LEADING SOFTWARE PROJECTS

Applying agile principles

Master's thesis

Degree Programme in Strategic Leadership of Technology-based Business

Visamäki 23.05.2014

Anssi Aamurusko

VISAMÄKI
Degree Programme in Strategic Leadership of Technology-based Business

Author Anssi Aamurusko **Year** 2014

Title of Master's thesis LEADING SOFTWARE PROJECTS
Applying agile principles

ABSTRACT

Agile methods are commonly used in software development and teams are encouraged to apply those methods. Training courses and literature to increase understanding about agile methods is widely available for software developers but the offer for project managers is not that wide.

This thesis is commissioned originally by Nokia to find out how the project managers are leading agile software development teams and how the team applying a specific method should be leaded. Nowadays the researcher and the target group of this study belong to Microsoft.

The theories utilized in this study are related to agile project management, leading self-organizing teams, leading subcontracted teams and the theory of agile software development methods.

Research method used was focused interview. Interviews were recorded and partially littered. The data was categorized and further analyzed by grounded theory method.

The target groups were mainly applying agile methods so freely that it was not possible compare management practices against certain method. Instead the practices from whole group were analyzed against agile principles to define general guidance for project management in the target group.

Main obstacles for applying agile methods in the teams were for example mixed and distributed team structure and too wide customer base. Instead supporting and encouraging team for self-direction and continuous improvement are examples of actions that assist agile methods applying.

Keywords Agile methods, software development, project management

Pages 60 p. + appendices 4 p.

HAMK
HÄMEEN AMMATTIKORKEAKOULU

VISAMÄKI
Teknologiaosaamisen Johtaminen

Tekijä Anssi Aamurusko **Vuosi** 2014

Työn nimi LEADING SOFTWARE PROJECTS
Applying agile principles

TIIVISTELMÄ

Ohjelmistosuunnittelussa käytetään yleisesti ketteriä menetelmiä ja suunnittelutiimejä rohkaistaan näiden menetelmien soveltamiseen. Ohjelmistosuunnittelijoille on runsaasti tarjolla koulutusta ja kirjallisuutta ketterien menetelmien omaksumiseen mutta tarjonta projektiesimiehille on huomattavasti vähäisempää.

Tämän opinnäytetyön on alun perin tilannut Nokia selvittääkseen kuinka projektiesimiehet johtavat ketteriä ohjelmistosuunnittelutiimejä ja kuinka tällaista tiimiä, joka soveltaa tiettyä menetelmää, tulisi johtaa. Nykyään tutkija ja kohderyhmä työskentelevät Microsoftin palveluksessa.

Teoria, jota tässä työssä sovelletaan, sisältää ketterää projektijohtamista, itseohjautuvien tiimien johtamista, alihankitun työn johtamista ja ketterien ohjelmistosuunnittelumenetelmien käytäntöjä.

Tutkimusmenetelmänä tässä opinnäytetyössä on käytetty teemahaastattelua. Haastattelut nauhoitettiin ja litteroitiin osittaisen litteroinnin menetelmällä. Aineisto purettiin luokittelemalla ja edelleen analysoitiin ankkuroitu teoria menetelmää käyttäen.

Tutkimuksen kohderyhmässä ketteriä menetelmiä sovellettiin niin vapaasti, ettei ollut mahdollista verrata johtamistapoja tiettyä menetelmää vasten. Sen sijaan toimintatapoja analysoitiin ketteriin periaatteisiin verraten tavoitteena määritellä yleinen ohjeistus kohderyhmän projektiesimiehille.

Tuloksista esimerkkeinä ketterien menetelmien soveltamista haittaavina asioina olkoon hajautettu ja sekava tiimin rakenne ja liian laaja asiakaspohja. Sen sijaan tukeminen ja kannustaminen itseohjautuvuuteen ja jatkuvaan kehittymiseen ovat toimia jotka edistävät ketterien menetelmien soveltamista.

Avainsanat ketterät menetelmät, ohjelmistokehitys, projektijohtaminen

Sivut 60 s. + liitteet 4 s.

TERMS AND ABBREVIATIONS

ASD	Adaptive Software Development
Crystal	An agile software development method
DSDM	Dynamic Solutions delivery model
FDD	Feature Driven development
Genchi-Genbutsu	Japan, means "go and see"
IIBA	International Institute of Business Analysis
IID	Iterative and Incremental Development
Kaizen	Japan, means "improvement" or "change for the best"
Kanban	software development process
Scrum	An agile software development method
TPS	Toyota Production System
WIP	Work-In-Progress
XP	An agile software development method, Extreme Programming

TABLE OF CONTENTS

1 INTRODUCTION

This chapter introduces the reader to the background and ordering organization of this study. Also the structure of this study is represented.

1.1. Background for this research

Deployment of agile methods in software development is nowadays very common habit and in many cases software development teams are even required deploy some of agile methods in their projects. Background for this kind requirement could be that by implementing agile methods projects can be finalized faster or with less resources, when compared to earlier popular software development methods, like waterfall process and its' variations.

Agile method trainings for software developers are widely available and also a lot of literature about the issue has been written by several authors. Anyway dedicated literature about project management practices have not been so well covered by the authors of the agile software development books.

Running a software development project is newer a predicted series of action. It cannot be compared to a manufacturing process, which is in most cases predictive and controlled process. Controlling the output of that kind of process is mainly controlling the flow of materials and scaling the size of manufacturing operations.

Software development is made by human people with their intelligence, not by machines and the operators, thus leading software development teams and managing software projects needs different management skills than running a factory manufacturing some products.

This study is ordered by Nokia and by the group which develops software applications for cellular device manufacturing and aftermarket operations and research and development applications running on cellular device itself. Teams and their work products will be anonymous as request by Nokia. It is possible that until this study is finished the hosting company will be Microsoft, but it should not have any impacts to this study and the target teams.

In 2012 Koivusalo made a thesis about measuring the maturity for teams in the same target group than this study is now done. Actually most of people working in teams are the same but organization and responsibilities has changed quite a lot. The findings in Koivusalo research are still relevant.

Findings on study where, that management suffered practical knowledge of agile methods, even though they have had training and their teams are

applying agile methods. Also making schedules and work amount estimates was difficult for management. Management was also tempted to manage the work of the developers, which was against agile principles. (Koivusalo 2012, 61-63)

Researcher itself has experience of software project management for about 10 years from the beginning of the 2000 until today. It has mainly been leading a team using waterfall method, with strong requirement for up-front design and planning. Implementation was executed incrementally and also in maintenance phase software design was improved iteratively.

Researcher has experience about working in projects partially using agile methods. It has been a team leader activity in a large scale global software project. In that setup some teams were planning and executing by following waterfall method and some other were applying agile Scrum method. That setup worked pretty well, but in some cases those agile teams had resistance to change their iteration content to correct an error causing problems on dependent software components. Waiting to next iteration was frustrating for other teams.

Researcher worked as a product owner for a team using Scrum method. First was a trial of three moths which ended with the collective decision to go back to waterfall model. There were problems with sharing implementation responsibilities and chopping requirements to small enough tasks. Even the idea of changing to agile development was initiated from team itself it did not succeed. The main reason for failure in applying of the agile method was that the expertise within team was so specialized to individuals and knowledge and responsibilities could not be shared.

At the time when the researcher started to plan this study the researcher was also starting in role of product owner of one scrum team. It sounded a good idea to improve own knowledge about agile methods and leading the teams applying those methods. The team started to work pretty well and iteration after iteration the process was settled comfortable for developers and customers. Unfortunately after few months of working on this role of a product owner, a big organization change in company caused this team to be split and integrated to other teams. Thus it was the end of the product owner role within that team.

1.2. Research problem and outlining

The target of this study is to find out how the product owners or the project managers are leading and managing their software development teams from agile software development point of view. Agile values and principles formulate basis for project manager behavior in agile environment. Two research questions for this study are:
-How the agile values and principles are applied in project management?

-What are the management practices suitable to particular agile software development method?

This research is outlined to six selected application software development teams. The environment in those teams might not be comparable to different software development environments. Although these teams are developing computer applications the work has similarities to embedded software development. These applications have dependencies to very low-level layers of the software in the mobile devices. This study can't be generalized in other industries than software development. Anyway the leadership parts are quite general.

1.3. Structure of the research

The research was initiated in December 2012 by creating a research plan. The research then continued actually September 2013 and continued then actively by theory studies. On February 2014 it was time to start empirical research and after that the work continued with research result analysis and finalizing this study report. Report was finished on May 2014. This report document is divided into five parts which are described in the following subchapters.

1. Introduction

First chapter open describes the background for this research. The questions to be answered are described in this chapter. Also there is mentioned previous research on the same working environment. Research outlining is described in technical wise and also the teams to be studied are introduced. The structure of study is explained here and the sources for theories are criticized, both literature and researcher own experience in subject area.

2. Theory

The target in this study is to study agile software development management, thus it is obvious to find information about agile software development methods and about leading agile software development teams. XP and Scrum methods were assumed to be used at least in some of the target teams, thus those methods were studied thoroughly form the point of management and other methods were just went through by their characteristic properties. Some teams were using Kanban process, which required studying that process and also Lean Software development, which is the basis for the Kanban process.

From the management and leadership point of view there is information about Lean management and leading self-management teams. All except one team were using subcontracting at least for testing and many also for development. Thus it is clear that some background information is needed from that area as well.

