioritized backlogs was also seen as a challenge. Furthermore, *losing visibility and transparency* were listed as the major problems of agility reflecting the information sharing purposes: how to ensure the big picture was lost when development is carried out for a large project/program. (Ibid 2010, 284.)

## 2.5 Synthesis for software development management

Agile methodologies address the challenge of an unpredictable world by relying on people and their creativity rather than on rigid processes and tools often associated with the traditional, planned driven software development process. Due to the unpredictability, the agile software development process can be considered as a complex adaptive system in which the outcome, novel software, emerges as a result of constant interactions among the parties in the development organization (Figure 13). In order to manage the complexity in the agile software development, constant feedback loops are applied to manage the social complexity of the development organization.

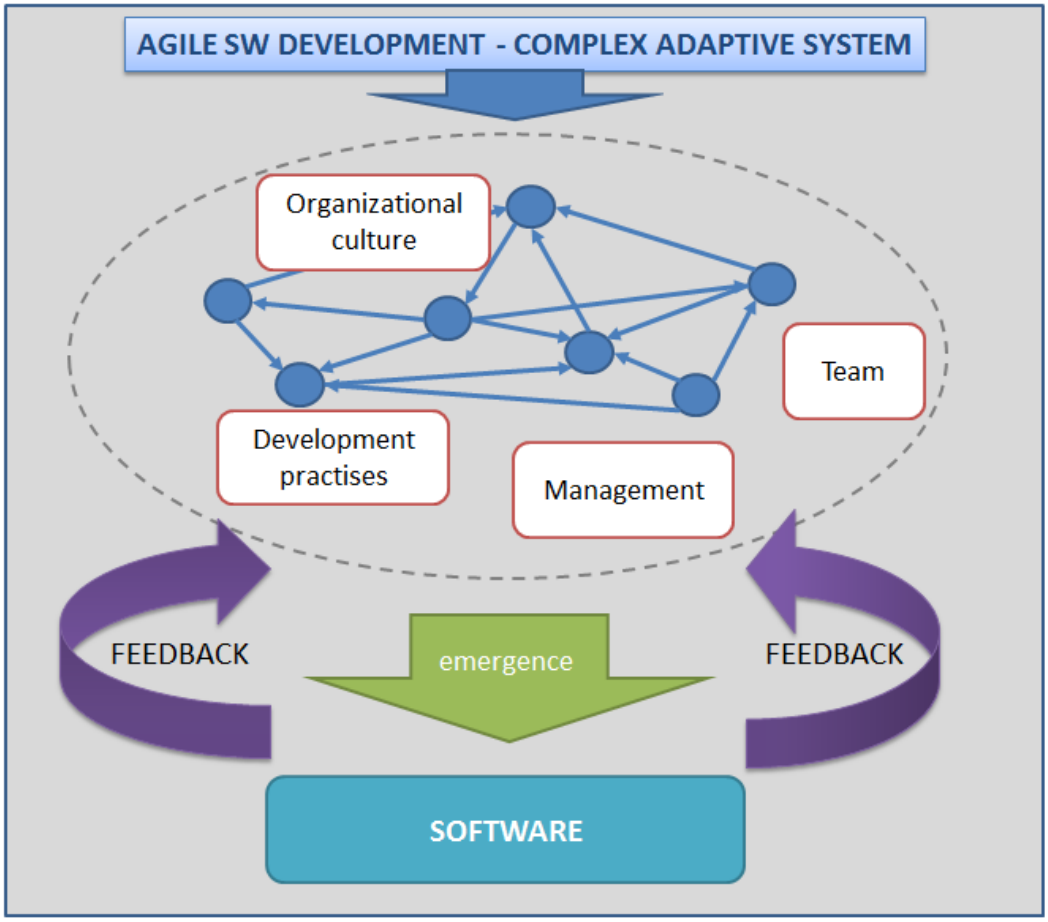


Figure 12 Framework for agile software development and aspects affecting to its success

The interactions within the development organization are, however, influenced by several different aspects which may impose challenges to an optimal functioning of the whole system. Previous literature concerning challenges in agile transformation has noted issues derived from or connected to organizational culture, management style, development team characteristics and from agreed development practices as illustrated in Figure 13. This high-level theoretical framework is used as a basis for the subsequent empirical part to identify, explore and understand the challenges in the agile transformation process in the case organization.

# 3 METHODOLOGY

## 3.1 Research approach

Intention of this research is identify and create in-depth understanding of the main issues and their management associated with the agile transformation in order to answer the research question: *"Why managing agile software development transformation is experienced challenging in the case organization"*. Theoretical chapters provided a lens to contemplate the phenomenon and associated potential issues further in the case organization which has undergone agile transformation recently.

A qualitative research approach was selected as the most suitable method for gaining understanding of the set research problem. Qualitative research is a mixture of the rational, explorative and intuitive focusing often on social process and placing emphasis on the process of understanding (Ghauri & Grønhaug 2002, 86). The reasons for choosing qualitative research approach for this study were due to the experiences of the researcher, possibility to elicit sensitive data, the topic being complex phenomena to communicate with structure based questions and the possibility for holistic outlook of the situation of interest from multiple perspectives (Malhotra & Birks 2007, 153-155). Qualitative research provides an opportunity to focus on the complexity of the phenomena on its context and the opportunity for a critical view about the social world of business and its core processes (Eriksson & Kovalainen 2008, 3). The research problem of this study focuses on uncovering experiences, behaviours and underlying issues in the case organization which are difficult to examine via quantitative research methods (Ghauri & Grønhaug 2002, 87).

In this research a case study approach was used in preparing the empirical part of the research. A case study method has been claimed to be the preferred strategy when "how" or "why" questions are being asked (Yin 2003, 1). When the concepts under study are difficult to quantify, or there are too many variables to be considered, case study research is seen particularly useful (Ghauri & Grønhaug 2002, 171). This case study is intensive by the nature of the research design aiming at understanding a unique case from the inside by providing a thick, holistic and contextualized description (Eriksson & Kovalainen 2008, 58).

From the from ontological assumption point of view this research is based upon social constructionism philosophy as the focus is on the contents of the empirical data and the full complexity of human sense making as the situations emerge (Eriksson & Kovalainen 2008, 19)

## 3.2 Selection of the case company

According to Yin (2009, 26, 32) the selected case should represent a real-life phenomenon with sufficient access to the potential data and likeliness to illuminate the set research questions. Essentially any large software development organization who has recently pursued agile transformation would have been a conceivable candidate to this research topic. The success of the agile transformation is not considered important, but relevant amount of agile experience – more than a year – was set as a requirement in order to fulfill the aim to focus on more experienced and mature agile settings. Additionally, an organizational setting utilizing scaled agile was preferred in order to capture the aspects of more complex issues in the study. The location of the case company was considered irrelevant in the case selection phase; however, from the data access point of view the selected location in Finland was seen beneficial. The selected case company operates in the field of ICT and telecommunication internationally and employs over 100 000 persons globally.

As part of its new strategy, the case company has underlined a need to its employees to adopt a challenger mindset. This challenger philosophy emphasizes results, speed and accountability which entails a request to change, adopt new attitudes and new ways of working with a specific focus on driving faster decision-making and reducing complexity. Thus, the size and scope of the selected case company creates good basis for exploring the research topic.

One of the concrete means to implement this new strategy has been agile transformation efforts in the global IT division of the case company. Before the company wide new strategy creation, the case software development organization responsible of company's factory IT software development globally and hosting approximately 100 employees had already started as an internal IT pioneer the agile transformation process, employing practices and processes to manage large scale software development initiatives. Hence, the case company, and particularly the case IT organization was considered representing well a large-scale mature agile software development setting for studying challenges they had faced in their transformation journey.

## 3.3 Data collection

A case study often involves data collection through multiple sources (Ghauri & Grønhaug 2002, 171). The primary data collection for this study was conducted by personal face-to-face in-depth interviews in the premises of the case company. The advantage of such interviews is in gaining accurate and clear picture of informant's position or behavior especially in case of complicated or sensitive issues where the inter-

viewer can ask for further elaboration of answers and attitudes (Ghauri & Grønhaug 2002, 101-102).

In addition to personal interviews, participatory observation was used as a secondary data collection tool. In ethnographical research methods, such as participatory observation, the researcher is part of the community under study with aim to learn its culture, mindset and ways of working from inside perspective (Eskola & Suoranta 1998, 106). In this study, the researcher works as a part of the case organization which has allowed longitudinal observations of the organizational set-up and team culture via daily tasks and collaboration (Koskinen et al. 2005, 81). The added value of observations was important particularly when enhancing understanding of the current agile development process and culture of the case organization, and when planning the interview guide. The main advantage of observation is a possibility to collect first-hand information in natural setting, interpret and understand the observed behavior, attitudes and situation more accurately, and to capture the dynamics of social behavior in a unique way not attainable via interviews. However, among the limitations of observation techniques are the difficulties in translating them into scientifically useful information (Ghauri & Grønhaug 2002, 90).

Selection of the informants was planned in terms of people, timing and situation (Koskinen et al. 2005, 90). Managerial roles of three different agile teams were selected as representatives of the case organization. All these teams began their agile transformation simultaneously, they are rather equal in team size and they possess equivalent amount of agile experience in their current team setup. In particular, these three teams develop, maintain and support case company's manufacturing execution system which contains tens of application modules, has thousands of users globally and millions of transactions are handled and stored per month. The studied teams are co-located in Finland and they have started their agile transition during spring 2011. The teams' agile mode of operation is based on large scaled agile framework, but the teams have adopted certain practices from other methodologies as well, such as pair programming from XP. Therefore, due to above-mentioned characteristics, these teams were identified as good candidates representing the whole case organization.

The individual roles identified behind these managerial positions were following: Product Owner/Capability Manager (in two of the teams this was a shared responsibility), Lead Developer, Release Manager and Scrum Master. In addition to these roles, a former Product Owner of one of the team was also selected in order to cover also the start-up time of the agile transformation in this team. The aforementioned roles are largely involved with the case organization's agile development process with all the related organizational levels and key stakeholders; hence, they have a holistic view from their central role concerning the agreed agile mode of operation in their organization.

From timing and situation wise in regards to agile transition, the participatory observations began when the agile transformation was initiated in the case organization. Written notes were created on weekly basis from selected observations which were considered important from the challenges management point of view. Interviews, however, were conducted only after the definition of maturity set in this study was received: the teams had more than a year of agile experience. All the interviews were carried out in 2 weeks' time, right after the summer holidays, since there was a threat of organizational restructuring which was estimated to have potential impact on team setups and individual roles.

All the invited 14 persons accepted the interview proposal. The interviews were conducted in the case company premises during August 2012 (Table 2). Semi-structured and open-ended questions giving respondents the freedom to answer with their own words (Eskola & Suoranta 1998, 87) were posed in the interviews lasting an average of one hour.

Table 2 Summary of the conducted interviews for the case study

| **No** | **Team** | **Role** | **Date of Interview** | **Company experience (years)** | **agile experience (years)** |
|---|---|---|---|---|---|
| 1 | Team 1 | Product Owner/ Capability Manager | 22.8.2012 | 13 | 1,5 |
| 2 | Team 1 | Lead Developer | 27.8.2012 | 12 | 1,5 |
| 3 | Team 1 | Release Manager | 21.8.2012 | 13 | >1 |
| 4 | Team 1 | Scrum Master | 23.8.2012 | 8 | 1,5 |
| 5 | Team 2 | Product Owner/ Capability Manager | 20.8.2012 | 24 | 2 |
| 6 | Team 2 | Lead Developer | 23.8.2012 | 6 | 1,5 |
| 7 | Team 2 | Release Manager | 22.8.2012 | 16 | 1,5 |
| 8 | Team 2 | Scrum Master | 27.8.2012 | 12 | 1,5 |
| 9 | Team 3 | Capability Manager | 28.8.2012 | 13 | 2 |
| 10 | Team 3 | Product Owner | 28.8.2012 | 18,5 | ~1 |
| 11 | Team 3 | Lead Developer | 30.8.2012 | 16 | 5 |
| 12 | Team 3 | Release Manager | 30.8.2012 | 21 | 3 |
| 13 | Team 3 | Scrum Master | 27.8.2012 | 11 | 1 |
| 14 | Team 3 | Former Product Owner | 3.9.2012 | 10 | 2 |

The language used in the interviews was mainly Finnish, only one of the interviews was held in English. First background questions concerning agile and company experience were asked, followed by the interview questions and finally, the closing questions. The linkage between research problems, theoretical equivalents and interview themes is visible the operationalization table in Appendix II. A comprehensive list of the interview questions is available in the Appendix III. The interviews did not follow the interview guideline fully: the themes of the interview questions were all covered, but the order of the questions or themes might have been different depending on the natural flow and progress of the interviews characterized by conversations. Many of the interviewees did not have any prior agile experience before the transformation started in the case organization, but most of them had long experience within the case company (Table 2). In order to maintain a complete anonymity of the interviewees, the citations in this paper indicate only a team number of the informant who has provided the particular comment.