3. Research and analysis

Ontology, the fundamental categories of being, used in this research is intersubjectivity. A common understanding of the research area is required between the researcher and target person of the research. Epistemology, the theory of knowledge, used in this research is antipositivism, as the understanding of the person behavior is based mental processes and by all means is subjective. Paradigm, a distinct concept or thought pattern, used in this research is interpretative hermeneutics, as the research requires human understanding. Logic is inductive reasoning as the target is to generalize the research results. (Niiniluoto 1997)

The research method family is qualitative and from that family an interview is selected as a research method. More precisely a semi-structured or focused interview is method to implement empirical research. Rationale of the method selection is described in the beginning of the research chapter. The implementation and the practicalities of the interview are explained there, too. The actual slides used for supporting the discussion are presented in Appendix 2.

The interview recordings are transcribed by using partial littering, meaning only remarkable sentences are littered and all non-meaningful words and sentences from discussions are left out. The actual littering is not included into this document.

The Analysis of littered interviews is made by categorizing and using grounded theory method.

4. Discussion

Discussion chapter includes reliability and validity evaluation of this research. There is also researcher's reflection to whole research process and how well the targets were reached. Possible future study subjects are also evaluated there.

5. Contribution

Contribution chapter gives a view how the research result can be utilized and what is the value of those for the company, the target group and the researcher itself.

1.4. Source criticism

The main sources for the theory are described in this chapter. Sources are also criticized if it is seen necessary.

Craig Larman is an advocate for iterative and agile development methods. He is not dedicated to any specific method, but handles several different methods in his books. David J. Anderson is a pioneer of utilizing Kanban technique in software development and nowadays he leads his company of consulting, training and publishing business promoting evolutionary approaches for managers of the knowledge workers. Alistair Cockburn is one of the authors of the Agile Manifesto and thus known and trusted

source of information. Cockburn has described the Crystal method family for agile software development. Mary and Tom Poppendieck are the originators for lean software development, as well as Jesper Boeg is known to be an advocate for it. IIBAs' Agile extension provides an independent view for agile software development methods. Their view point is in business analysis, not in the methods comparison. Tutorialspoint has a large collection of training materials related software development. They do not provide sources for their materials, thus it is researcher decision to trust that information. Auer and his co-writers in the book Ketterää Kehitystä are writing about their own experiences and opinions and there are no source references in that book.

Michael Hackett is one of the founders of Locigear company specialized for software testing services. He has been authoring two books about software testing and has long career on that subject area. Steve McConnell is a famous software consult for many global companies and has been authoring several books software development area. Jeffrey Liker is well-known writer and consultant for TPS.

For empirical research implementation the theory and guidance is mainly from the book authored by Sirkka Hirsjärvi. She is a professor and researcher on pedagogics area and has been authoring several books about qualitative research methods and about research work overall.

One source of information is the researcher itself and his experiences. Researcher has participated training for Scrum product owners and agile management training. And like mentioned earlier in this chapter also in practice Scrum method has become familiar. Some of the experiences grew some resistance against agile software development in the mind of the researcher. Anyway the last experience gave a positive feeling about agile software development and it should balance the attitude of the researcher between traditional and agile development methods.

2 THEORY

The theory chapter first introduces agile software development and lean software development. Then the agile project management practices are presented. Next there are leadership principles for self-organized teams and subcontracted teams are discussed and also lean management. Then the agile software development methods are described, mainly XP and Scrum methods, but also few others are shortly explained.

2.1. Modern software development

Iterative and evolutionary development (IID) is a foundation for modern software development methods in which also agile methods belong. IID is an acronym for Iterative and Incremental Development. Development and requirements are handled in an evolutional manner. Planning is adaptive and driven by the priorities from customer or by the requirement risk level. Releases are scheduled with time-boxes. (Larman 2007, 9-13)

The target of single iteration is to release stable, integrated and tested partially working system. Single iteration can contain in addition to coding also requirement analysis and design activities. One can say that the product grows incrementally iteration after iteration. (Larman 2007, 10)

2.1.1. Agile software development

In 2001 The Agile Alliance created a manifesto and statement of principles, also known as manifesto of the Agile Alliance. Agile project management is guided by these principles and agile practices reflect to these principles. (Larman 2007, 27)

Agile manifesto reads as follows:

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

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

That is, while there is value in the items on the right, we value the items on the left more." (Agile Alliance 2001)

The first value can be understood like undocumented with good interactions is better than a documented process with hostile interactions. The second value expresses that running the code is the only visible proof of the progress of the project. Documentation is something that can be used to support the team work. The third value highlights the joint decision making between the team and customer. Contract may be needed to establish collaboration, but not needed any longer. The fourth value is

explained like every project needs a plan, but within agile methods planning and development cycle is short to be able to respond changes quickly. (Cockburn 2007, 371-372)

And the twelve principles of agile software deepen the meaning of values in practice:

“

- 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.

” (Agile Alliance 2001)

Delivering early gives possibility for quick wins and early feedback about the requirements, the team and the process. Delivering frequently enables continuous wins for the team, fast feedback and up-to-date priorities. (Cockburn 2007, 373)

Software delivery frequency should be as short as possible. It is a matter of customer, how often one can take in the new delivery for their review. If the frequency is months, it is suggested to use intermittent, light mid-reviews. (Cockburn 2007, 373-374)

Working software proofs to the customers how the project is progressing. It tells more about the situation than plans and documents. Working software has to main functionalities, the user interface and the algorithms. Both can be evaluated by customer and valuable feedback can be gathered. (Cockburn 2007, 374)

Agile processes are able to take on late-changing requirements because of early and frequent delivery of working software, iterative and incremental delivery and iteratively evolving architecture and design. (Cockburn 2007, 374)

Daily cooperation with developers and business people or end user representatives is one success factor for the project. Daily availability is not practical in most of the projects, but it can be said that stakeholders need to be available on short notice when a discussion is needed. (Cockburn 2007, 375)

Motivated individuals are keys to success in projects. Management should provide the tool and training and support to get their work done and then keep out from their way to leave them space to express themselves. (Cockburn 2007, 375)

When people are working near each other and easily accessible, information can be shared immediately and problems can be solved without any delays. There is no need to arrange a meeting and move people to common meeting room. Face-to-face conversation possibility removes the delays from communication.

Self-organizing teams are able to adjust the architecture and design during the project life-cycle. Architecture can't be predesigned too strictly, otherwise it can't adapt to changing requirements during implementation. (Cockburn 2007, 375-376)

Sometimes one must make shortcuts on design to get things done and build working software. When this happens it is called technical debt and has to be paid back later. It means the design needs to be fixed later when the team has better understanding of what is the correct or optimal design approach. (Cockburn 2007, 376)

Overtime working is sometimes needed, but if it is continuous it will cause people to get tired and ineffective. It will lead increased errors in implementation and more work effort is needed to fix errors. If continuous overwork is needed, then one should start evaluating the project layout and find the problem areas which are causing the overload. (Cockburn 2007, 376)

Simplicity in design and implementation makes it easy to understand for everyone in the team. It is important to accomplish a requirement by implementing only the most important features and may be the rest are not even needed by the customer anymore. (Cockburn 2007, 377-378)

Improving the team work habits regularly can be said to be agility in work methodology. If the team does not evolve its working methods, it is not effective and adaptive and it will stay on where it is. (Cockburn 2007, 378)

2.1.2. Lean software development

Mary and Tom Poppendieck have translated Toyota Production System (TPS) original Lean principles into Lean software development principles presented in table 1. Lean software development has its ground in Kanban thinking, which is customer-oriented manufacturing model. Lean software development shares many principles and ideas with those presented in agile manifesto, but lean is giving more weight to continuous improvement of delivery process and change management.

Lean software development principles target to minimizing and eliminating waste, which can be for example unnecessary documentation or implementation. Too early decisions in requirement processing or in implementation can lead to producing waste into final product, as well as negligence of thorough testing produce unnecessary rework.
(Poppendieck M. and T. 2003, 2-4)

It is very important to recognize these wastes like work or phase product which do not bring any additional value for the customer. When the wastes are recognized, those will dropped out, starting from the ones which eat the most work effort. Finally when all the waste is removed, it comes obvious that even some mandatory process parts or phase products were only included for extra certainty, not for the customer value Poppendieck M. and T. 2003, 2-4)

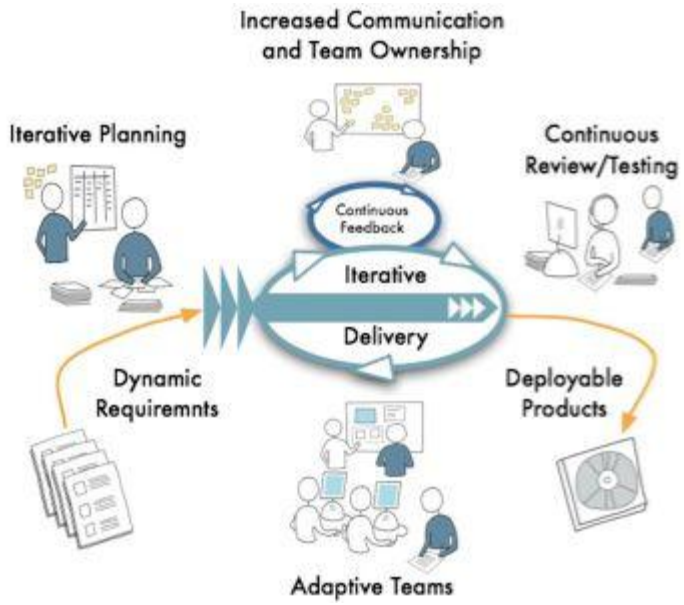
Fast delivery needs early decisions, but delivering fast gives an opportunity for fast feedback and fast learning. To make decisions individually and deliver fast teams need to be enough empowered. (Poppendieck M. and T. 2003, XXV-XXVII)