## 3.4 Data analysis

Data analysis is a process of coding the data so that it can be broken down, conceptualized, put together and presented in an understandable form (Ghauri & Gronhaug 2002, 179) Analyzing case study evidence is particularly difficult since the strategies and techniques have not been well defined (Yin 2003, 109). After collecting the data, the audio-recorded interviews were transcribed from the tapes (Ghauri & Grønhaug 2002, 108), and then carefully re-read and studied for identifying categories from the respondents' answers. The mass of data was carefully divided into analyzable units by creating themed categories (Malhotra & Birks 2007, 240).

Thematization as an analyzing method has been found useful especially in solving a practical problem. In thematic analysis the focal themes in terms of the research problem are extracted from the data. (Eskola & Suoranta 1998, 176, 179.) An analysis based on development of a case description forming the grounds for framework for organizing the case study is based on direct interpretation of the research materials instead of formal coding procedures, and it is referred to as inductive-oriented strategy of case material analysis (Eriksson & Kovalainen 2008, 65).

Although the analysis of the case is not based on a pre-given theoretical framework, sensitizing the empirical data by creating a general sense of reference into the analysis by looked at prior research on the subject can help in describing and analyzing the central organizing features of empirical data and the meanings invested in them (Ibid 2008, 66). During the analysis, the described issues and speculated underlying reasons for them feel into four general themes which are also identified by the agile related prior

literature (e.g. Drury et al. 2012; Laanti et al. 2010; Petersen & Wohlin 2009): challenges connected to organizational culture, management aspects, team dynamics and development process.

## 3.5 Trustworthiness of the research

Compared to the qualitative research approach in which data collection and analysis occur often parallel, distinctive process phases can be identified from quantitative research approach (Mäkelä 1990, 45). In qualitative research the researcher's subjective viewpoints form the baseline for the research in contrast with the traditional view associated with quantitative research, which focus on evaluating trustworthiness by measuring reliability and validity (Eskola & Suoranta 1998, 211-212.) Thus, the traditional concepts of validity and reliability may not be applicable for evaluation of qualitative research.

Mäkelä (1990, 47-48) suggests four different factors for evaluating qualitative research: significance and sufficiency of the data, and coverage and transferability of the analysis. The collected data of this study is significant in terms of the needs in order to answer the research questions. Furthermore, triangulation by complementing interviews with participatory observations and internal document analysis was utilized to make the data more sufficient and significant. Internal documents refer to company confidential material, such as sprint retrospective notes, and process and training documents. However, there is a possibility for bias in terms of the case company relationship and the researcher due to the full time employment of the researcher within the case organization. Due to this fact, it is conceivable that the interviewees may not have revealed their genuine opinions as the theme under study – challenges in agile implementation – can be a sensitive by its nature. To mitigate this possibility, the researcher has made an effort to establish trust and comfortable settings for the interviews, and emphasize the ethicality of the research by highlighting and preserving the anonymity of the answers throughout the process.

In terms of evaluating the sufficiency of the data, the material gathered from three agile teams of the case organization with informants having all long-term employment experience within the case company and in the case organization should represent rather broadly all the relevant topics for uncovering the challenges in the agile transition. However, all the interviewees were not engaged equally in the daily agile routines or in the other scaled Scrum activities, such as release planning. Therefore, in order to receive more reliable information, the viewpoints of the agile team members would have made this study more comprehensive although Scrum Masters acted as the representatives of the teams. Moreover, although all the interviewees had strong experience of the organi-

zational culture and ways of working prior agile and during the agile transition phase, it is likely that not all the issues from the early stages of the transition started over a year ago were brought up due to the human mind which tends to concentrate on the most topical issues only. Similarly, the organizational change, which occurred during agile transition over a half a year ago before the interviews were held, may have also influenced to the answers as some of the responsibilities and team set-ups were changed. Due to this reason, in order to capture a broader picture of the early stage challenges, an additional interview with a former Product Owner was also conducted.

In general, however, the answers of the interviewees were in line with each other and the participatory observation supported the notions received through the interviewees. Consequently, the data received through the interviews gave no reason to suspect its truthfulness as no significant contractions emerged. The interview questions were originally formed in English as that is the official language of the case company. Most of the interviews were held in Finnish except one. However, the basic concepts were clear and understood the same way both by the interviewees and the interviewer because of the already existing relationship between interviewer and interviewees irrelevant of the language used.

As the amount of interview data was relatively small (~120 pages of transcribed texts), the data was analyzed in full scale. The interview questions were formulated in such a way that there exists natural overlapping (see Appendix III) between the themes as the literature review and previous participatory observations already suggested the topic under study has multiple facets. Consequently, answers sometimes emerged to questions other than those being discussed at the particular moment, or similar type of answers were received to questions under different thematic sections, confirming the notion of multifaceted and complex theme under study.

Although an agile transition process can vary since the central element of the process, the team setting and environment, are unique in each transition, and the body of scientific agile literature in terms of the challenges is still rather scarce, there is a potential issue of transferability of the results: different organizational setting with individuals with different mindset may produce different results. However, the objective of this study is to contemplate on the challenges the case organization has experienced in terms of agile transformation process and how to mitigate them, and therefore, generalizations of the results as such has not been in the focus.

# 4 CHALLENGES IN LARGE-SCALE AGILE SOFTWARE DEVELOPMENT IN THE CASE ORGANIZATION

## 4.1 Case organization introduction

### *4.1.1 Organizational structure of the case organization*

Prior to agile transformation the case organization was aligned based on matrix organization structure which was seen as a factor causing issues: the responsibilities and direction were not always clear and communication was not always on adequate level.

When the studied case teams started their agile transformation, the matrix organization of that time was left rather intact, but new roles of Product Owner, Scrum Master and portfolio management where introduced in addition to the existing lead developer and release manager roles and specialist roles. The Product Owners did not have line management responsibilities, but the group of Product Owner, lead developer and release manager with line managers where mainly responsible of the product level planning and requirements management. Due to the organizational change in turn of the year 2012, the products became capabilities, the matrix organizational structure was broken up and new functional vertical structure created was as illustrated in Figure 14. Portfolio planning team as well as the groups of capability teams with their group heads are being led by a Head of Capability Area. These roles are referred to middle management in this study. Each capability team includes normally roles of Capability Manager and Product Owner which are often performed by same person. The teams also have dedicated Process Developer, Lead Developer, Release Manager and Scrum Master roles in addition to specialist roles. Capability groups host also Project Manager roles.

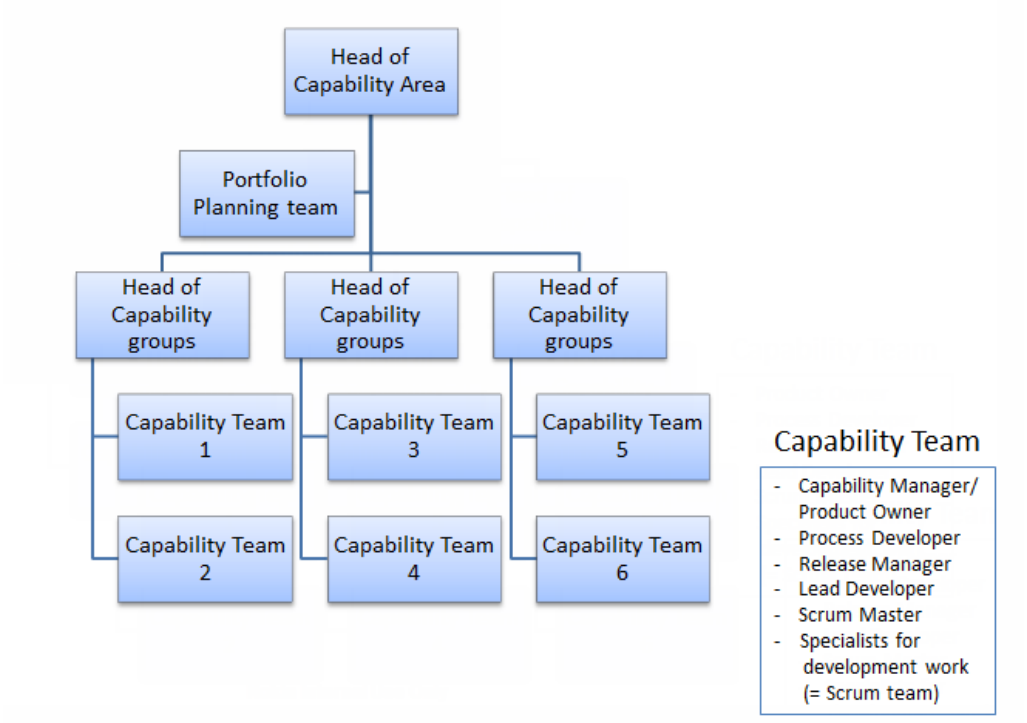


Figure 13 Case organization's structure after organizational changes

In the interest of this study are three capability teams and their selected roles, which are involved in planning, managing and leading their capabilities from agile software development perspective. The new role of process developer introduced after the organization change has its roots in business providing business process management related knowledge and input to the teams which previously possessed mainly technical knowledge. Therefore, the role of process developer was out scoped of the interest of this study and the focus is on studying the perceptions of roles which were existing from the beginning of the transformation process albeit small changes in the role names: capability managers/Product Owners, release managers, lead developers and Scrum Masters.

### *4.1.2 Scrum based scaled agile framework model replacing plan-driven processes*

The case organization utilized previously waterfall type of plan-driven software development process in managing its projects. The waterfall projects encountered number of obstacles: schedules were failing due to huge size projects which were tried to be managed by one chunk. As a consequence, failing schedules meant losing opportunities elsewhere. Prior to agile transformation the organization development process was characterized as rigid, plan-driven and micromanaged. Time was consumed for creating extensive plans before any implementation could take place. Initiatives were processed

in too large entities in order to reserve resources which were then not efficiently used. All the small enhancement items were left aside due to this. Multiple project allocations per person were common and certain resources were fully overloaded. Bureaucracy and stiffness increased during the previous years as the case company began to grow extensively which resulted in decrease in efficiency and achieved targets in the case organization.

Changing project scope after the plans were agreed was often difficult if target was to keep the planned timeline. Multiple small project allocations and multitasking of resources decreased productivity; efficiency was lost, deliverables were not in time or in scope, teams did not have capacity to response fast enough and were feeling frustrated.

Furthermore, the lack of portfolio prioritization put the pressure on line managers to make decisions on prioritization as the higher organization levels where they should had been made failed to do so. Hence, due to lack of proper requirement analysis and prioritization process, the requirements were prioritized and implemented on the basis of the voice of the stakeholders who could keep the most noise and demand most intensively the fulfillment of his requirements. Overall decisions were made as needed, but no one seemed to have the ultimate accountability of them and their business impact.

The numerous challenges involved with the projects and development ran by waterfall model motivated the organization to change the way its projects were run. Striving to develop new approaches to be competitive and more responsive, the case organization selected scaled agile framework as the new method and started agile transition project to enable the transformation in the organization.

To achieve the targets of the transformation initiative, the case organization has adopted similar agile framework model to scale its planning efforts and bring more discipline to the process to address governance and risk as proposed by Leffingwell (2009, 4) and often referred to as scaled agile framework. At the top level (Figure 15), portfolio and release management planning team makes sure that strategically important initiatives for business are planned and prioritized on the next six months portfolio roadmaps accordingly as epics, follows up the progress and makes changes to the prioritization if needed based on stakeholder management inputs. Several capability teams normally contribute for deliverables required for filling the goal of an epic within the release cycles. Further on, this level agrees the release schedules with business stakeholders.