Table 1. Seven Lean Software Development Principles as defined by Mary and Tom Poppendieck with the DELIVER mnemonic (Equinox, 2011)

	Lean Principle	Description
D	Deliver Fast	**Deliver value to the customer quickly**, rapid delivery; high quality; low cost **Queuing theory** to Limit Work in Process (WIP) and context switching **Manage workflow is easier than managing schedules,** using repeatable workflow
E	Eliminate Waste	Waste is anything that does not add value to the customer. The three biggest wastes in software development are: 1. **Building the wrong thing:** building features that aren't needed 2. **Failure to learn:** policies that interfere with our ability to learn 3. **Thrashing:** anything that interferes with smooth flow of value
L	Learn Constantly	**Predictable performance is driven by feedback:** rapidly respond to change **Maintain options;** keep code change tolerant, minimise irreversible decisions Defer commitment, schedule irreversible decisions to **Last responsible moment**
I	Build Quality In (Integrate Quality)	**Final Verification should not find defects!** Prevent with executable requirements **Mistake proof your process with test first development** to establish correctness **Break dependencies:** architecture should support addition of any feature at any time
V	Optimize the Whole (Value the Whole)	**Focus on the entire value stream** from customer request to deployed software **Deliver a complete product,** a complete team delivering not just the software **Think long term** rather than local optimization
E	Engage Everyone	**Autonomy:** Empowered self-organising feature teams with effective leadership **Mastery:** Provide challenge and environment which enables people to grow **Purpose:** Tie the work to value and a common vision
R	Keep Getting Better (Relentless Improvement)	**Failure is a learning opportunity:** investigate and correct them as they occur **Standards exist to be challenged and improved** **Use the scientific method** Plan-Do-Check-Act process

2.2. Agile Project management

Project management for agile software development requires good understanding of agile development process. Picture 1 illustrates generally an agile development process.



Picture 1. Agile development process (Tutorialspoint 2013)

There are many differences in agile development model when compared to traditional models. Tutorialspoint lists four of them:

- The agile model emphasizes on the fact that entire team should be a tightly integrated unit. This includes the developers, quality assurance, project management and the customer.

- Frequent communication is one of the key factors that make this integration possible. Therefore, daily meetings are held in order to determine the day's work and dependencies.

- Deliveries are short-term. Usually a delivery cycle ranges from one week to four weeks. These are commonly known as sprints.

- Agile project teams follow open communication techniques and tools. These techniques and tools enable the team members (including the customer) to express their views and feedback openly and quickly. These comments are then taken into consideration when shaping the requirements and implementation of the software.

(Tutorialspoint 2013)

2.2.1. Responsibilities of an Agile Project Manager

ASD method creator Jim Highsmith has listed nine principles for agile project manager:
“

1. Deliver something useful to the client , check what they value
2. Cultivate committed stakeholders
3. Employ a leadership collaboration style
4. Build competent, collaborative teams
5. Enable team decision making
6. Use short, time-boxed iterations to quickly deliver features
7. Encourage adaptability
8. Champion technical excellence
9. Focus on delivery activities, not process compliance activities”

(Larman 2007, 29)

Two managers, Augustine and Woodcock, with experience in projects applying XP method, have listed six recommended practices for agile project managers.
“

1. Establish a guiding vision for the project and continuously reinforce it through words and actions
2. Facilitate collaboration and teamwork through relationship and community
3. Establish and support the team’s set of guiding practices, such as Scrum and XP
4. Provide visible and open access to project management and other information
5. Apply just enough control to foster emergent behavior of self –directed team
6. Reinforce the vision, follow or adapt the rules, listen to the people”

(Larman 2007, 29)

According to Tutorialspoint following are the responsibilities of an agile project management function. From one project to another, these responsibilities can slightly change and become interpreted differently.

- Responsible for maintaining the agile values and practices in the project team.
- The agile project manager removes impediments as the core function of the role.
- Helps the project team members to turn the requirements backlog into working software functionality.
- Facilitates and encourages effective and open communication within the team.
- Responsible for holding agile meetings that discusses the short-term plans and plans to overcome obstacles.
- Enhances the tool and practices used in the development process.

- Agile project manager is the chief motivator of the team and plays the mentor role for the team members as well.

(Tutorialspoint 2013)

And the following are manners to avoid by agile project manager:

- Manage the software development team.
- Overrule the informed decisions taken by the team members.
- Direct team members to perform tasks or routines.
- Drive the team to achieve specific milestones or deliveries.
- Assign task to the team members.
- Make decisions on behalf of the team.
- Involve in technical decision making or deriving the product strategy.

(Tutorialspoint 2013)

2.2.2. Scope of Agile Project Management

In agile projects, it is everyone's (developers, quality assurance engineers, designers, etc.) responsibility to manage the project to achieve the objectives of the project. It is not just the project manager's responsibility. The common sense is used over the written policies, like processes and procedures.

Agile project manager plays a key role in agile team in order to provide the resources, keep the team motivated, remove blocking issues, and resolve impediments as early as possible. In this sense, an agile project manager is a mentor and a protector of an agile team, rather than a manager.

The agile project management function should also demonstrate the leadership and skills in motivating others. This helps retaining the spirit among the team members and gets the team to follow discipline. Agile project manager function facilitates and coordinates the activities and resources required for quality and speedy software development.
(Tutorialspoint 2013)

2.3. Leadership

This chapters introduces lean leadership and the leading the self-managed teams. This should give some ground for the management of the self-organized teams like agile teams should be. Subcontracting is said to be used in the teams that will studied, thus leading the subcontracted teams is also an issue which needs some background information.

2.3.1. Lean management

Liker writes about Toyota way of lean leadership. It is actually a model of continuous development of leadership. There are five core principles of which the leadership development is based on. These principles are:

1. **Challenge**. We form a long-term vision, meeting challenges with courage and creativity to realize our dreams
2. **Kaizen**. We improve our business operations continuously, always driving for innovation and evolution
3. **Genchi-Genbutsu**. Go to the source to find the facts to make correct decisions to build consensus and achieve goals at our best speed.
4. **Teamwork.** We stimulate personal and professional growth, share the opportunities of development and maximize individual and team performance
5. **Respect**. We respect others, make every effort to understand each other, take responsibility and do our best to build mutual trust

(Liker & Convis 2012, 30-33)

According to Liker and Convis leaders do grow by continuously improving four aspects.

1. **Self-development**. Managers actively develop themselves and their skills with the help of a mentor, a more experienced leader.
2. **Developing others**. Manager should participate mentoring and coaching of all ones subordinates. The best measure of your success as a leader is what was accomplished by the people you coach.
3. **Supporting daily Kaizen**. Teach the correct application of maintenance Kaizen, to cope with daily problems, and improvement Kaizen, to find better ways of working. As a trainer and coach, like you experienced yourself as a trainee, you challenge people to improve, but you do not force them to accept your own solutions.
4. **Creating vision and aligning goals**. Manager and organization together set the targets for the long-term development and decide the division of the effort and resources for reaching those targets.

After the managers have reached the high level of lean leadership, they are capable of adapt to continuous and vast changes provided by their business or working environment. (Liker etc. 2012, 33-37)

These lean management principles support project manager with agile teams. Managers can develop themselves in the areas of agile processes and leadership, and they can also coach their team members and other managers on these skills.

Managers should create and communicate the vision of team and together with team set the targets. Division of the effort and resources is interesting as usually it is higher level management who decides the resources for project. It is project manager responsibility to indicate these needs to higher management to get the resources and environment for successful execution.

2.3.2. Leading Self-managed teams

Self-directed and self-organizing are the words used to describe agile development teams. Thus it is worth to find some information about leading self-managed teams. Yeatts writes about the roles of management in his book about the self-managed teams. These roles include providing support and encouragement to the team. Also there are mentioned roles like assisting team's dimensions and acting as a team resource and coach.

Managers usually observe their team's performance and whenever they see low scores measures they are keen on why it has happened. In case of self-managed team the manager should rather ask what are going to do to improve the situation and how management could support the initiatives. It is important to make clear message that the team is responsible for their performance and productivity by itself and the manager is always available for supporting the team on its actions. (Yeatts 1998, 190-191)

Team members need encouragement in setting own performance goals, to establish high expectations for themselves. They need encouragement for self-criticism on the decisions they have made, before they start implementation on the basis of those decisions. Also the encouragement is needed for team members to evaluate their performance and take actions if it is low. And in positive case when performance is in good level they need to take care of rewarding themselves. (Yeatts 1998, 192)

Assistance from manager is important, not only making decision and solving problems. A manager should be able to give tools and methods to team for decision making and problem solving. It is also important to assist team by searching and sharing information which could improve the performance of the team. One more important assistance task for manager is seeking and providing training and education programs for team. (Yeatts 1998 193-194)

A manager being a resource for team, means helping the team to have enough resources they need. Coaching is important to be provided early enough, before any failures happen. Self-managed teams need coaching in both technical and management responsibilities. Usually team member are motivated to learn needed skills and thus do not need coaching from manager anymore. (Yeatts 1998, 195-196)

2.3.3. Leading subcontracted development and testing teams

Hackett writes in his article about the ten risks rising from leading an offshore software testing teams. Despite the background in the software testing service company his findings are true also in case of software development outsourcing or subcontracting.

1. Work effort measuring.

To be confident about the work effort the other company workers are investing to your project you need to have proper metrics in place to follow up the progress. Hackett (Hackett 2014) writes about test cases, how many of those are run and their results. In software development that amount of implemented or released task can give the information needed to follow up the effort of the developers.