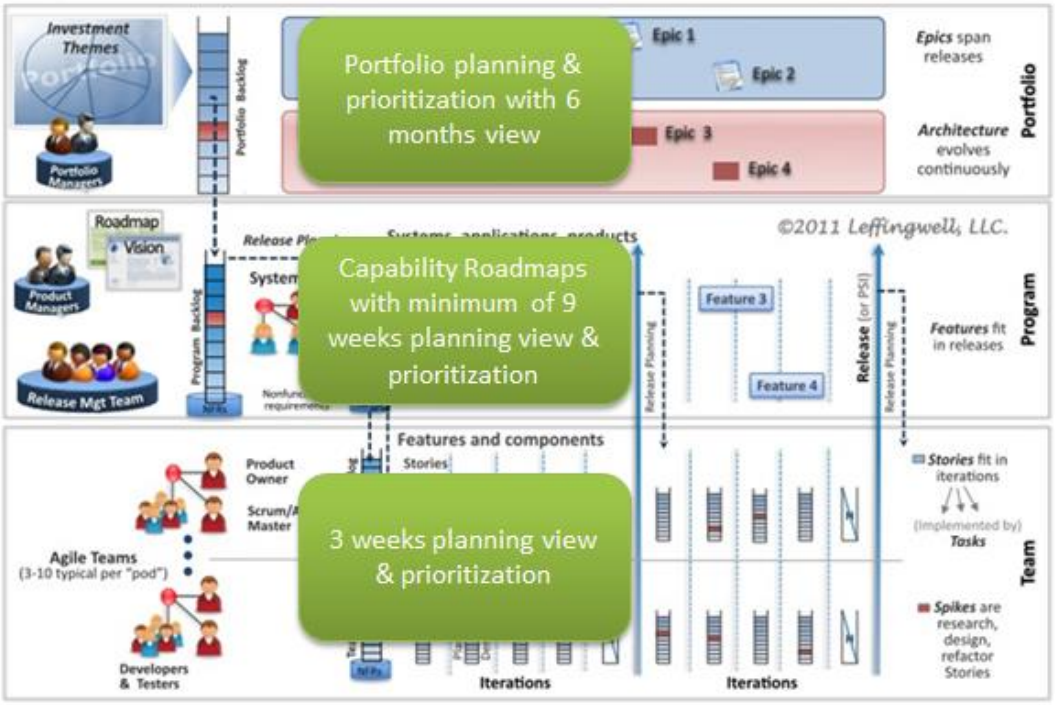


Figure 14 Case organization's scaled agile planning and release model (modified from Leffingwell 2009, 4)

At the capability management level, each capability identifies features and purifies them into user stories derived from higher lever epics. Such features and user stories form the detailed product backlog for each capability. In this level, also capability specific release roadmaps of created for minimum visibility of next release cycle lasting nine weeks. In the roadmap, the features and user stories are prioritized by following the guidance set by the portfolio management.

At the lowest level, the Scrum teams of capabilities of 5-10 team members define, implement and test user stories in a series of three weeks sprints by following the priorities set in each sprint planning meeting. The sprints fit to the release cycles. The length of the sprint is a result of constant retrospectives: originally the length of the sprint was two weeks (10 working days), but it was soon realized to be too short to be able to finalize the user stories of the sprint fully.

From the requirements management point of view there are artifacts of need, epic, feature and user story involved. A need is a basis for any changes in a system or processes. It can range from setting up a new factory to small change in specific application model. There are various stakeholders who can initiate needs.

In case the need is identified as having large impact on several capabilities and lasting several releases to complete it, it is directed to portfolio management for further purification and prioritization which will eventually create *epic* out of the need. Epics as such are not implemented, but they are broken down into features which in turn are fur-

ther broken into user stories which as smallest artifacts of requirements are used by the Scrum teams for implementation.

*Features* are characterized by value-oriented descriptions which can be implemented within a release, that is, in nine weeks timespan. User stories created out of features define in detailed level how the feature should be achieved. Every *user story* should be of such size that in can be fully finalized within one sprint. Also, each story should state who is in scope of the requirement ("As a forklift driver in Factory X distribution center.."), what is the goal ("I want to be able to…") and the reason for the requirement (.."so that I can...."). Also, in case the story requires specific actions outside of team's Definition of Done they should be listed down to the story. Definition of Done is a checklist of valuable activities that is required to produce software and must by default be always completed for all released software, such as comments in the code, unit tests passed, user guide updates. Sometimes a story also specifies the implementation specification in very detailed, but in most of the capabilities, the Scrum teams are expected to contribute to the implementation by themselves. Product Owner of each capability team has the overall responsibility to manage capability specific features and user stories as they move through the value stream. The constant progress of user stories in three weeks sprints and features in nine weeks release create the agile release train for the case organization.

### *4.1.3 Agile practices of the case organization*

The case organizations agile roles include Epic Owner, Product Owner, Transition Manager, Scrum Master, Scrum team members. Larger business requirements, needs, are handled weekly by the Portfolio level planning group where the respective Product Owner of its Scrum team represents their team. Smaller business requirements are directed and are of responsibility of each team's weekly *need management process.* This team level analyzing group is normally formed by following roles of each team: Product Owner, Transition Manager, Scrum Master, Process Developer.

Portfolio planning produces epics which are further purified into smaller detailed requirements - features and user stories - by the teams that are identified as contributors for enabling the development work required by the epic. Smaller needs by default produce only features or user stories that are completed within a release. A release spans over nine weeks which is split into three sprints. Release content is planned and prioritized in portfolio level for epic type of requirements based on business impact, and the Scrum teams plan their release content in feature and user story level based on the priorities set by the portfolio level. Release planning meetings are held before each release where the high level release plan for the whole organization is reviewed, each team pre-

sents their detailed plan for the release and capacity estimates for resourcing review, and risks are addressed if applicable. After the review all the teams commit to agreed release scope. After the release is completed, release demos are held to showcase the achievements by each Scrum team.

Once the agile transformation began, the organization gave up projects the way they were seen so far. Thereafter, the projects were known as epics, project managers were only taken in for coordination tasks where seen appropriate and all the required work was carried out by the backlogs of agile Scrum teams. Therefore, project based resource planning as such was ceased to exist; developers were not anymore allocated to multiple projects at the same. Instead, they had full commitment to the Scrum team they belonged to. Furthermore, team members did not report any longer to project manager if there was one. Only necessary meetings were to be held on a need basis, and when possible, the team members of the analyzing group participated in them in order to disturb the development team as little as possible.

The necessary mandatory meeting practices to keep the agile team iterations constantly on-going are defined by Scrum (Figure 16): 1) *grooming* meeting before sprint planning where the objective is to check the maturity of the user stories of the coming sprint with the whole team and speed up sprint planning; 2) *sprint planning* during the first day of the sprint 3) a *demo* session during the last day of the sprint 4) a *review and retrospective* session after the sprint was finished 5) *daily Scrum* meetings or short general meetings.

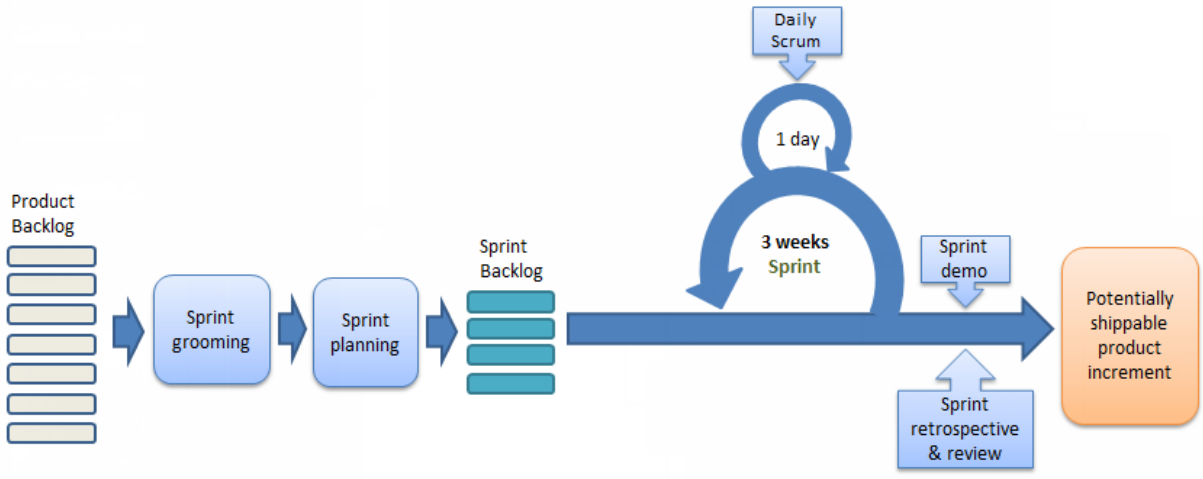


Figure 15 Case organization's Scrum process model for one sprint

In addition to these, the persons belonging to capability planning and analysis team (Product Owner, lead developer, release manager, process developer) responsible of capability directed needs management, release planning, and feature and user story purification with the team have their own short weekly meeting. Furthermore, Product

Owner is involved in portfolio management meetings to represent his or her team on a weekly basis.

The previously mentioned agile meetings are the decisions points how to continue further. After the release level planning has been concluded, the teams have sprint planning sessions. Before a sprint planning the teams estimate their capacity for the sprint by checking planned holiday periods and other reasons which reduce the default capacity target of each member. Once the sprint planning begins, the user stories are added to the Scrum teams' sprint backlog from the prioritized product backlog. Product Owner and other story owner explain the story content for the whole team if not otherwise clear. It is common to have sprint grooming sessions as well during the preceding week of the sprint planning where the target is to review with the whole team all the new stories created since last grooming session which are to be proposed as candidates for the next sprint. The purpose of the grooming session is to check the maturity of the proposed stories; are they in such a condition that they are understandable by the team and they are doable, or is further investigation still required in order to explicitly understand what the story requires in order to be fulfilled.

The team agrees the amount of stories to be committed for the sprint together mainly based on the estimated sprint capacity and previous sprint velocity, but also available competence is used as criteria. The status of the stories agreed to be moved to the sprint backlog are updated as committed. After the planning, the team starts working with the stories in priority order and creates detailed tasks required to be done in order to conclude the story. A dedicated centralized IT system is hosting all the agile requirement elements and product and sprint backlogs. As the tasks proceed, the status of the tasks is updated in the IT system. Most of the teams also have a short daily Scrum meetings with different agenda. Others follow more strictly the advised guideline of Scrum Master to ask everyone standing around a table what (s)he has achieved today, any impediments, and what (s)he will continue to work with whereas some teams prefer more informal daily short sessions over a cup of coffee.

Specific practices can be used to make the development process agile. The Table 3 shows the high-level practices used by the teams in the case organization.

Table 3 Agile practices in use by the case teams

| Agile practise | Team 1 | Team 2 | Team 3 |
|---|---|---|---|
| Pair programming | ✖ | ✖ | ✖ |
| Code reviews | ✖ | ✖ | ✖ |
| Unit tests | ✖ | ✖ | ✖ |
| Continuous Integration | ✖ | ✖ | ✖ |
| Collective Ownership | ✖ | ✖ | ✖ |
| Refactoring | ✖ | ✖ | ✖ |
| Test automation | ✖ | | ✖ |

Usage of test-driven development, continuous integration and pair programming are considered as hallmarks of agile (Schwaber 2007, 2). These practices ensure the quality of the code during the iteration in order to avoid any surprises and enable the software under development to evolve over time as requirements, priorities or technologies change. The Scrum teams of the case organization are using commonly pair programming and code reviews to ensure quality of the coding working and transfer knowledge to others to foster collective ownership, thus, all the code is available in a centralized repository with version control and all the team members are encouraged to work with applications belonging to their team. Writing unit tests, which must be passed before any code is checked in, and running continuous integration where the code is checked in daily and build is automated are also contributing to the quality management. Constant refactoring continuously improves the code as the system evolves, thus, integrity and architecture emerges continuously. To speed up testing phases, two of the teams started developing test automation for the application areas where it was seen feasible where as one of the team is still focusing on manual tests only.

When the sprint is over, a sprint demo session is hosted during the last day of the sprint. In the case organization, the three teams under this study gather together for a couple of hours' session to get feedback and present each other and invited stakeholders the results achieved during the sprint. The deliverables are showcased via live or beforehand recorded application demos, and via other presentation means in case of non-software related deliverables. After each sprint the target is to produce working software which could be deployed already after the sprint if required, but most of the software deployments are organized after the release has been finished.

As a final step to finish the sprint, *review and retrospective session* is held in order to keep continuously learning and improving the ways of working. In the case organization this session is held separately from the demo: if the *demo sessions* are on Fridays, the sprint review and retrospective is held on Monday morning. During the review session the user stories of the sprint are reviewed one by one include and their status is updated by Scrum Master; those ones finished are set to completed and those ones not finished are discussed for making the decision whether to move them to next sprint or to put them back to product backlog in case the priority has changed. During a retrospective Scrum team members are all encouraged to give there feedback of the previous sprint: what went well and where improvement would be needed. Also, the sprint burndown graph is viewed to see how well progress was made during the sprint. The comments are documented by many of the teams, and the comments for improvement are reviewed again during the next retrospective session to see whether any improvement took place, hence, enabling constant inspect and adapt efforts. After the sprint is completed, next iteration is planned right after the last one is completed – often coupled within a same meeting with the sprint review and retrospective. This practice helps to keep the team focused on the subject and minimize the amount of different separate meetings and thus, task switching.

## 4.2 Challenges in agile transformation management in the case organization

This part of the thesis will report the key challenges that emerged from the interviews related to the agile transformation process in the case organization. First, the baseline situation before agile transformation is summarized as the teams have experienced it in order to draw an understanding of the previous state. After that, the key challenges perceived by the informants are elaborated and their possible interconnections to each other also contemplated. The challenges are categorized under themes of management, organizational culture, team dynamics and development process which are also identified as focal themes of Scrum (Schwaber & Beedle 2002, 57-60). Lastly, this chapter will summarize the key findings.

Although agile practices have been claimed to be simple and easily understood (Schwaber & Beedle 2002, 57), it is not self-evident that organizations would be running full speed in agile after a year's of learning. When the organizations are not static (Laanti et al. 2010, 287) they especially may encounter major changes during a transformation period.

## 4.3 Issues in relation to leadership and management

Although gaining sponsorship from upper management for agile transformation is considered essential for the transformation success (Leffingwell 2007, 301), it may require extra effort to receive such a support. Although the case organization could learn from the agile experiences of its resource and development unit, to translate that agile understanding to IT proved to be challenging in the beginning. Training and convincing the management by the agile project team required increasing the overall understanding of agile: what does agile transformation means in practice for the case organization, how it affects to the existing controlling mechanisms and which ways of working needs to be changed in order to enable a real transformation and its benefits. For instance, to give up from existing project allocations per person and reporting of the time spent on a specific project raised heated discussions. New practices for investment item cost estimation and actual calculation were required.

The questions related to the middle and top management's level of understanding of the agile ways of working produced lengthy comments and discussion. It was considered to be enough if the higher levels of management know the basic principles of agile, as further contribution would require further knowledge of agile as well. The focal role of such management levels in agile should be in setting the strategic prioritization of different ventures and communicate this prioritization to the teams without further interference to the actual software development itself. All the respondents brought up issues encountered with middle management in relation to attitude and behavior in agile context. The management layers were considered to be too many in order to support efficient and fast decision-making from the agile perspective. Furthermore, the middle management was criticized of causing slowness in being too bureaucratic, micromanaging issues and inclined to work still according to the traditional waterfall type of mindset especially in case of hectic times:

*"I feel the middle management has not yet internalized the agile mindset, they still work the way they used to work before agile by trying to micromanage etc…. in a hurry they seem to forget agile and continue with the old habits they have worked with before." (Informant 3, Team 1)*

Furthermore, a climate of mistrust seems to prevail between middle management and the capability teams, which is causing problems in communication. The lack of trust is experienced in the team level in manner of different classes: middle management and portfolio management in the upper class and teams in the lower class where lack of listening at both ends prevails. Additionally, the teams feel they lack authority for decision-making especially when it comes to smaller development items, and they feel that

the management, including the portfolio planning group, is not able to provide clear common direction, which causes decrease in motivation at the team level. The following comment summarizes the current situation:

*"…The aspect where I'm in a sense would want to see a change in their (middle management) ways of working related to the agile principles is trust. They consider matters much broader and coarser perspective than us…. But they could utilize the agile mindset just there: empower the team or larger entity for decision-making, give them a clear goal and schedule, but let them do their work without micromanaging every single issue or change request separately." (Informant 6, Team 2)*

The participants believed that agile mode of operations has decreased the role of middle management and it may be difficult for them to find tasks they are responsible of in the new mode. As the external resources have been given up, managing the supplier resources was not needed any longer, and the focus has shifted from supplier resourcing forecasts, budgets and balancing to fixed, internal resource capacity estimates provided by the capability managers.

In the scaled agile framework model of the case organization, the portfolio planning with middle management has a key role in running the release management activities. Many of the informants described the problems specifically in relation to release management and its practices which include release pre-planning meetings, release planning meetings, middle release check point meetings and release demo meetings. The release management practices where considered not serving their purpose as being too bureaucratic and time consuming having also too many persons invited to contribute to the planning (75 invitations sent out). Also, problems related to the release planning itself where expressed. In the dynamic environment of the case organization establishing a release plan for the next nine weeks was not considered very valuable. New investment items or smaller initiatives required by the business may emerge within the release. Hence, the release content agreed upon the release planning is not fixed in reality: new, higher priority requirements are coming in and others in lower priority are removed from the release. Also, it was noted that sometimes the agreed release plans are not realistic, thus, although the theoretical capacity of the teams would suggest certain amount of work to be able to be completed, the reality is often different from the plans:

*"Management should start to create real plans, not just management plans. Do we want to make such plans that we most likely are able to implement, or plans that we wish we could execute? At the moment the plans are according to the latter. The general prevailing environment of uncertainty may have posed the pressing of panic button. At least the numbers are being exaggerated." (Informant 13, Team 3)*

The comment also notes the overall challenging business situation of the case company, which may have led to internal politicization in terms of the release content and amount of work that each team should commit to for their part of the release.

Agile implementation has changed the role of project managers. In the case organization, in effect, there are no longer projects *per se*. The work previously carried out by traditional projects having dedicated resources and budgets has changed into a mode where individual capability teams commit to epics, which have replaced the projects, and the role of a project manager is to coordinate the agreed activities and ensure communication across the teams in order to keep the planned schedule. Hence, the epics do not have dedicated resources, but the Product Owner of each team gives commitment to the required work, which is then executed via product backlogs of the teams without resource earmarking beforehand.

Changing the responsibilities of a salient role such as a project manager's has also had its implications. According to the interviewees project managers have perhaps had the greatest difficulties in understanding the change in resource allocation in agile: not a named person is not anymore allocated with certain percent of his/her monthly capacity to a specific project a month before the project kick- off. Thus, the shift in responsibilities has eliminated the previous resource management responsibilities from project managers. Additionally, the new channels and practices of communication brought by agile were not fully adopted by the project managers according to the interviewees. They were still inclined to prefer constant, direct status communication with such Scrum team members which used to be mostly involved in their projects instead of trusting the team members to carry out their work as per the commitment made during the sprint planning, or participating the daily Scrum meetings or other agreed agile communication channels. The unclarity of the responsibilities of the project managers and their learning path to the new ways of working has also been experienced by full disappearance in the areas where the contribution and involvement would be highly valued:

*"They (project managers) are nowadays more out of sight than I thought they would as the idea, nonetheless, was that project managers particularly would take part in release planning and sprint demos… I don't think the purpose is that they should completely disappear. I think they should still look after that their project or epic is progressing and sometimes even challenge a bit could something be done faster for instance" (Informant 7, Team 2)*

After the official transformation project was closed the teams have not utilized any key performance indicators or evaluation other than sprint retrospectives to follow-up

the progress in agility and improved efficiency. It was also noted that there seems to be no need for such measures: performance indicators were not seen beneficial nor suitable to creative type of work, such as software development, but also noted that no easily followed indicators were not available at the moment. Compiling the data for the indicators is considered often time-consuming which does not correlate to the possible benefits achieved by the effort.

Agile has not only changed the ways of working, but also the responsibilities of the teams. Even though multiple project allocations per person were perceived negatively prior to the agile transformation, the specialists, however, were able to utilize their skills more broadly in various development process phases: in specification, development, testing, documentation, deployments. According to the feedback received from the Scrum teams some of the interviewees noted that responsibility areas in agile are experienced too narrow. Thus, this implies that the new mode of operation may have not succeeded yet in merging the different development phases into a loop where the cross-functionalism would flourish. The previously diverse specialist work was described as a change to a factory work after agile transformation: the one and same activity only is carried out by a person when he arrives to the factory each morning at eight and leaving at four, for instance testing. It was felt important to see also in real environment the results of the work by the means of participating to support, training and deployment activities. This was seen also as a source for improvement and learning opportunities.

## 4.4 Effects of the organizational culture

After the agile introduction, the organizational culture has become more open according to the interviewees. Moreover, agile mode of operation has brought the team members closer to each other compared to old ways of working when one might have only barely known what his colleague was working with in a specific project without any further knowledge of the project itself. Others also noted that the teams had already worked in flexible and nimble ways before the official change to agile methods although without the structure that agile implementation brought to the work.

The organization where the selected teams of this research belonged to was the first from the company's IT organizations to move to agile mode of operation. Hence, the teams were in the company level IT agile pioneers suggesting that they were required to probe and establish their ways of working inside the teams as well as with the rest of the IT organization still working according to the old waterfall based methods. Therefore, it was a strategic choice for only one part of the IT organization; however, not expanding the change to whole IT organization had its implications in the beginning. The teams felt they were small isolated island among the rest of the IT organization.

When the agile transformation project part was officially completed in the turn of the year 2012, the case organization underwent a large company wide organizational change in which roles and responsibilities were affected. Portfolio planning management of that time was replaced by new organizational setting with new members outside of the case organization. The organizational change had also implications to essential roles defined by Scrum, for instance, the line managers became by default Product Owners of their teams and a new role of Process Developer was introduced to the teams.

Original agile Scrum team setups where changed after six months from the transformation start up, hence, the teams have experienced changes both in personnel, roles and application responsibilities during their journey. The change management of the reorganization process itself was difficult especially for the line managers who were required to find and reapply to their positions.

After the change, the responsibilities of different roles were seen ambiguous. The Product Ownership, for instance, is still seeking its right form as there are roles for business owner, Product Owner and process developer. Thus, it is not clear what the genuine responsibilities of the Product Owner are: what are the tasks of the role and where it should be involved. In regard to the reorganizational effort during agile learning, others also highlighted the negative impact of organizational changes to team dynamics and team spirit:

*"I don't think in agile you change that (team)…we have worked a while in agile and then comes an organizational change, and even most of the guys staying in our team are senior experienced ones, you have new team members coming and the newbies are wondering how the "old" ones and the team is working… Even if you change one person, it still means a change for the team; people will somehow transform again and try to find the values and culture within this new team…" (Informant 8, Team 2)*

The teams in scope of this study have undergone reorganizational efforts twice during their agile transformation process. As a consequence, one of the teams was split into two and the other experienced a merger with another smaller team. All the teams were and still are developing same major software system from their process responsibility point of view.

When the respondents were asked whether the changes in the team structures have generated more silo-thinking and boundaries between the teams, the responses were mixed. Others have seen it, in fact, as a positive possibility to focus and deepen the knowledge in narrower area with clear responsibility area in system wise although with the implication of more superficial knowledge of other areas. Others, however, pointed out negative experiences and the potential danger of falling into to a trap of negative silos. The teams and their work products contributing to same system solution have

clear dependences to each other which require proactive communication between the teams. However, if the communication is neglected or dependences to the development of other teams are not understood well during the implementation, the development may be incomplete or even in conflict with the work of others. These surprises have been realized and caught in the sprint demos where all the case teams demonstrate their sprint results in the last day of the sprint. The root cause for the behaviour is suggested to lie in the lack of trust between the teams, implying that the cultural change brought by agile transformation is not yet anchored into the organization:

*"We won't speak to each other enough and share the essential information and we cannot keep the schedules in synch, those are perhaps our greatest challenges at the moment. The teams do not trust each other enough. Perhaps it is some kind of echo from the past when this application was mine and this one yours." (Informant 2, Team 1)*

All the respondents had a uniform opinion that communication inside the team has increased mainly by the structure and regular practices that agile and Scrum have provided: different ceremonies from sprint grooming and planning to daily meeting and sprint demos. However, the communication across team and also across organizational boundaries has in general diminished. The farther the teams are located in organizational wise, the less communication exists and less is known of the other organization's ways of working:

*"We have had some communication troubles with the guys from organization X, as the way they are working is very much people-centric, thus, you discuss directly with their developer and when the developer is of the opinion of something, you settle for that. At least it looks like they are not yet in the agile work mode." (Informant 4, Team 1)*

The success in cross-organizational communication has a direct impact on how the dependences between the teams, for instance cross-team epics requiring development work from multiple teams, are managed. If the priorities are not aligned between the teams and clarity of the importance of the prioritized work does not exist between the teams, the problems are more likely to occur. Although most of the teams are co-located in the same site and office space, they have also some co-operation with the offshore teams developing applications for the same system. This virtual setting has created and is expected to create even further barriers for ad hoc communication and knowledge transfer efforts in the agile ways of working.

In relation to developing and understanding the agile mindset, the interviewees highlighted the importance of proper agile training of Product Owners which they considered did not take place for them. The teams received new Product Owners in the organi-

zational change during the turn of the year 2012 whereas the Scrum teams had already practiced agile for half a year at that time. It was felt that the agile mindset which the teams were already familiar with was lacking from the Product Owners. As a consequence, the teams had to train and guide their Product Owners to the new mindset which they considered not being their job:

*"He had little bit of old project type of thinking… but he quickly adapted to our new ways of working, but I would have wanted that someone else would have held the training instead of the team. I think that Product Owner should know how the team is working and what its working methods are." (Informant 4, Team 1)*

Also, the influential role of Product Owners was pointed out as in most cases Product Owners were the line managers of their teams as well. If the Product Owner has good agile understanding and is familiar with the practices supporting agile, his encouragement to the team to become more agile has more influence.