2. Visibility to daily work

It is important to be aware of every developers and testers daily tasks and progress. This can be achieved by having a daily reporting from every team member to project leader (Hackett 2014). This might sound like micro management, but in case of remote team one does not have possibility to easily observe the progress of the team located near the manager.

3. Competent contact point

When the remote team is large or consists of several teams, it is essential to have a single, competent and reliable contact point to represent the effort of the whole team or project. This person should not be business account manager, but rather a technical lead. (Hackett 2014)

4. Plans for downtime

Remote teams, especially when located on developing countries can experience problems which can prevent them to perform the tasks they are supposed to do. Power outages, network problems, viruses and lack of needed equipment are such problems. Most important thing to do is to do all possible preventive actions to avoid this kind of problem. (Hackett 2014)

For such cases a contingency plan is anyway good to have. If it is impossible to do software development and testing work, it will be possible to do for example brainstorming for improving processes and tools.

5. Access to host company network and services

Like already mentioned in previous chapter, network problems can prevent remote teams to work. Especially in software development remote teams and host company exchange data in many ways and the databases and tools might common or provided by hosting company.

To avoid problems caused by network breaks or slowness there few things to improve the situation, for example databases can be replicated locally to

remote team. This minimizes the delay caused by occasional network breaks. Sometimes it is still possible to use e-mail even the development tools do not have connection between two sites. Of course you need to be sure that this kind of substituting method is secure enough. (Hackett 2014)

6. Trustful partner

When the distance between team and project manager is long, it can happen that remote team does not report the full truth about their progress on the work. They may want hide problems or lack of competence they may have. There is no other way to avoid this problem than having a contact person who you can trust. (Hackett 2014)

7. Attrition

When the work is done in the country with booming economies, it is natural that staff is frequently searching for better job opportunities and higher salary. And those really are available if economics are growing. It is also possible that subcontracting company reassigns best resources to another customer with better contract. (Hackett 2014)

To avoid attrition it needs to be built into contract that project personnel should not be changed. It should be also possible increase the compensation of the employees to be competitive. (Hackett 2014)

8. Culture and personality differences

Personality conflicts are possible inside any team or working environment. With remote there are certain things which make possibility for such conflicts even higher. Those are differences in culture, lack of face-to-face communication, inequality in task distribution and possible hostility as a consequence of outsourcing. Personality and culture differences hinder communication between remote team and hosting company project management. (Hackett 2014)

Cultural training for both parties is the most common means to minimize the risk of conflicts. Another good way is to gather the remote team and local people to same location to work together and get to know each other better. It can take a couple of weeks of time, but it helps on knowledge transfer and it also makes people closer to each other and by that way easies the communication.

9. Time for communication

When teams are distributed, communication has a big role in keeping all parties up-to-date. And communication in that case is more difficult due cultural and personality issues. This leads to situation that the communication takes a big share of the daily working time. The problems in communication cause easily the feeling that no one is listening. Proper listening takes time. (Hackett 2014)

Time difference makes effective communication harder, as it means that someone has to move out from comfort zone. It is important to have a

person on remote team who can effectively communicate both inside the team and to other parties outside. (Hackett 2014)

10. Language barrier

Language skills are different in different countries. In software development programming languages are the same globally. All technical documentation should be written in one common language, preferably English. This helps the communication as at least the words for technical issue are then familiar. Of course, if possible, providing language trainings helps the situation. (Hackett 2014)

Steve McDonnell has been writing several books about software development and he lists following things to keep in mind when outsourcing abroad.

1. Communication

In target country it is possible that services and connectivity are not in same level as in company home country. Common language is essential. Sometimes subcontracting companies provide a contact point on your language, but it is still possible that foreign languages are found from final delivery. (McConnell 2002, 496)

2. Time difference

Communication outside of office hours due time difference requires flexibility from either hosting company or sourcing company. If the communication takes place mainly by e-mails, the time difference gives more time to answer questions to both parties. Still the communication is slow. If both parties are using same services, like software building servers, those are in more efficient use around the clock. (McConnell 2002, 496)

3. Travelling

When cooperating with the team abroad, one must prepare for travelling. All things can't be handled by e-mails or phone calls. Face-to-face negotiations are the most effective way to communicate. It is suggested have on-site meetings at least in the beginning of the project and in the end during the project with couple of month's interval. (McConnell 2002, 496)

According to McConnell outsourcing has couple of well-known risk.

1. Losing visibility

Manager need to have means and tools for following up the progress of the project. . (McConnell 2002, 499)

2. Losing expertize

Expertize on the product or development transfer to outsourcing provider. And expertize on ordering company decreases. It may not wise outsource core competencies. . (McConnell 2002, 499-500)

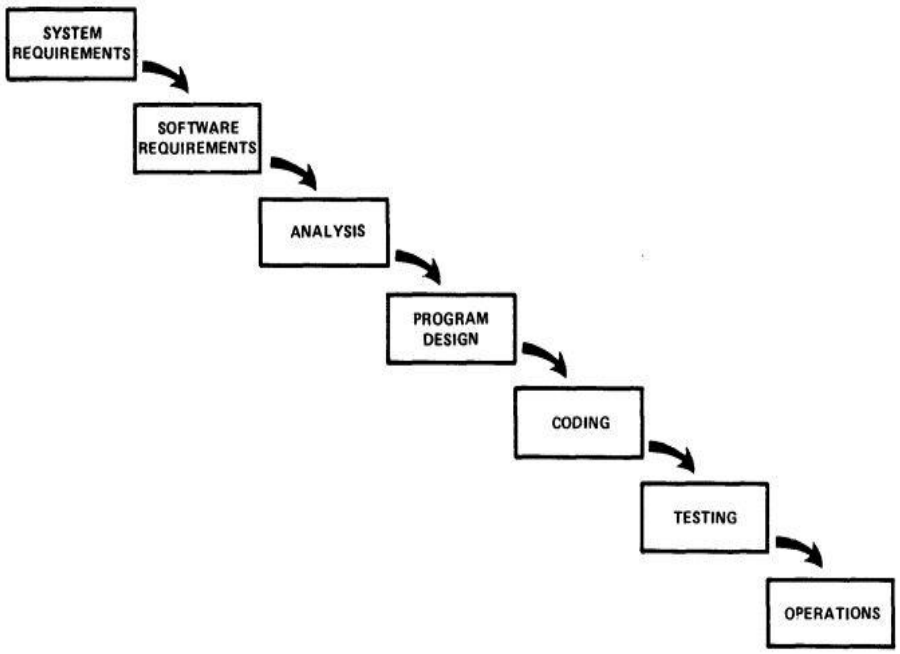
Often the subcontractor work on other town or other country than local developers and in large companies there sites where development takes place. Auer writes in his book that in this kind of setup it is best to formulate own Scrum teams to each site and share the same product back log. If the teams still need to be distributed to separate sites, the more they have opportunities to work together face-to-face, the more they converge. (Auer etc. 2013, 18)

2.4. Software development methods

This chapter introduces the traditional waterfall software development process and agile software development methods. Scrum and XP methods are explained more precisely and other methods properties are briefly discussed. Kanban process is also introduced.

2.4.1. Waterfall process

In waterfall process implementation phase can happen iteratively and incrementally, but the difference is in up-front design and predictive planning.
Waterfall has got its name from article written by Winston W. Royce in 1970. In that article he was originally demonstrating that a single-pass sequential approach can't work in software development.



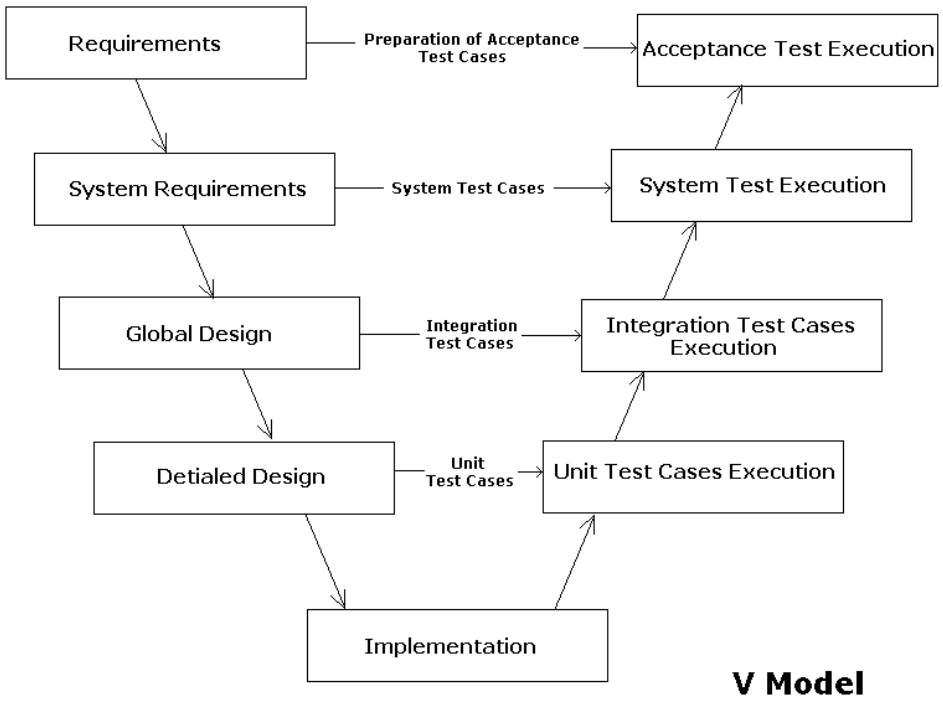
Picture 2. Implementation steps to develop a large computer program for delivery to a customer (Royce 1970)

Picture 2 illustrates the idea of the waterfall process. Previous step needs to be finished before continuing to next one. And the planning and scheduling of those steps in waterfall process is predictive.