## 4.5 Challenges in team dynamics

Although the previous organizational culture existing prior agile introduction seem to have had positive contribution to the agile transformation process in the case organization as noted in the previous section, several hindrances in realizing the change vision (Kotter 1996, 21) have been encountered. The interviewees brought insight of change resistance among the teams in the early days of the agile journey, but others disclosed, particularly in Team 2, that they are still experiencing the challenges in regard to changing the mindset and work mode:

*"What we are maybe still a bit trying to struggle with is the mindset change of the senior team members. That has not been quite easy, although half of the team likes very much of the new style. That has probably been the main challenge." (Informant 7, Team 2)*

The resistance to change seems to be more peculiar to specific senior team members. This behavior was also reported to impede others work, but the manifestation of the change resistance differed from person to person. The change resistance resulted in conflicts in the beginning of the process when the ways of working were also in conflict – the early adopters (Moore 1999, 15) already employed the new ways of working whereas the skeptics had not yet embraced the new mindset and practices.

*"We have had problems, quite large ones even, in people interactions, when everyone has not yet been in the new agile work mode and have played still according to the old roles. There has been plenty of effort in trying to change the mindset, but when there is a hurry in getting the new undertakings completed, we then often cut corners somewhere. At least if we have a possibility, we should select the longer path. In the long run that will be beneficial for us. In the new projects we should look after that we will not develop any one-man's systems under any circumstances." (Informant 5, Team 2)*

*It was a big change, and it still is." (Informant 1, Team 1)*

The underlying reasons of the change resistance and related difficulties were pondered by many, however, several respondents noted that one can only speculate and guess the motives on behalf of the concerned ones. Others suggested that being too attached to the old habits might be one source of the perceived problems. Lack of knowledge and ignorance, negative attitude and resistance to any changes in general, using change resistance as a protective mechanism, and possessing different personalities were also suggested as root causes. Also, the chosen approach for the agile introduction to the case organization was mentioned as a candidate root cause due to its long length and project type of top-down implementation.

The challenging business situation of the whole case company was also brought up as a possible barrier: the fear and uncertainty of the possibility to loose one's job coupled with the pressure to perform even better in one's role as the agile brought in the visibility on everyone's work. Likewise, the overall business situation may have distracted the work and focus. Having possibility to see the new practices in action and experimenting the new ways of working by oneself was seen beneficial in order to change the mindset of the reluctant ones.

Many of the informants brought up and highlighted the importance of self-learning and shared responsibility especially in relation to software code ownership. Previously it was normal that team members were highly specialized developers of specific applications only. For instance, if one specialized team member was out of office, it was common that the project had to wait until the particular person was back in order to continue the work started by him/her.

Strong shared responsibility is expected to be manifested by stronger collective Product Ownership, facilitate competence development and knowledge sharing, and encourage learning. It should also result in more equal workload balancing where over employment of few experts is avoided. Further on, knowledge sharing and extended competences are expected to have positive influences on capabilities and enthusiasm of the team members to engage various types of work assignments. Lastly, strong shared responsibility among the team members is expected to result into a cross-functional

team in the long-term; a characteristic which is a focal for Scrum teams (Schwaber & Beedle 2002, 37). By keeping the focus on simplification and usability in any new application development in terms of software design and development the teams aim to facilitate the competence development towards shared ownership.

Others, especially team 1, felt that their team has succeeded well in removing the barriers for shared code ownership by using for instance XP originated development practices, such as pair programming or code reviews. Others, especially in the team 2, however, pondered they have not yet adopted such practices fully and felt that this was their most topical challenge for the moment which would alleviate also other problems, such as planning and commitment for a sprint and work effort estimates. The slow adoption of collaborative agile practices was considered to be due to a fear of losing one's importance especially in the beginning of the agile transformation. The thinking entails the notion that in case everyone in the team would be able to perform any task, the individual would lose its unique importance from the team point of view.

Questions related to the utilization of the decision-making authority on the team level produced mixed responses. Although the respondents felt that their teams are empowered to make the necessary decisions to turn the product backlog into sprint increment, others however felt that their team members are not fully utilizing this authority. One of the reasons brought up was the lack of experience of the responsibility in making the decisions which could explain why the team would like someone else to make the decisions for which they would now be entitled to:

*"If we are sitting together in a team meeting and you ask for opinion of others, like in the sprint review, people don't really speak out…When you ask from the team to decide, it is really difficult to get a decision. I think the whole team should realize that now it is our chance because previously we didn't have that, somebody made the decisions for us and then we did according to that." (Informant 8, Team 2)*

However, it is also conceivable that the number of ad hoc work outside of the sprint backlog and related priority changes brought in by middle management during a sprint could cause disinterest in decision-making. If management decides that new higher priority requirements outside the sprint backlog must be done immediately in a given schedule although the team states that it cannot commit them in the middle of the sprint, as an end result the team learns it does not have any real empowerment and decision-making power as there is always an authority who will decide on behalf of them.

## 4.6 Development process and practices related issues

The interviewees revealed several challenges in relation to the development iteration planning and process. Many of the items are connected with the themes that have already been discussed in previous sections, such as competence development, dependences management and management decision-making.

In all of the case teams, during the sprint planning meeting the *stories agreed to be added to the sprint are earmarked* to individual team members based on their competences. This earmarking is also utilized as a tool for balancing the work for the sprint. Preliminary planning of who could implement which item in the next sprint is already considered before the sprint planning when the smallest requirement artifacts, user stories, are ranked in prioritized order in the product backlog by the capability team. When the stories are reviewed and initial effort estimates validated with the Scrum team, the preliminary work balance plan is then adjusted if needed. However, interviewees also noted that there might be team members for which the identification of suitable stories matching their competences is not that easy due to their specialized skills.

It was noted that more experienced the team gets in planning the sprint content, the more accurate the plans become. However, especially in Team 3 the planning has caused problems and debates of the purpose of the whole sprint planning since typically the *committed sprint plan is not fixed*:

*"In our case the sprint content is always changing, we have never been able to keep the originally agreed plan, but we are always adding more stories to the sprint. However, we are less frequently pulling any stories out from the sprint, so they all stay there, which has in a sense caused that the planning is not seen that important. In best case the plan can change already in the following day of the planning meeting."(Informant 13, Team 3)*

The reason for constant sudden requirements could be an inefficient requirements management handling or requirements process communication, or the reluctance from the business to follow the agreed process coupled with rapid changes in the business landscape. At the time of the interviews, this team was considering another agile method, Kanban, in order to keep the focus better in the critical items and to relieve the problem of not having cross-functional team setup. Others also noted the high probability of sudden emerge of urgent requirements which are must to do right away even though the sprint content would have already been agreed and fixed for the coming three weeks. Therefore, certain amount of slack as a buffer per sprint is intentionally planned in the capacity if possible.

In relation to sprint planning, it was noted that *team's capacity estimate versus a sum of story point effort estimates for a sprint rarely match*. The latter indicates a jointly agreed estimate of how many days of work ie. story points it takes to complete the committed stories by following the team's Definition of Done. Hence, if one plans the sprint based on the capacity estimates without any slack for buffer, one may not reach the sprint goal, that is, completing all the stories agreed for the three weeks' sprint. The underlying reasons may be various: the stories may take longer time to complete than initially estimated, or earmarking the stories to individuals may cause overload on specific specialists as the teams are not cross-functional. Thus, in case of development effort underestimates, the *sprint goal may be missed*. If an expert of a specific application area estimates a story, he will end up different work effort estimation than more inexperienced person. Therefore, depending on who is contributing to the effort estimation planning and who is performing the actual software development work have their consequences on the success of the sprint outcome.

By following others' interests and pressure, for instance of middle management, business stakeholders or Product Owner, to select large amount of *stories exceeding the realistic capacity estimates* for a sprint has also been a problem. At the time when a team has been given a possibility to express their comments concerning the content of the sprint, the sprint has already been overloaded. The sprint goal may be missed also due to system dependences requiring development effort also from other teams. If the schedules are not aligned, dependences are not understood and communication is not taken place before the sprint starts with all the teams required for creating the solution, there is a possibility that the activities are not synchronized and a team may not be able proceeds with development before other teams have finished their development part. In general the main challenges in the sprint planning are associated with the success of the requirements analysis phase:

"*The challenge in sprint planning is how to identify enough detailed the general effect: how much the story requires work and what the real and effective work estimate is for it. I think this is partly influenced by how well the original need has been understood and written down to the story." (Informant 2, Team 1)*

Sometimes the *user stories are not added in prioritized order from the product backlog to the sprint backlog*. The problem may occur if team members have highly specialized competences and are willing to work only with stories touching application areas they are somehow familiar with. Also, if team members have rather clear roles focusing either on implementation or testing activities of the software development lifecycle, work queues may arise for testers if developers complete their work all at the same time.

Hence, several stories may be under development simultaneously instead of completing agreed amount of stories fully before proceeding to the next one.

During a sprint, *work load balancing challenges* may appear for instance at the beginning of the sprint if the sprint content is very development-oriented, but the testing specialists do not yet have any released software delivered for testing. Potential balancing problems may also be realized in case developers are releasing the developed software too late in the sprint, thus, there is not enough time to complete the testing during the same sprint in order to have a shippable product increment when the sprint ends. However, as already mentioned, the teams have gained experience and learned to prepare for such cases, for instance, by suggesting for the beginning of the sprint some non-functional work for testing resources, or educating the developers further to continuous software releasing during the sprint.

The interviews also disclosed issues involved in *a decrease of the agile meeting practices development* lately indicating that the new practices are not either valued enough or anchoring them to the team culture is still on-going (Kotter 1996, 151):

*"I think we have been a bit of backtrack what comes to agile meeting practices. We should stick to the meeting topics and not to slide any sidetracks… It feels like the participants will even go along with that and no one is stopping the train. It feels unbelievable that aren't these practices not yet coming from the backbone… Perhaps the basic principles of agile disappear time to time. Also, if the Product Owner is extremely busy, he doesn't have time to prepare for the meetings and thus, the work required for preparation is carried out during the meetings then." (Informant 7, Team 2)*

Furthermore, *lack of customer presence in the sprint demo meetings* was also reported as a constant issue prohibiting constant dialog and deteriorating the feedback loop fundamental to agile development process (Leffingwell 2010, 292). Without proper customer feedback the system development to the right direction was reported to decreased along with the motivation of the development team. Possible reasons for not acquiring the customer presence were pondered to be due to the inability to attain their focus, or just plain disinterest of the customer. In the case organization the business stakeholders or production site-specific IT teams represent mostly the customer voice.

Perceived challenges, such as inefficient Scrum meeting practices, seem to focus on the practical side of agile on a surface, but the underlying root cause could be also in a difficulty to embrace the agile and lean philosophical principles. The ceremonies may not be seen valuable, productive part of the development process where decisions are made in a transparent manner. Instead, the different meetings are treated as any other meeting the people in the organization may be accustomed to: time is wasted on sitting in meetings which do not produce any outcome.

A software development process contains often certain amount of *waste*, such as waiting time, tasking switching or extra functionalities not specifically required, do not generate any added value and should be eliminated according to the lean principles (Poppendieck & Poppendieck 2003, 4-8). Waiting time, for instance between development and testing handoffs, was noted as a potential cause of waste:

*"Some stories are rather big, so if we really do the sprint grooming, discuss and split those stories smaller and if everybody could do little bit of everything, we could reduce waiting time". (Informant 8, Team 2)*

Task switching was also brought up as a source of waste in the form of work which is executed by the teams out of the sprint backlogs. Agreed amount of the team's capacity is reserved for advanced application support work in each sprint. Also other issues, such as urgent production bug fixes and problem solving, are causing task switching within the teams. Lack of up-to-date documentation was also indicated as a root cause for wasting time as the informant of the tacit information needs to be first found who can then share it with others.

Architecture and design decisions have been found in previous studies (Petersen & Wohlin 2009, 5-9) as one of the neglected areas of agile. The interviewees also supported this by noting that architecture is not always receiving the attention in the agile ways of working which it would deserve. In principal, the roles of architects in the higher level of the organization have been responsible to address these issues, but in the level of individual teams it was seen important to focus on the conformance of the development since all the case teams develop the same information system. Lead developers and developers in general of each team have been encouraged to discuss and share their thoughts together. However, the teams have experienced cases where the clarity of the architecture lead has not been obvious, especially if the system solution has required contribution from multiple teams and communication across the teams:

*"I think agile does not address design and architecture decisions at all. In fact, I hope that lead developers together could take these issues into account, since we cannot make major architectural change decisions by a lead developer of one team only. I wish these issues would be reviewed a little bit better, since we always end up having surprises, and the Scrum team itself is in practice concentrating only on the development." (Informant 7, Team 2)*

This is in line with the findings of Petersen & Wohlin 2009, 5-9) in which architecture received only little focus and led into bad design decisions especially in case dependences to other systems were not recognized early enough.

## 4.7 Synthesis of the challenges

The interviewees believe that genuine agility is received through having a co-located team with can-do attitude in a constant dialogue and sharing information with each other in an office space which supports ad hoc interaction and communication. However, in order to achieve this target the teams have still several roadblocks even though they have practiced and learnt the new ways of thinking and operating more than a year. The Table 4 summarizes the key challenges the teams have experienced during the agile transformation process and of which many are still prevailing. The implications of these challenges are further discussed in the next chapter.

Table 4 Summary of the challenges during the agile transformation process

| Theme category | Issue description |
|---|---|
| ORGANIZATIONAL STRUCTURE | Organizational changes in the middle of transition causes role unclarity, team dynamics rebalancing, additional learning efforts for those new to agile. |
| | Increasing silo-thinking between teams working on with same system can cause misconceptions and communication barriers |
| | Lack of trust between teams |
| | Cross-organizational communication has diminished, conflicts in ways of working between different organizations |
| MANAGEMENT | Too many management layers causing slowness in decision-making and communication |
| | Lack of agile internalization of agile mindset in middle management |
| | Climate of mistrust between middle management and agile teams |
| | Release management practices not optimal |
| | Unclarity of the role of project manager in agile |
| | Responsibility areas of specialists have become narrower |
| | Functional issues in selected agile requirements management IT tool |

| | |
|---|---|
| **DEVELOPMENT PROCESS** | Earmarking development items to specific specialists in advance reduces the flexibility |
| | Difficulties in running Scrum meeting practices efficiently |
| | Demo sessions lack attendance of customer presence |
| | Sprint planning is challenging due to constanct changes within a sprint |
| | Estimating work effort for individual stories is difficult |
| | Agreeing sprint content vs. Available capacity difficult due to conflicts of interests |
| | Balancing the work during a sprint is difficult due to specialized skills |
| | Waste in the form of waiting time and task switching during a sprint |
| | Architecture and design not receiving the focus they would require |
| **TEAM DYNAMICS** | Agile mindset change effort difficult, resistance to change |
| | Conflicts in social interactions |
| | Shared responsibility in code ownership difficult to achieve |
| | Developing competences towards the target of cross-functional team is difficult |
| | Empowerment not fully utilized by the teams |

Some of the perceived challenges have similarities with the previous research findings (see Chapter 2.4), but some of them, especially the organizational issues, are specific to the case company, its corporate culture and related on-going organizational activities. Thus, the issues may arise from different organizational level and from the interactions within and between them: a team, a management, an organization (Table 4). Also applied practices in the agile development process may be the source for challenges, even though many of the issues categorized under this theme may have their root cause in the lack of internalizing the agile values and change the way of thinking accordingly.

# 5 CONCLUSIONS

## 5.1 Theoretical contribution

The purpose of this research was to bring insight to the challenges perceived during the agile transformation process within the case organization and create understanding on the underlying factors of the challenges in order to exploit the benefits of agile even in a greater degree. To achieve this target following research question and sub-question were formulated:

*"Why managing agile software development transformation is experienced challenging in the case organization?"*

Sub-questions:

1. *What characterizes agile software development process compared to plan-driven process?*
2. *What kind of challenges the case organization has faced during the transformation to agile software development process?*
3. *How to mitigate the challenges?*

Agile methodologies aim to address the weaknesses of the traditional, plan-driven development management methods by relying on people and their creativity rather than on rigid processes. The linear, sequenced waterfall type of software development process realizing business value only at after the final phase is changed to time boxed, smaller development cycles in which the vital objective is to release working software. Hence, in agile the team concentrates only on requirements needed first hand, focus on delivering them fast, collects feedback for the outcome and adapts based on received information. Thus, agile methods aim at creating reliable software more quickly and achieving this while eliminating unnecessary waste and unproductive overhead. The underlying reasons for unsuitability of the plan-driven methods lie in the surrounding reality which is requires fast adaptation to constantly changing needs.

Scrum, management and empirical process control framework and one of the most commonly employed agile methodologies, aims to bring in purposeful and productive rhythm for development cycles and enable self-organizing by allowing freedom within the agreed frame.

Achieving agility is a multifaceted phenomenon where following aspects are important for its success especially in a scaled agile model: organizational, management,

development process, people perspective. However, these are the areas where most of the issues identified in the case organization were elicited. The agile practices may sound very simple, but implementing such a new type of mode of operation based on philosophical principles novel to software development seems to be challenging. The practices bring the visibility in issues that are preventing building software optimally of which many are embedded in the organizational culture. Managing such issues seems to contribute positively to the speed of transformation and overall success of anchoring the mindset change to an organization.

Complex adaptive systems theory provides one perspective to approach this issue. Approaching software development as an uncertain, adaptive process and viewing the team as an adaptive system in which the members, agents, are interacting within a boundary with a capacity to change and learn based on the feedback (Figure 17) to avoid chaos may help in understanding the role and importance of organizational, managerial and team related interactions. Based on this concept, all these aspects have a great importance, especially in scaled agile framework based implementation, in which all the parts of an organization have a significant role in contribution to the success of agility. Viewing software development as a complex adaptive system could facilitate the understanding of the underpinning philosophy: interactions are more important than processes and solving a complex problem, such a novel software development, requires constant feedback and adaptation to changing requirements. This is the basic tenet of agile. Furthermore, according to this view causal links are impossible: what works for one team may not work for another due to the different agents involved. However, the interactions are also a source of challenges: the key ingredient of agile methods - interactions which are valued over processes and tools – can bring up issues rooted from the difficulty to change the mindset to become agile (Figure 17).

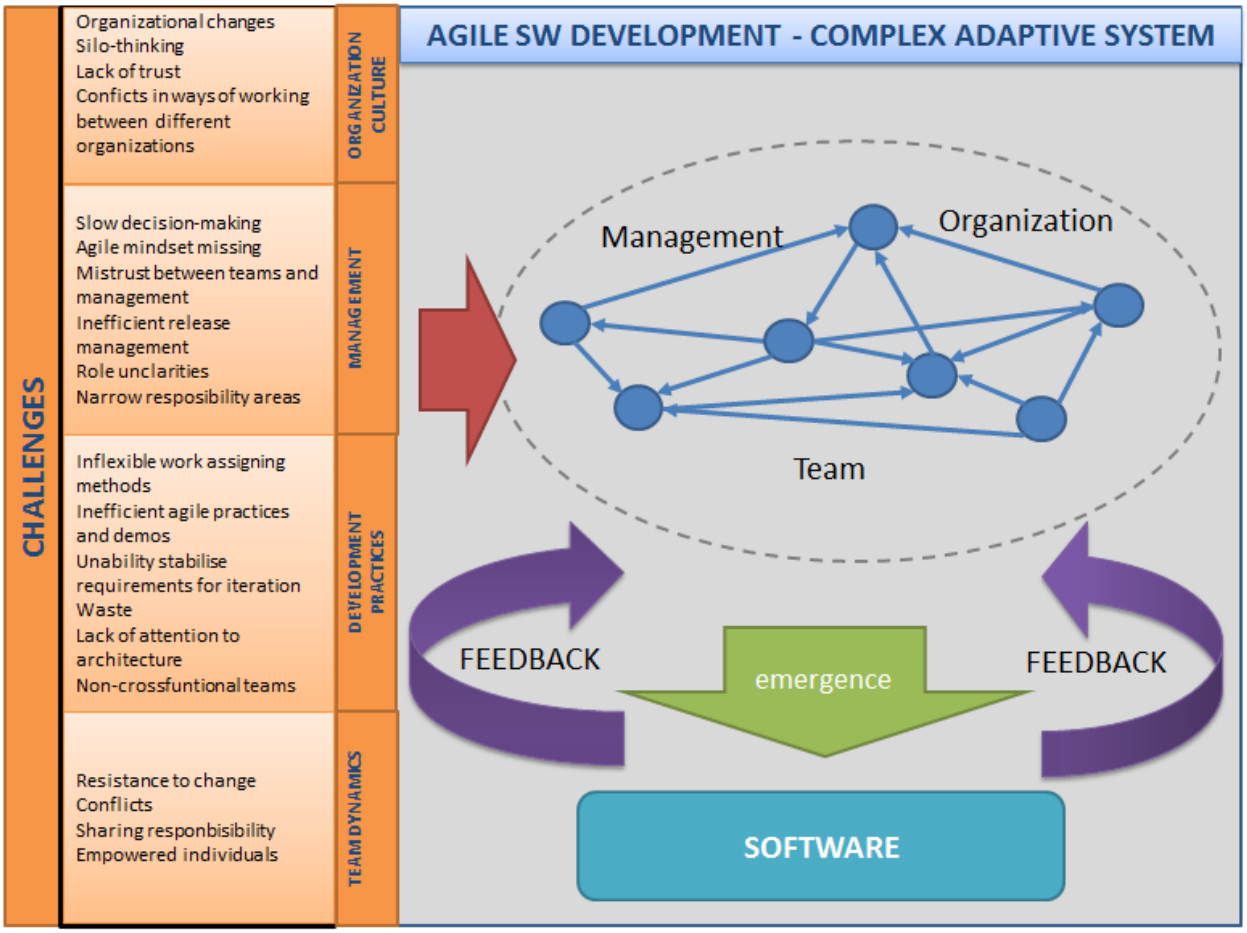


Figure 16 Complex adaptive agile software development system interactions bringing up challenges

Although this research did not focus on comparing the challenges between the different teams the informants belonged to, the experiences brought up during the interviews did reveal that all the identified challenges were not present in a same form or intensity in all of the teams. This implies that every agile implementation is unique in nature and the agents engaged in the interaction are the pivotal part of the success of achieving agility. For example, in one of the teams it was peculiar that senior developers showed more resistance towards new ways of working resulting conflicts among other team members. Whether the reasons for change resistant lie for instance in the lack of knowledge, trust to a shared objective, lack of sense of urgency and leaning towards complacency, or general reluctance to change the ways of working, it became clear that agile transformation is, in fact, rather radical change which requires active change leadership: establishing a real sense of urgency to override the complacency, communicating the vision and making sure short-term wins in performance are created in order to continue the change vision implementation. Subsequently, building momentum to trans-

form reluctant resisters to active participants before anchoring of the changes to be part of the corporate or organizational culture becomes possible.

In case the agility is not a strategic choice for the whole organization, it seems additional issues may arise due to the different ways of working in different parts of organization. Further on, the agile transformation *per se* is a massive change. Major change initiatives have often multiple sub-initiatives which can be executed parallel requiring strong leadership to complement the complexity of such efforts. In case additional organizational changes due to strategic alignment are executed, the impacts to agility should be acknowledged and considered based on the understanding that human interactions are at the very core and the enablers of it. It should be acknowledged in all levels - organizational, management and team level - that in case of large scale organizational changes are carried out, they will impact and even disrupt the development cycles depending on the scale of the change.

## 5.2 Suggestions for the case organization

When an organization is having parallel two major change programs - one focusing on strategic re-alignment of resources and one focusing on agile transformation where practices and ways of thinking are affected – there is a great risk that such colliding initiatives will slow down the progress of becoming agile. This was already expressed in the findings via the interviews: changing the Scrum team members and Product Owners affected the dynamics of the work groups and additional changes on top of the agile transformation initiative were seen negatively affecting the agile adoption causing change resistance. *The impacts of organizational changes during agile transformation should be carefully evaluated and acknowledged in advance*. Furthermore, *communicating a sensible vision with strong leadership as a driving force* would benefit in tackling possible corporate inertia and motivate the organization to continue its change efforts. Building momentum and establishing the sense of urgency again after reorganizations within agile transformation can be very challenging and regression may follow. Based on the findings, it seems that the behavioural change towards agile ways of working has taken place whereas the cognitive change effecting the belief system and social norms has not yet been fully internalized and rooted in the organizational culture. It seems the case organization would benefit *from stronger, supportive and empowering leadership to embrace and strengthen the adaptation to agile*.

The role of middle management in reinforcing the agile vision and helping the new mindset to grow deep roots into the core culture was not apparent in the case organization. The responses highlighted the lack of agile internalization experienced as micromanagement and inclination towards planned-driven approaches. The answers imply

there is a great room for improvement in enhancing the common understanding and trust in both levels: *middle management to show interest in understanding and supporting the agile* ways of working without interfering too much on the sprint level, *setting up the business priorities and clear direction, and showcasing leadership by empowering the teams to make the decisions* they are entitled to. The teams, however, could contribute in *building trust by trying to foster open dialogue towards the management and via examples express their concerns* as soon as they appear without waiting for official ceremonies, such as release demos.