According Haikala etc., the first phase, system requirements, also known as system architecture defines the software relationship to hardware in embedded systems. Second phase defines the requirements for the software, how the software is divided to components and how those connect to each other. Also the performance requirements can be defined. (Haikala 1997, 24-25)

Analysis phase includes definition of software features, external requirements and restrictions, communication methods with other systems. (Haikala 1997, 25-26)

Design phase outcomes following work products: architectural design, module design, detailed design. This phase is mainly answering to question – How the software is implemented. (Haikala 1997, 26)



Picture 3. V-Model illustration (Software Testing Class 2013)

Coding, or programming phase includes writing all the program code needed. Next phase, testing, links to coding and these two phases are commonly called implementation phase. Testing takes place in all the levels above the implementation phase, too. This is illustrated by the V-model in picture 3. In that model the testing is divided to module testing, integration testing and system testing. System testing is driven by system

design and requirements, integration testing is driven by architectural design and module testing is driven by module design. (Haikala, 1997, 26)

The last phase, called operation in picture 2, includes deployment and maintenance of the software product.

2.4.2. Scrum

Scrum is may be most famous agile development method. According to Larman (Larman 2007, 109) the key practices of Scrum method are:

- Self-directed and self-organizing team
- No external addition of work to an iteration
- Daily stand-up meeting with special questions
- Usually 30-calendar day iterations
- Demo to external stakeholders at the end of each iteration
- Each iteration, client driven adaptive planning
- Avoidance of prescriptive process

(Larman 2007, 109)

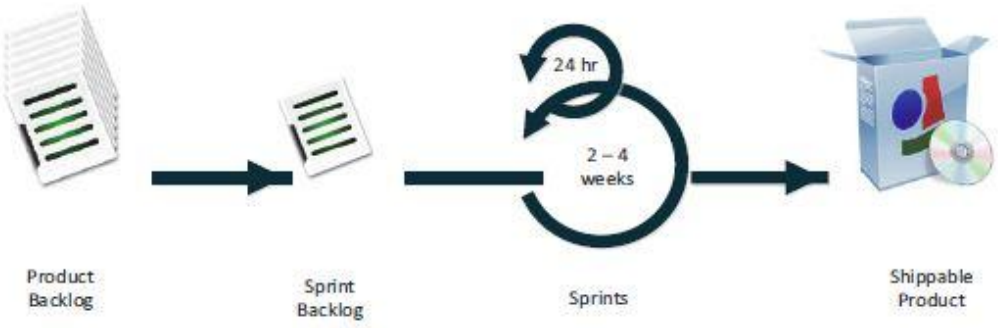
Three different roles can be defined for Scrum method: Product owner, Scrum master and the team. Fourth one is customer. Product owner is responsible for maintaining the list of work to be done and keeping in touch with customers and other interfacing parties. Scrum master responsible for managing the Scrum process and related ceremonies. The team is responsible for development delivery of the working product.
(IIBA 2011, 9)

Iterations are called sprints and the result of the sprint is a working software release, which can be delivered or demoed to customer. Software requirements are listed on the product backlog. Backlog includes both technical and customer requirements. Items in backlog are kept in priority order which gives guidance for the team for selecting most important tasks for development. (IIBA 2011, 7-9)

In scrum there are four formal meetings: sprint planning, the daily scrum, sprint retrospective and sprint review. In the beginning of each sprint a sprint planning meeting is held. The purpose of the meeting is to select highest priority items from the product backlog which can be implemented and finalized during the sprint. These selected items constitute a sprint backlog. Product owner should not affect to these selections even one usually participates the meeting. During the sprint the team refines these items and divides them to smaller tasks. (IIBA 2011, 7-9)

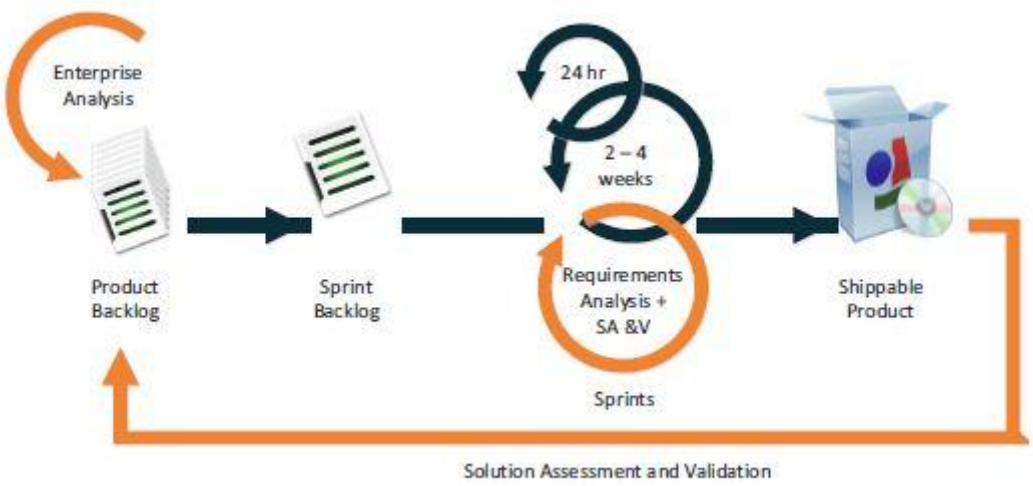
Every day during the sprint the development team meets in daily scrum meeting. It is short meeting where developers discuss their current work items and possible blockers for work continuation. Product owner presence is not mandatory on these meetings.
(IIBA 2011, 7-9)

At the end of the sprint team delivers working and tested software which implements all backlog items selected to that sprint. Sprint review is held with customer and the delivered software functionality is demonstrated to customer. Customer has opportunity to give feedback in review meeting and propose new items to product backlog. Sprint retrospective is held mainly within development team and its purpose is to improve the product or the processes used for delivering the product. Retrospective can also produce new items to product backlog. (IIBA 2011, 7-9)



Picture 4. Scrum development life-cycle (IIBA 2011, 10)

Picture 4 illustrates Scrum life-cycle development team point of view and picture 5 from management point of view. Populating product backlog and keeping it in the priority order, is away how product owner or project manager controls the work of the team. Scrum framework itself does not give too much guidance for backlog management. Product backlog is built through identifying gaps and new capabilities to fulfill organizational goals. Also the customer feedback about latest delivery generates new items to backlog. And the solution assessment and validation gives improvement ideas from the existing solution. (IIBA 2011, 9-10)



Picture 5. Scrum management life-cycle (IIBA 2011, 10)

2.4.3. XP

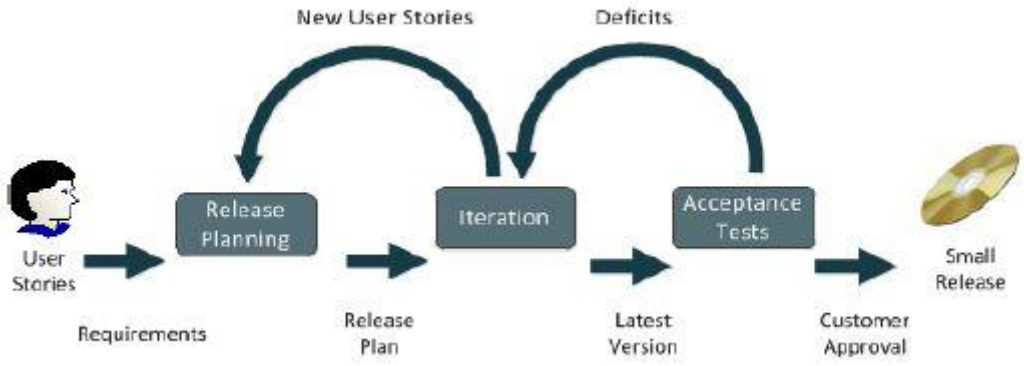
XP is a well-known agile method. It has 12 practices which are characteristic for this method. According to Larman these practices are:

- Planning Game
- Small frequent releases
- System metaphors
- Simple design
- Testing
- Frequent refactoring
- Pair programming
- Team code ownership
- Continuous integration
- Sustainable pace
- Whole team together
- Coding standards

(Larman 2007, 137)

Main roles in XP are Customer, developer, and tracker. The customer is responsible for creating user stories and outlines the risk analysis for those. Developers communicate directly with customers and develop only what is necessary for next iteration. Tracker is responsible for schedule updates and metrics gathering and presenting.

In XP requirements are defined as user stories. They are created by the users of the system to define features and functionalities of it. User stories do not give detailed description of wanted feature or functionality but they will help prioritizing work in iterations and identifying risks estimating the needed effort and opening the conversation between development team and product owner to get common understanding what really needs to be implemented. (IIBA 2011, 11)



Picture 6. XP development process (IIBA 2011, 13)

Team development work and output is planned in three phases, illustrated in picture 6. Those are: release, iteration and daily planning. Release plan describes what features and functionalities compose a product release, which can be delivered to customer. User stories related to next release are ordered on priority based on importance for customer. Team chops the stories to tasks and next iteration content is built about these tasks. Common agreement about the tasks to fit into next iteration needs to be settled down with the whole team. It may take several iterations to reach the release delivery phase and get customer approval. (IIBA 2011, 12) (Larman 2007, 143)

The project manager within XP is mainly responsible for tracker duties like collecting metrics and making them visible as well as giving feedback on estimations. Also the process conscience and customization is a management responsibility in XP method. (Larman 2007, 145)

2.4.4. Agile methods summarized

Kainulainen has made a very good summary of agile methods and their properties. Appendix 1 is the original table from his thesis and a translation is provided here in table 2. In his thesis Kainulainen (Kainulainen 2008) has studied agile software development method strictly and critically, without getting influenced with marketing sentences of the method creators. Scrum and XP are described in more detail earlier and those others are only summarized in this table.