According to the findings, the practical process agreements are bypassed by middle management by cutting corners in the requirements management process when urgent requirements suddenly appear. The teams within the *organization should consider is the time boxed iteration, such as sprint, the best way in future to optimize the development process*, or should other methods, such as Kanban, considered. If the time boxed development iteration is seen beneficial, then *the agreed ground rules should be respected and alternative creative methods used*, for instance by planning some free capacity for the sprint for urgent requirements, and if none appear, new items from the product backlog could be added to the sprint if time allows.

Understanding the impacts of agile mindset in the level of middle management, it can be expected that *the release management related practices will be enhanced* of which all the respondents had unified opinion and commented of not serving their purpose in their current form. The challenge of the case organization is the global form of the release planning and demos – the meetings are forced to be online as the participants are from dispersed locations. Furthermore, the organization is rather large with multiple key stakeholders who are all invited to the meeting. If one cannot make changes and reduce the number of participants, at least it should be carefully considered are there any overlapping meetings (for instance sprint vs. release demos) and how to *improve the practices to achieve more benefits and increase spontaneous mutual communication*. One concrete suggestion would be to gather all Finnish meeting participants to on a site which hosts most of the participants and dedicate at least one of the release management meetings, demo or planning, a face-to-face event with small snack to increase interaction and team spirit. If there are distant participants, they could join over the voice and video. Such sessions would *increase the celebration of a major achievement*, or put clear focus on the next release in case the event would implemented as release planning meeting. The sessions, should, however require prioritization: *all the key participants should prioritize and understand the value of the session*, which could trigger a signal for capability teams that their work is appreciated and valued.

Based on the findings, many of the roles in the organization seem to be still unclear in terms of responsibility and ways of working in agile setting. This seemed to be the case especially for project managers and middle management positions as perceived

from the team management level. Additionally, as Product Owners are often the managers of their teams as well, they have a key role in cultivating the agile mindset in their team which is risked by poor agile understanding. In a key role from agile perspective it is essential to have *proper agile training focusing on the important role specific aspects* prior to the new role. As a consequence, the head of the teams can better act as agile ambassadors towards middle management, and employ and foster the agile mindset and practices in all the communication with middle management.

One of the characteristics of agile is that all the work becomes transparent: requirements, development items and achievements. Instead of individual performance, the success is based on collective achievements of the whole team. Such visibility may create fear and uncertainty, especially in case the corporate culture has supported heroism and rewarding individuals rather than team behaviour. The heroes prior agile, often the ones fixing a release in a short notice or possessing expert knowledge of business centric application areas of which others are not that familiar, may be the ones actively resisting the change. The fear of losing its importance, the hero status, may cause issues in knowledge transfer inside the team for competence development when agile principles shift the focus on a team success. *Active leadership, open communication of common targets and understanding the motives of the heroistic ones* could help in setting up creative solutions for recognition and reward in the agile framework. On the other hand, the team members with passive behaviour may find the agile transparency threatening as their work becomes visible to others via review and demo sessions.

Due to historical reasons and legacy, the teams' application portfolios include several pieces of software which have been developed previously by a single developer. Thus, in order to gain the benefits of agile, *a roadmap for competence development inside the Scrum teams should be created* in order to avoid any 'one-man's software' in the future. This would be result in added flexibility inside the team by helping the balancing of the work inside the spring and reduce the need for earmarking development items in the sprint planning session in advance for specialists with specific skills. Also, expected long-term result would be stronger shared ownership of the whole application portfolio and higher job satisfaction. Such an approach would require *commitment from management* as competence development requires time even though performed via real development items, and *motivation from the team members to teach others and learn new out of own's comfort zone*.

Practical matters concerning sprint planning sessions may require added attention in order to make the meetings efficient and motivational for all. The nature of sprint planning meeting, for instance, requires disciplined and committed Product Owner in the case organization to make sure the stories are in the product backlog are available, purified enough and ranked in the priority order. Hence, *Product Owners should prepare for the meetings in advance*, which could act as a motivational signal for the team that the

work they are performing is important and valued Also, examples of short-term wins and repetition of the vision as well as genuine empowerment for the team would be beneficial in order the team to feel valued and appreciated.

The new requirements management process with new responsibilities has brought in structure and end-to-end visibility and accountability, but at the same time it has reduced the responsibility of the Scrum teams to fully decide and manage the requirements applicable to their application portfolio. Either the portfolio planning or analyzing roles of the capability teams have the responsibility on working on with the requirements. Prior to agile implementation in the case organization responsibilities of a specialist role may have stretched from requirements management and specification to trainings and deployments. However, according to the findings a concern has risen of narrower responsibilities and lack of diverse work assignments, resulting in lack of motivation and creativity. Thus, there seems to be an improvement opportunity in *maintaining the level of motivation due to the changes in responsibilities*.

The case teams have often cross-organizational solutions under development which requires additional focus on communication of the requirements and design decisions across organizational boundaries. The agile iteration and basic practicalities tend to keep the focus on a team level, resulting in diminished communication across team/organizational levels if no added effort is applied. Visibility on the dependences and communication over the organizations in a setting where projects with dedicated resources do not exist any longer requires clear role clarity and responsibilities. The *role of project manager could be clarified further,* which could play an essential role also in agile mode to ensure overall coordination of the project activities especially in complex cross-organizational initiatives.

According to the findings, architecture and design is a neglected area in the case organization which would benefit of additional focus and role clarity. The architects up in the organizational hierarchy are setting up the coarse directions for the case organization, but they are too far from processing the detailed issues on a team level. The lead of architecture is amplified in solutions involving multiple teams and multiple system contribution. Therefore, the *role of lead developers in terms of architecture ownership and decision-making could be clarified and sharpen*.

The findings propose that Scrum can brought in discipline and structure to the development process, and there is an overall feeling that that more right things have been done at the right time with the help of agile practices. Also, it was felt that performance has improved, but no clear indicators or assessment other than sprint or release retrospectives neither are in place nor conducted after the official transformation project was completed. Thus, the possible performance improvement is not fact-based, however, it was noted that such *easily supported agile measurements* do not currently exist. This could be a potential improvement possibility for the future.

## 5.3 Limitations and suggestions for further research

The focus of this work has been a single organization setting, therefore, the uniqueness and special access to key informants applies to this study. The rationale for choosing one company and single organization with representatives from three teams was the unique opportunity for longitudinal participatory observation which helped in forming the interview questions and building trust among the interviewees in order for them to feel comfortable on revealing sensitive information on the challenges.

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. The data collection and analysis based on the semi-structured interviews and participatory observation represent a possible limitation. The consequence of this limitation is that the findings are under the influence of author's interpretation of the phenomena observed and investigated. The use of multiple data sources, such written agile documents, talking freely with team members and managers and observing, made it possible to study the challenges from different viewpoints, thus, reducing the limitation and author's bias as working within the organization. There is a risk that the findings could also be explained by factors that evaded author's intention or were not revealed during the interviews due to the sensitiveness of the matter under study. However, the summary of the interpretation was reviewed by a case team manager, which helped in validating the conclusions.

This research focused on challenges in a mature, international, large-size organizational environment characterized by providing software development services to its own business units. However, for future research is would be interesting to learn how the challenges differ in an organization which has outsourced to offshore its software development service. Outsourcing and offshoring the software development may create far more demanding environment for communication and team building, and contractual issues add complexity to the cooperation.

# 6 SUMMARY

Transformational large scale change efforts, such as agile framework implementation, are aiming at increasing the nimbleness of the organization to manage constantly changing business needs and the complexity inherit in implementing such needs via software development initiatives. Compared to traditional, plan-driven software development approaches agile methods, such as Scrum, emphasize speed and flexibility, and recognize creative, self-organized individuals and their interactions as critical success factor of agility.

The purpose of this research was to amplify understanding of the challenges and their underlying factors of large-scale agile software development transformation process in the case organization. Furthermore, the target was to present suggestions how to address the faced challenges. A qualitative research approach was selected as the most suitable method for gaining understanding of the set research problem. A case study approach was used in preparing the empirical part of the research. The primary data collection for this study was conducted by personal face-to-face in-depth interviews with informants of the case company.

Complex adaptive systems theory provides theoretical framework in understanding why agile works by treating software development as complex endeavor where the solution emerges via interactions of the agents in the system with the help of constant feedback. It also helps in understanding the possible challenges which often derive from the interactions of different agents in the system: team, management, organization. It was learned that challenges faced by the case organization ranged from organizational issues to management and leadership, as well as from development process related issues to team dynamics of which many are rooted in the corporate culture and previous ways of working. Thus, agility seems to be a multifaceted issue. Furthermore, the basic agile tenets and practices may sound simple on the surface; however, they seem to expose the bottlenecks and problems embedded in the organization which eventually prevent the optimal organization of software development activity.

Change and unlearning of old habits was learned to be difficult. It seems to be even more difficult to achieve agility in case the organization is simultaneously experiencing companywide re-alignment activities. Instead of change management, strong change leadership with ability to understand and live the agile values, communicate a clear vision and celebrate short-term wins is important with ability to empower and motivate the agile teams even in turbulent times. Anchoring the change into an organization seems a long journey, and constant improvement mindset and mitigation strategies for arising issues - such as change resistance, practices enhancement and competence development – are beneficial in order to realize the benefits of agile and achieve the set business missions. Furthermore, it was learned that there are no silver bullet solutions or

best practices which would fit to all teams as showcased by the different challenges and situations among the case teams. Therefore, it seems each organization should find its own set of suitable practices that are based on the agile values for optimal solution.

## REFERENCES


Abrahamsson, Pekka – Salo, Outi – Ronkainen, Jussi – Warsta, Juhani (2002) agile software development methods. *Review and analysis. VTT publications 478.*

Adolph, Steve – Kruchten, Philippe – Hall, Wendy (2012) Reconciling perspectives: A grounded theory of how people manage the process of software development. *The Journal of Systems and Software*, Vol.85 No. 6, 1269-1286.

Manifesto for agile Software Development (2010) < http://agilemanifesto.org/>, retrieved 20.11.2010.

Alleman, Glen B. (2002) agile project management methods for IT projects <http://www.niwotridge.com/PDFs/PM%20Chapter%20(short%20no%20email)%20Update%202.pdf>, retrieved 23.11.2010.

Appelo, Jurgen (2011) *Management 3.0. Leading agile developers, developing agile leaders.* Addison-Wesley Professional, US.

Barlow, Jordan B. – Giboney, Justin Scott – Keith, Mark Jeffrey – Wilson, David W. – Schuetzler, Ryan M. – Lowry, Paul Benjamin – Vance, Anthony (2011) Overview and guidance on agile development in large organizations. *Communication of the Association for Information Systems*, Vol. 29, No. 2, 25-44.

Barton, Brent (2009) All-out organizational Scrum as an innovation value chain. In: *Proceedings of the 42nd Hawaii International Conference on System Sciences*, ed. by Ralph H. Sprague Jr. IEEE, California.

Beck, Kent (1999) Embracing change with extreme programming. *Computer*, Vol.32 , No.10, 70-77.

Boehm, Barry – Turner, Richard (2005) Management challenges in implementing agile processes in traditional development organizations. *IEEE Software*, Vol. 22, No. 5, 30-39.

Brown, Andrew (1998) *Organizational Culture*. Prentice Hall, USA.

Davies, Andrew – Hobday, Michael (2005) *The business of projects: managing innovation in complex products and systems.* Cambridge University Press, United Kingdom.

DeMarco, Tom – Boehm, Barry (2002) The agile methods fray. *Computer*, Vol. 35, No. 6, 90-92.

Dingsøyr, Torgeir – Nerur, Sridhar – Balijepally, VenuGopal – Moe, Nils Brede (2012) A decade of agile methodologies: towards explaining agile software development. *The Journal of Systems and Software*, Vol 85, No. 6, 1213-1221.

Dybå, Tore – Dingsøyr, Torgeir (2008) Empirical studies of agile software development: A systematic review. *Information and Software Technology*, Vol. 50, No 9/10, 833-859.

Eckstein, Jutta (2004) *Agile software development in the large: diving into the deep*. Dorset House Publishing, New York.

Eriksson, Päivi – Kovalainen, Anne (2008) Qualitative methods in business research. Sage Publications Ltd, London.

Eskola, Jari – Suoranta, Juha (1998) *Johdatus laadulliseen tutkimukseen*. Gummerus, Jyväskylä.

Fernandez, Daniel J. – Fernandez, John D. (2009) agile project management: agilism versus traditional approaches. *The Journal of Computer Information Systems*, Vol. 49, No. 2, 10-17.

Ghauri, Pervez – Grønhaug, Kjell (2002) *Research methods in business studies*. Prentice Hall, Great Britain.

Grenning, James (2001) Launching extreme programming at process-intensive company. *IEEE Software*, Vol. 18, No. 6, 19-26.

Gould, Peter (1997) What is agility? *IEE Manufacturing Engineering*, Vol.76, No.1, 28-31.

Hass, Kathleen (2007) *The blending of traditional and agile project management* <http://www.pmforum.org/library/tips/2007/PDFs/Hass-5-07.pdf>, retrieved 23.11.2010.

Highsmith, James (Jim) A. (2000) *Adaptive software development: a collaborative approach to managing complex systems*. Dorset House Publishing, New York.

Highsmith, Jim – Cockburn, Alistair (2001) agile software development: the business of innovation. *Computer,* Vol. 34, No. 9, 120.127.

Hirsjärvi, Sirkka – Remes, Pirkko – Sajavaara, Paula (2001) *Tutki ja kirjoita*. Tammi, Helsinki.

Holland, John H. (1998) *Emergence: from chaos to order*. Oxford University Press, Great Britain.

Jaffee, David (2001) *Organization theory: tension and change*. McGraw-Hill Book Co, Singapore.

Karlstrom, Daniel – Runeson, Per (2005) Combining agile methods with stage-gate project management. *IEEE Software*, Vol. 22, No. 3, 43-49.

Katayama, Hiroshi – Bennett, David (1999) Agility, adaptability and leanness: a comparison of concepts and a study of practice. *International Journal of Production Economics*, Vol. 60/61, No. 3, 43-51.

Kettunen, Petri – Laanti, Maarit (2008) Combining agile software projects and large-scale organizational agility. *Software Process Improvement and Practice*, Vol.13, No. 12, 183-193.

Khalifa, Mohamed – Verner, June M. (2000) Drivers for software development method usage. *IEEE Transactions on Engineering Management*, Vol. 47, No. 3, 360-369.

Kniberg, Henrik – Skarin, Mattias (2010) *Kanban and Scrum: making the most of both*. <http://www.infoq.com/resource/minibooks/kanban-Scrum-minibook/en/pdf/KanbanAndScrumInfoQVersionFINAL.pdf>, retrieved 15.6.2013.

Koskinen, Ilpo – Alasuutari, Pertti – Peltonen, Tuomo (2005) *Laadulliset menetelmät kauppatieteissä*. Gummerus: Jyväskylä

Kurtz, C. F. – Snowden, David (2003) The new dynamics of strategy: sense-making in a complex and complicated world. *IBM Systems Journal*, Vol. 42, No. 3, 462- 483.

Laanti, Maarit – Salo, Outi – Abrahamsson, Pekka (2011) agile methods rapidly replacing traditional methods at Nokia: A survey of opinions on agile transformation. *Information and Software Technology*. Vol. 53, No. 3, 276-290.

Laanti, Maarit (2012) *agile methods in large-scale software development organizations: applicability and model for adoption*. Dissertation, Oulu University. <http://herkules.oulu.fi/isbn9789526200347/isbn9789526200347.pdf>, retrieved 20.1.2014.

Larman, Craig – Basili, Victor R (2003) Iterative and incremental development: a brief history. *Computer*, Vol. 36, No. 6, 47-56.

Leffingwell, Dean (2007) *Scaling software agility. Best practices for large enterprises*. Addison-Wesley, Boston.

Leffingwell, Dean (2009) *The big picture of enterprise agility*. Whitepaper. <http://scalingsoftwareagility.files.wordpress.com/2007/03/the-big-picture-of-enteprise-agilitywhitepaper.pdf>, retrieved 10.1.2014.

Malhotra, Naresh K. – Birks, David F. (2007) *Marketing research: an applied process*. Pearson Education, England.

Meghann, Drury – Conboy, Kieran – Power, Ken (2012) Obstacles to decision making in agile software development teams. *The Journal of Systems and Software*, Vol. 85, No. 6, 1239-1254.

Meso, Peter – Jain, Radhika (2006) agile software development: adaptive systems principles and best practices. *Information Systems Management*, Vol. 3, No. 3, 19-30.

Moore, Geoffrey A. (1999) *Inside the tornado*. HarperCollins Publishers, New York.

Mäkelä, Klaus (1990) *Kvalitatiivisen aineiston analyysi ja tulkinta*. Gaudeamus, Helsinki.

Nerur, Sridhar – Radhakanta, Mahapatra – Mangalaraj, George (2005) Challenges of migrating to agile methodologies. *Communications of the ACM*, Vol. 48, No. 5, 73-78.

Ohno, Taiichi (1998) *Toyota production system: beyond large-scale production*. Productivity Press, New York.

Ohno, Taiichi (2007) *Taiichi Ohno's Workplace Management*. Gemba Press: Washington.

Pelrine, Joseph (2011) On understanding software agility: a social complexity point of view. *Emergence: Complexity & Organization*, Vol. 13, No. 1-2, 26-37.

Petersen, Kai – Wohlin, Claes (2009) A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case. *Journal of Systems and Software*, Vol. 82, No. 9, 1479-1490.

Pich, Michael T. – Loch, Christoph H. – De Meyer, Arnoud (2002) On uncertainty, ambiguity, and complexity in project management. *Management Science*, Vol. 48, No. 8, 1008-1023.

Poppendieck, Mary – Poppendieck, Tom (2003) *Lean software development: an agile toolkit*. Addison-Wesley, USA.

Rzevski, George (2004) *Designing complex engineering systems*. Volga Key Note Paper. Conference on Complex Adaptive Systems, Samara, Russia, 2004. <http://www.rzevski.net/04%20Designing%20Complex%20Engineering%20Systems.pdf>, retrieved 24.8.2012.

Rzevski, George (2009) *Using tools of complexity science to diagnose current financial crises*. Keynote Paper, Conference on Complex Systems: Control and Modelling Problems, Russian Academy of Sciences, Samara, June 2009. <www.rzevski.net/09%20Financial%20Crisis%20Diagnosis%20Paper.pdf>, retrieved 24.8.2012.

Sahota, Michael (2012) *An agile adoption and transformation survival guide: working with organizational culture*. < www.infoq.com/minibooks/agile-adoption-transformation>, retrieved 10.10.2013.

Sathe, Vijay (1983) Implications of corporate culture: a manager's guide to action. *Organizational Dynamics*, Vol.12, No. 2, 5-23.

Scaled Agile Framework (2014) <http://scaledagileframework.com>, retrieved 3.4.2014.

Schein, Edgar H. (1991) *Organisaatiokulttuuri ja johtaminen*. Gummerus, Jyväskylä.

Schein, Edgar H. (2001) *Yrityskulttuuri – selviytymisopas. Tietoja ja luuloja kulttuurimuutoksesta*. Laatukeskus, Helsinki.

Scrum Alliance (2010) <http://www.scrumalliance.org>, retrieved 23.11.2010.

Snowden, David – Boone, Mary E. (2007) A leader's framework for decision-making. *Harvard Business Review*, Vol.85, No. 11, 68-77.

Stacey, Ralph D. – Griffin, Douglas – Shaw, Patricia (2000) *Complexity and management. Fad or radical challenge to systems thinking?* Routledge, New York.

Schwaber, Carey – Leganza, Gene – D'Silva, David (2007) *The truth about agile processes*. Forrester Research. <www.rallydev.com%2Fsites%2Fdefault%2Ffiles%2FThe_Truth_About_agile_Processes_Forrester_white_paper.pdf>, retrieved 5.1.2014.

Schwaber, Ken – Beedle, Mike (2002) *agile software development with Scrum*. Prentice Hall, New Jersey

Smith, Greg – Sidky, Ahmed (2009) *Becoming agile in an imperfect world*. Manning Publications, United States of America.

Smits, Hubert (2007) The impact of scaling on planning activities in an agile software development context. In: *Proceedings of the 40th Hawaii International Conference on System Sciences*. IEEE Computer Society, Washington DC.

Morien, Roy (2005) agile management and the Toyota way for software project management. In: *Proceedings of 2005 3rd IEEE International Conference on Industrial Informatics (INDIN)* 10-12 August, 2005, 516-522.

Tessem, Bjørnar (2003) Experiences in learning agile practices: a qualitative study. In: *Proceedings of the 4th International Conference on Extreme Programming and agile Processes in Software Engineering (XP 2004)*, 131-137.

VersionOne (2011) State of agile development survey results. <http://www.versionone.com/state_of_agile_development_survey/11/>, retrieved 25.8.2011.

Winston, Royce (1970) *Managing the development of large software systems.* <http://leadinganswers.typepad.com/leading_answers/files/original_waterfall_paper_winston_royce.pdf>, retrieved 26.8.2011.

Yin, Robert K. (2003) *Case study research. Design and methods.* Sage Publications, USA.

Yin, Robert K. (2009) *Case study research. Design and methods.* Sage Publications, USA.

Yusuf, Y.Y – Sarhadi M. – Gunasekaran, A (1999) agile manufacturing: the drivers, concepts and attributes. *International Journal of Production Economics*, Vol.62 , No.1/2 , 33-43

# APPENDIX I 12 PRINCIPLES OF AGILE MANIFESTO

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

- Welcome changing requirements, even late in development. agile processes harness change for the customer's competitive advantage.

- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

- Business people and developers must work together daily throughout the project.

- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

- Working software is the primary measure of progress.

- agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

- Continuous attention to technical excellence and good design enhances agility.

- Simplicity--the art of maximizing the amount of work not done--is essential.

- The best architectures, requirements, and designs emerge from self-organizing teams.

- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

APPENDIX II OPERATIONALIZATION TABLE

| The research problem | The sub research problem | The theoretical equivalents | The main themes of the interviews | Relative interview questions |
|---|---|---|---|---|
| Why managing agile software development transformation is experienced challenging in the case organization? | What characterizes Agile software development process compared to plan-driven process? | Agile principles and practices;<br>Plan-driven software development;<br>Scrum;<br>Complex adaptive systems | Software development process prior and after Agile transformation;<br>Organizational culture prior Agile | 2, 3,8,9,14,18,25,35 |
| | What kind of challenges the case organization has faced during the transformation to Agile software development process? | Change leadership;<br>Organizational culture; | Organizational culture;<br>Management style;<br>Development process;<br>Team dynamics | 5,6,11,12,15-17, 9,21,22,24,27,28, 30,31,34 |
| | How the mitigate the challenges? | Practices development;<br>Change leadership;<br>Team empowerment | Mitigation efforts | 7,13,23,29,32,33 |

# APPENDIX III INTERVIEW QUESTIONS

1. How long have you have worked for Nokia / in this team and how long experience you have from applying agile software development practices in your work?

2. How would you describe the development process and mindset before agile transformation?

3. What in your opinion makes a development process agile?

4. How much you received training of why agile process works when the agile transformation started?

5. How would you describe the main challenges of the agile transformation process in your organization in the beginning?

6. Are these challenges still present and have you faced any other challenges along the process?

7. How these challenges were mitigated?

8. How would you describe the organizational culture within the team before the agile transformation?

9. How the organizational culture has changed after the agile introduction?

10. How did the company wide organizational change in the turn of the year 2012 change your role in the agile software development process?

11. How do you think this same organizational change affected your team's agile transformation process and progress?

12. Have you seen any increase in silo-thinking in your organization when agile has been in introduced? Has it grown after the company wide organizational change split teams?

13. What was most difficult for you as a manager/leader to learn about agile?

14. How do you see the role of management in the agile development process compared to traditional plan-driven type of development?

15. Do you think middle and upper management understands the agile ways of working and how do they support you in that? How do you see the agile values and principles are aligned with the middle and upper management goals and objectives?

16. Do your other stakeholders, e.g. project managers, business, local factory teams, understand the agile ways of working and how do they support you in that?

17. How well do you think your organization has succeeded in scaling agile? What kind of practices for scaling you have adopted?

18. How do you manage dependences e.g. cross-capability EPICs requiring development from several teams?

19. What kind of challenges from agile perspective your team has faced when multiple simultaneous EPICs have been on-going within your team?

20. How do you measure team performance and velocity development?

21. Have you experienced any challenges in agile ways of working related to the team dynamics and people interactions in your team?

22. How is your team using its empowerment to make decisions and self-organize? Is this authority fully used and if not, what could be the reason for preventing it?

23. Has there been change resistance towards agile methodologies in your team?

24. Has the team communication increased after agile transformation? Have you noticed any challenges in communication practices?

25. How do you manage cross-organizational communication?

26. How is your current requirements management process handled? How well you think the process works?

27. How would you describe the challenges in sprint or release planning in your opinion?

28. How do you address design and architecture decisions in your Scrum ways of working?

29. How the agile practices used in your team in general have developed over time?

30. During the sprint, what do you consider as most challenging in balancing the workload and reducing waste?

31. What are the main factors or issues which have prevented your team to reach the sprint goal?

32. What would you do differently if you should participate again to an agile transformation process?

33. How would you describe the factors which have led to success in agile development in your team?

34. What are the present main challenges that you consider your team should focus next to improve?

35. Do you think your team will continue with Scrum development process and agile mindset also in future?

36. Is there anything else you would like to add that you think is interesting in this context, but not covered by the questions asked?