Table 2. Agile method properties

Method /Property	Scrum	XP	ASD	Crystal	FDD	DSDM
Design phase length	Partly defined, short but adequate	Partly defined, nearest to short but adequate	Strictly defined, thorough and long	Not defined	Strictly defined, fairly thorough and fairly long	Strictly defined, thorough and fairly long
Iteration usage	Light design, iterative implementation	Light design, iterative implementation	Cyclic design phase, iterative implementation	iterative implementation	Cyclic design phase, iterative implementation	Cyclic design phase, iterative implementation
Length of Iteration	30 days	2-4 weeks	4-10 weeks	2-4 months	2 weeks	Not defined
Customer participation	Design, iteration planning, reviews	Daily, as an active team member	Design, Iteration review	Not defined, but active	Design, not defined for iterations	Design, Iteration review
Customer technical knowledge	Not required	High	Not required	Not defined	Not defined	Not required
Process definition level	Strict process definition, methods freely selectable	Superficial process definition, phase products and methods strictly defined	Process defined, methods freely selectable	Not defined	Process and phase products defined in detail, methods freely selectable	Process and phase products defined in detail, methods freely selectable
Phase products	4	6	5	Not defined	5	7
Size of team	Not limited, scalable	Small, below ten members	Not limited, scalable	Different process for different sizes of teams	Not limited, designed for large projects	Not limited, used for large projects

2.4.5. Kanban process

On the basis of these Lean software development principles Kanban method was established. Kanban is Japanese word meaning a signboard or billboard and in TPS it model for customer oriented manufacturing model. Kanban is not actually a software development method, it can be thought like a change management process focusing on the following principles mentioned by Boeg:

- **Visualize Work** - Visualize every step in your value chain from vague concept to releasable software.

- **Limit Work-In-Progress (WIP)** - Set explicit limits on the amount of work allowed in each stage.
- **Make Policies Explicit** - Make the policies you are acting according to explicit.
- **Measure and Manage Flow** - Measure and Manage Flow to make informed decisions and visualize consequence
- **Identify Improvement Opportunities** - Create a Kaizen culture where continuous improvement is everyone's job.

(Boeg 2012, 13)

Kanban is often combined with other agile methods. It relies on the work queues to manage the flow of activities towards final working product. Backlog can be used as a source for inbox items in work flow. As the amount of work-in-progress items is limited a technique called grooming is used to optimize the items waiting in the queue. If those are too large to be completed for the next release, project team can analyze them and break them down to smaller items. Those smaller items are then prioritized and reordered to queue. (IIBA 2011, 15)

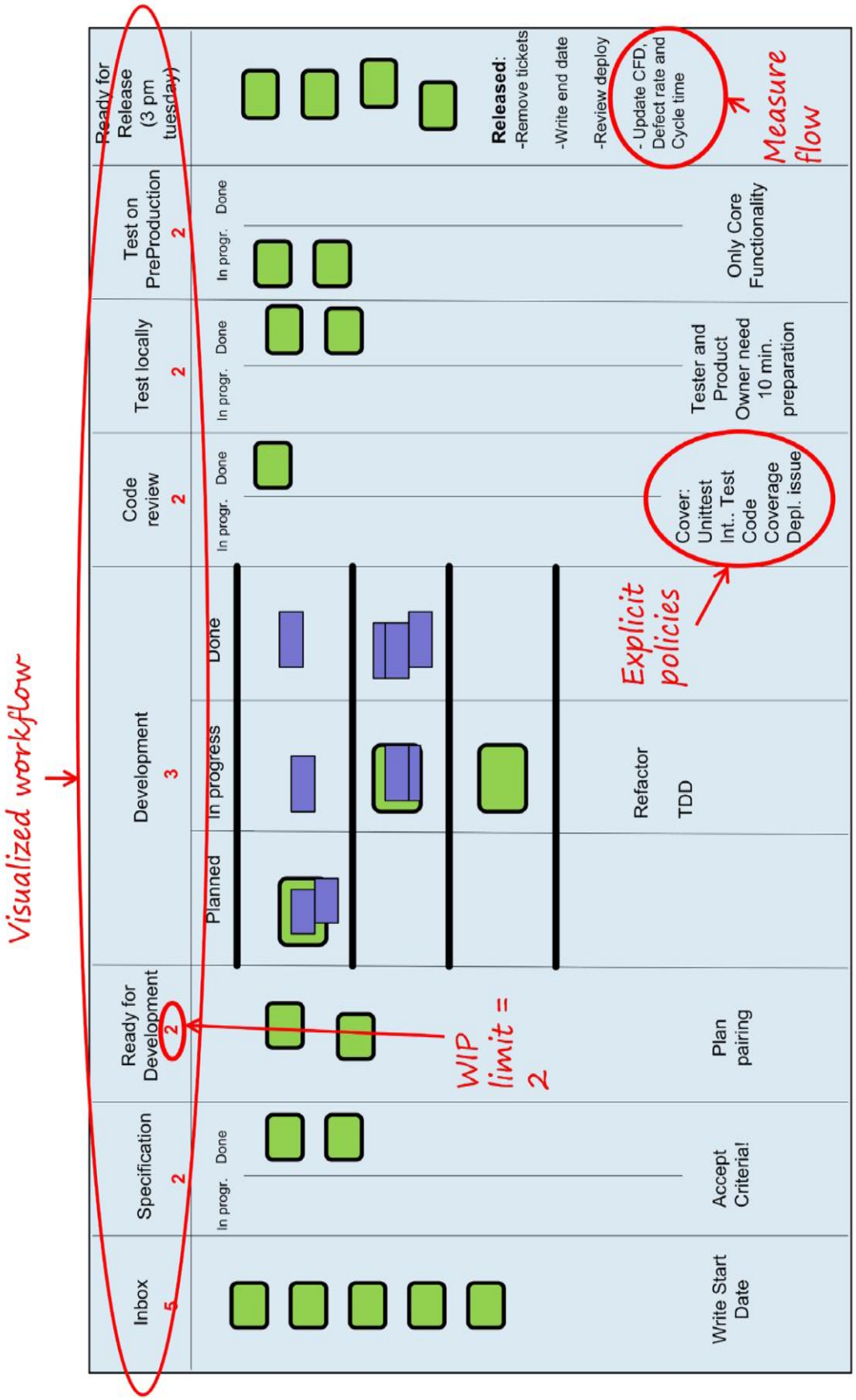
The Kanban board is the communication method for the team to manage their work. Picture 7 illustrates such board. It gives very clear visualization about what is going-on in team and implements the principle 'visualize work'. Work-In-Progress is very clearly visible in every step of development as are the explicit policies. Measurement of flow takes place after every release.

Kanban board gives visibility to whole process. When the work flow is transparent everyone can see the bottlenecks and change the process to flow smoother. It supports identifying improvement opportunities. (Anderson 2010, 60)

The method of pulling the work from requirement repository, Inbox in picture 7, empowers developers to make decisions without supervision or direction from supervisor. (Anderson 2010, 60)

In addition to a Kanban board or a card wall, it is preferred to use some application to track the work in Kanban process. Those make it easier to get metrics for retrospective use. And also with those applications it is easier to control the limits and barriers set for the work in progress. (Anderson 2010, 81-82)

Kanban is suitable for situation where that team has many customers but only few resources. It does not mean that Kanban solves the problems caused by broad customer base. By using Kanban the team can control their current work load and in due course they learn how to optimize work flow. Kanban process needs to visible to customers as well and project management needs to communicate with the customers about release contents and priorities. (Auer etc. 2013, 63-64)



Picture 7. Example of Kanban board (Boeg 2012, 15)

3 EMPIRICAL RESEARCH

3.1. Research problems

The goal of the empirical research is to find out answers for the research questions:

- How the agile values and principles are applied in agile project management?
- What are the management practices suitable to particular agile software development method?

Agile values are pretty sublimed and agile principles provide more practical approach to those values. Agile principles should be used as a guideline for project manager to enable a working environment which is suitable for agile development.

The second question requires knowledge about the method the team is using. The third question or actually the first question is then:

- What is the agile method the team is using?

3.2. Research method selection

Ontology, the fundamental categories of being, used in this research is intersubjectivity. A common understanding of the research area is required between the researcher and target person of the research. Epistemology, the theory of knowledge, used in this research is antipositivism, as the understanding of the person behavior is based mental processes and by all means is subjective. Paradigm, a distinct concept or thought pattern, used in this research is interpretative hermeneutics, as the research requires human understanding. Logic is inductive reasoning as the target is to generalize the research results. (Niiniluoto 1997)

The original idea in research plan was to create a survey and try it out with target persons. A survey with predefined questions and predefined choices for answering sounds easy to do. Pretty soon after studying some theory about agile methods, it became clear that there won't be too many possible questions which can be answered with Yes or No or simple with a number. And what comes to management or leadership questions there will so much variation caused by the personality and environment that there is has to be room for discussion and wider explanations. Because there is no need to prove the theories studied in this thesis but rather study the behavior of the target persons against the theory, this kind quantitative research method is not suitable. According to Hirsjärvi (Hirsjärvi S. & Hurme H. 2004, 27) the quantitative methods are more suitable for the situations where one wants to prove a theory with some measurable results.

More suitable method family for this research is qualitative. Hirsjärvi says that qualitative methods are typically used in research which tries to reveal personal experience or behavior. (Hirsjärvi etc. 2004, 27)

Historical study, group discussion or focus groups, observation, case study and interviews are known qualitative research methods. Historical study is mostly meant for the studies where one wants to understand the present state and plan for future on the basis of the historical phenomenon. Group discussion is good for getting a lot of answers while investing only short amount of time for interview. The main difference to a person-to-person discussion is that the communication happens also between the interviewees and thus impacts the information. Observation is listening and learning about person behavior in natural environment. Case study is mostly used for building new theories usually from unknown research area. Interviews are always person-to-person discussions and commonly known as the best methods for the acquisition of the information. (Eskola J & Suoranta J. 2012, 86-95, 99-104, 128)

For this research the group discussion, observation and interview sound like the most suitable methods. The teams of target persons are so different to each other and there it is hard to get all persons to a common session, group discussion can be left out as method. Observation could work in this case but it would take consume too much time of researcher itself. Interview is left as the most suitable way of implement the research.

According to Hirsjärvi there are four different types of interviews used in researching. Those are: structured -, semi-structured -, unstructured - and in-depth interview. In structured interview questions and categories for answers are standardized and interview event is similar for every interviewees. Unstructured interview leaves a lot of freedom for interviewee to discuss about reactions, behavior and opinion about issue at hand. Interviewer's task is to guide the discussion with leading questions and record the discussion for later analyzing. In case of semi-structured interview the interviewer must have defined the subject matter, the questions and interviewees. Still there is flexibility ask other questions and discuss freely. In-depth interview gives more information about interviewee's statement and behavior as the questions are open and the interviewee has freedom to answer according to one's own will. (Hirsjärvi etc. 2004, 43-47)

Semi-structured interview seems to be the most suitable method for this research, since there will be certain question which needs to be answered, but at the same time there needs to be freedom to discuss beyond the questions. According to Hirsjärvi this method is also known as The Focused Interview (Hirsjärvi etc. 2004, 47)

3.3. Implementation of the interview

The interviewees were invited personally to this interview. All participants agreed without hesitation. Formal invitation was send by e-mail two weeks before scheduled time for interview. The interviews were executed during February and March on 2014.

The interviews were facilitated in a meeting room to have enough peace and isolate interviewee from daily work environment. Two hours of time was reserved for discussing through the subjects. Interviews were facilitated by the researcher itself. Interviews were carried through in Finnish as all interviewees were Finnish speaking. Interviews were recorded and permission for that was confirmed with interviewees beforehand. Transcription of the recordings is done in English. Method for transcription is selective littering were only meaningful sentences are littered. Full content of littering is not included to this research.

3.3.1. Interview support material

Six slides were used as a teaser or background for discussion. These slides are presented in Appendix 2.

First slide has a generic picture of agile software project life-cycle. It helped to discuss what happens during the whole project and also daily and weekly practices. Interviewees were also asked to describe their team structure and roles in the team.

The second slide listed four agile software development values and the target of the discussion was to find out how these values are applied in interviewees' work and how the team is applying those.

The third slide was about agile software development principles and the target of the discussion was to find out how these values are applied in project interviewees' work and how the team is applying those.

Table 1 was represented as the fourth slide. It was about Lean Software development and was used to deepen the discussion about principles and practices used inside the team and by the interviewee.

The fifth slide was about leading self-managed teams and management development. The idea was to find out what the interviewee thinks about self-organizing and self-directing teams and how interviewee supports the team in this context. In addition it was discussed how the interviewee has developed own skills in the area of agile management.

The sixth slide was about working with outsourced remote resources. There were listed main problem areas when working in this kind of situation. There was only one team which did not have any remote

subcontracted members. The target here was to collect experiences with remote resources.

3.4. Analysis of the interview results

In this chapter the interview results are analyzed to find answers or solutions to research problems. The analysis is performed by the researcher itself. No one else has studied the littered material as a whole. Time spend on interviews varied between 75 and 124 minutes, average being 101 minutes. Thus the reserved time of two hours was enough for interview.

All participants had a positive attitude against the interview and discussion about the topics. Some free comments to mention from participants:

- "This was a good reminder about agile principles"
- "Good discussion which made me think these issues"
- "Interesting discussion about team leading"
- "this was a good retrospective for own working methods"

First in the analysis the team structure is clarified to better understand and categorize the opinions from interviewees. This information is available directly from littered material as it was directly asked in the beginning of the interview.

Then the agile method used in the team is tried to clarify by comparing to table 2 from theory chapter. The Interview results are then categorized against agile principles. Even there was an own slide for agile principles, the relevant answers and opinions from the whole interview are mapped those principles. A table for each principle and findings for each team is created to make it easier to combine or compare the results. The analysis method used here is called classification according to Hirsjärvi (Hirsjärvi etc. 2004, 147-148). On the basis of these results it is possible to find obstacles and success factors for agile project management with certain agile method and in general the information about agile principles appliance in the teams.

It turned out that only one team was applying an agile method purely, in this case Scrum. Other teams were applying methods freely and it became obvious that method specific management practices are not possible to define from these results. Instead the target will be to gather general good leadership practices for target teams and also things to avoid in their daily work. The method for analysis is grounded theory and it is explained more detail in chapter 3.4.3.

3.4.1. Structure of the teams

Overall description of team structure and the product those are delivering is given in this chapter. It is important to understand the lay-out of the team and the product it is delivering.

Team A is developing one application running in computer and an application running in mobile device. Team has subcontracted one testing

person and two software developers from Poland. In addition there is a local software architect and three developers and one engineer for testing and integration.

Team B is developing several computer applications for aftermarket customers, typically operators and service companies. Team has a local product owner and subcontracted development team. There are a Scrum master, seven design engineers and two engineers for testing and integration in the team. Team is utilizing Kanban process.

Team C is developing a software tool for cellular network operators. The team consists of local technical lead developer and one developer and two half time resources for development work. Subcontracted resources in Poland are one test engineer, one user interface designer and one software developer. Team is applying scrum method.

Team D develops toolkit which is released separately for cellular device manufacturing and for care operations. Team D consist of local product owner and three local developers and six subcontracted members, one developer in Tampere and four developers and one test engineer in Poland.

Team E develops a software module for cellular device manufacturing. In addition to product owner team consists of scrum master and four software developers and one test and integration engineer.

Team F is developing three software tools for aftermarket service points. The team has a local product owner and a subcontracted development team consisting of a scrum master, three developers, a build manager and three test engineers in Poland.

3.4.2. Applied agile method in teams

In this chapter researcher tries to find out if a certain agile method is applied by the team. Table 3 collects characteristics related agile methods. By comparing to table 2, it can be seen that teams E and C are applying Scrum method pretty well. In team C the roles were not optimal for Scrum method, the Product owner and Scrum master roles were combined.

Table 3. Agile method characteristics by team

Team /Property	A	B	C	D	E	F
Design phase length	Inherited code	Short	3 moths	short	4 months	Inherited code, framework driven
Iteration usage	daily builds	daily builds, small increments	Requirements , design, implementation	Daily builds, small increments	Daily builds small increments	Daily builds , small increments
Length of Iteration	Not defined,	No sprints , bi-weekly release	3 weeks	Not defined	2 weeks	Not defined
Customer participation	Weak	Regular meetings with PO	Regular meetings with PO	Weekly meetings	Daily when needed	Varies
Customer technical knowledge	Weak	Weak	Good	Weak	weak	Good, varies
Process definition level	No process	Low, Kanban	High, Scrum	Low, big tasks	High , Scrum	Low
Phase products	Predictive plan, backlog	Demos, backlog, roadmap, releases (5 products)	Release, product and sprint backlog	Two products, product backlog, stories, test cases	Weekly, bi weekly and official release, product and sprint backlog, demos	Product backlog, release plan, demos
Size of team	11	10	5+ 2xhalf time	9	6	7

Original pre-assumption before interviews was that scrum is mainly used. Many of the teams were saying they are using scrum method and roles were named like in scrum. But still there seems to be quite a lot of freedom in implementation of method and it looks like only team E is applying Scrum method purely. And it also came visible that Scrum has been promoted by the company as five of these six interviewees have been participating in Scrum product owner training. Also other scrum trainings where mentioned as well as agile management trainings. In theory part of this study it was noted that there is not such agile method which fits to every project and environment and it is not wrong to combine best parts from different methods.

Only one team is not enough to start define good leadership practices for certain agile development method in this software development environment. Thus the target will be to gather general good leadership practices for target teams and also things to avoid in their daily work.

3.4.3. Agile principles implementation in teams

The grounded theory is commonly known methodology for developing new theories from systematically gathered and analyzed data. Theories evolve from continuous analysis and data gathering. When using grounded theory for interview material analysis, the data is categorized by coding. There are three defined types of coding. Open coding is the first and it happens already during data gathering by observation and evaluation. The second is axial coding, which means that certain characteristics are selected for further analysis and categorization tiers to these characteristics. The third is selective coding which is targeting to find the core category from the whole research material. (Hirsjärvi etc. 2004, 164-165)

In this chapter the interview results are first categorized against agile principles. All the opinions from each interviewee related to each principle were collected to tables and then summarized. After each table there is collective analysis where these findings are compared and combined to find useful practicalities supporting agile principle and also point out working methods or circumstances which are breaking against the principle and ideas behind that principle. These summaries provide the results for the research question about applying agile principles. From these collective analysis can be found features for and against agile development process. These features can be special to one or several agile principles. This kind of analysis can be said to be grounded theory with axial coding.

Table 4. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Team	Findings
A	Concurrent integration of the new implementation and daily software builds makes it possible to deliver the latest version of software to customer whenever needed.
B	The product is developed for customer, and delivering first version for customer fast is a way to get confidence that we are doing the right thing. Developing with small increments which do not break the whole are a way to provide customer new functionality frequently.
C	Customer release can be done whenever there is need for it, manual testing takes some time though. After two months from coding a first draft release to customer was ready and an official release in five months. Quick and continuous delivery prevents implementing wrong things. Implementing the most risky items first makes the challenges to get the first working version out. Incremental implementation and early delivery helps customer to fine tune the requirements and unnecessary work can be avoided.
D	Early delivery to customer and continuously with small increments is target for team. Anyway, unclear priorities given by customer lower the value delivery sometimes. Wrong things and done in wrong time.
E	Four months lead time to first customer release. Currently weekly delivery for customer review. Sprint length is 2 weeks and official customer releases are made when the needed functionality requested by a customer ready.
F	Internal release can be built and delivered to customer for testing purposes whenever needed. External release is tested and verified official release, which is done when a new feature is ready for publishing.

According to table 4 all teams are targeting early delivery of valuable software to customer. It still has been two to four months for the first delivery, where this value was recorded. It does not mean that it is perfectly working release, but customer can evaluate solutions and give feedback. It is important to be sure that the requests are understood correctly and they are still valid. There seems to be some variation how often customer can get a new delivery. Most teams are making automated daily builds and integration, which gives possibility to make an unofficial test release to customer whenever needed. Those teams which are using sprints are usually providing delivery according to sprint cycle. Official release always needs more testing and integration effort and those are made only when there is a need by a customer and the requested functionality is fully implemented.

Table 5. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Team	Findings
A	Maintaining options open to the latest decision point gives room for late changes. Changes are usually caused by changes in depended other entities and the interfaces to those, anyway requirements are pretty stable. But requirements are quite big and it takes pretty long time to implement, thus surrounding environment can change during implementation.
B	Team welcomes changes, usually customer does not know in the beginning of the project what actually is needed. Responding to change quickly prevents late phase changes, which are always harder to implement.
C	Late changes are acceptable, if those benefit the customer. Sprint plans change sometimes against scrum principles, approval is handled in extended daily scrum meeting with customer. It is possible to implement sudden changes in next sprint and get those released. When code design needs to be changed, also the automated testing requires test environment changes. This causes change resistance sometimes inside the team.
D	Changes in requirements happen all the time, but those are rather small interface changes and can be responded easily. Those do not overload team. Changes in priorities between customers and their requirements cause context switching in development work. These priorities are not clear and product owner needs to put a lot of effort to find out correct order for the requirements.
E	Changes happen all the time and customer and developers collaborate to make last minute changes to delivery.
F	It has happened that customer requirements were not valid anymore when finally implemented. Situation has improved by having more discussion with customer. Also the project has moved to maintenance phase, thus there are not so many new risky features to implement. Features are implemented incrementally towards a release, but no sprints used for iteration.

According to table 5 the changes are often caused due different working environment. The team can have multiple customers and priorities between those may have not been clear when prioritizing requirements and tasks. That causes priority changes to team implementation order. Team can have more than one product to deliver. In that kind of case there are usually more than one customer as well and risk for sudden changes rises. In agile processes usually that kind of changes are added to backlog and prioritized for next iteration. Those teams which are not using frequent iterations it is possible to respond to changes fast, but it causes context switching for developers and unfinished implementation.
When a team has only one customer and one product to develop it is quite clear to make quick changes and satisfy the customer. Canceling the iteration and re-planning a new one can be a way to go.

Table 6. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Team	Findings
A	Dependencies to other entities are blocking working software deliveries, three different entity software deliveries needs to be integrated together. Thoroughly tested and verified software release takes several days to prepare.
B	Release cycle is two weeks and daily builds can be delivered to customer for testing and demonstration purposes. Developing with small increments which do not break the whole are a way to provide customer new functionality frequently.
C	Three weeks sprint frequency is used. Official customer releases are made only when applicable. Unofficial releases are possible to do with short warning.
D	Continuous integration and daily builds makes it possible to have a demo version for customers every day. Official customer releases are done when there is need for those.
E	Release is delivered weekly, but bi-weekly a fully tested release is delivered. Testing round takes about a week to get feedback from final integration. In theory a release can be done whenever needed but there has not any need for that. Grooming meetings were held weekly to fine tune release content. Not needed currently as there are not so many new requirements coming in.
F	Internal release made from daily build can be done to demonstrate customer an unclear requirement. Official release is done when a new feature is ready for publishing.

Table 6 tells that official, tested and verified releases for customer are made only when needed. This means that customer needs the stabile release and also the needed implementation is finished. Building an official release takes time from couple of days to two weeks, depending on team. Unofficial releases are available for testing even daily. Team E is delivering bi-weekly an official release. There has been also grooming meetings with customer to agree final release content.

Table 7. Business people and developers must work together daily throughout the project.

Team	Findings
A	Project manager collaborates with customer, not the team. New ideas arising from the team are verified against customer needs.
B	Product owner collaborates with business people about planning and new requirements and delivers info to team and vice versa. That can happen even in daily phases. With the subcontracted team direct communication with customer and developers did not work too well. Some customer representatives tried to manage too much team work and pushed through their own ideas.
C	Customer representative is available every day and customer can contact directly developers and architects. Product owner has regular customer collaboration, usually before sprint begins. Customer is involved in release content planning.
D	Customer collaboration happens weekly by requirement status follow-up meeting. Demonstrations for customers and end users have been held but useful feedback was hard to get. Customers do not have clear understanding about end-user needs. Local developers can contact customers and even end-users, if those are known, but subcontractor developers are not that well networked.
E	Scrum meetings are held daily by scrum master and developers. Product owner takes part meetings couple of times in a week. Developers and end users are in contact daily, whenever it is needed. Sometimes new requirements are gathered from end user and customer feedback.
F	Product owner handles high level collaboration with customer for plans and new requirements. Technical details are fine-tuned by customer and developers. Some customers are more active and have more knowledge than the others.

From table 7 one can see that only one team had contacts to business people and the others are thinking here the customer co-operation. None of the teams are working with customer daily throughout the project, but occasionally it can happen and customer representative is available whenever needed. It seems that when team has both local and subcontracted developers, the customer collaboration goes mainly through product owner. It is possible for customer to contact subcontractor developers directly, but they may not have access to all information needed for proper decision making. That makes it almost mandatory to have product owner or other local team member assistance in subcontractor developer and customer co-operation. Regular participation of customer is seen in demonstrations, but results of those are highly depending on customer activity.

Table 8. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

Team	Findings
A	Motivation is hard with inherited code base, if error correction only takes so much time that certain developers can't finish any development tasks. Working environment is anyway motivating and learning new things is encouraged. Also developers have autonomy for development tools and environments. Work has a clear meaning and huge end user group help to realize it.
B	Team has the responsibility to design and implement the requirements which are pre-described in high level. They do chop the requirement to smaller work items for implementation. The team has full trust from product owner to get the job done. Test engineers are responsible for developing and optimizing the test system, not only running the tests. Testing is valued inside the team and everyone understands how important it is to find bugs in early phase.
C	Part time allocated resources and remote team members are not equal compared to fulltime and local team members. Those usually get the challenging task and thus get motivation. Senior level specialists are comfortable with broader perspective tasks and easily take responsibility. Team internal feedback is given in retrospectives and it is emphasized to be positive. It brings up success stories which have not been recognized by product owner. Coaching is provided for positive feedback and active participation in meetings and for team working.
D	It is not always possible to select team members and vice versa developers can't always work in the team which makes most interesting development. Team members have different skill levels and some needs technical mentoring more than the others. In some cases tasks need to be assigned to developer. Especially subcontractors have top-down management mentality. Environment for growing is available, but it is marketed enough by management. Product owner drives and encourages team to make decisions autonomously. Mixed and scattered team structure seems to be bottleneck for effective working and causes leadership challenges. Solid and centralized subcontracted development team was easier to manage according to earlier experiences.
E	Team has a good spirit and a face-to-face conversation possibility helps to solve all problems immediately. Product owner supports team by providing tools environment and contacts. Coaching available for agile processes.
F	Product owner trusts developer and they are motivated. Subcontracted team needs quite a lot of support by means of arranging tools and materials. Motivation is kept up by providing challenging tasks. Anyway resources have been lost by attrition, because of better salary.

When reading through table 8, it can be seen that in three teams there where nothing negative mentioned about motivation. Common for those is that all members are located in same area, no matter is it a local team or subcontracted. In these cases it seems to be easier for product to trust the team and its' deliveries. Thus it seems that working together improves motivation. It is also mentioned that compared to current situation, earlier experience with a team located in one place was much better. The teams were members are divided to different sites and to local and subcontracted resources, seem to suffer equality in challenging tasks distribution. Local

resources seem to get the most interesting things and remote developers get straightforward jobs, which are of course easier to follow-up by product owner. Altogether task assigning in agile teams is not valued. Challenging tasks are mentioned to be one source for motivation and product owner should pay attention to distribution of those.

The importance of testing should be emphasized and product owner can drive the team to understand this fact and value the work the test engineers are doing. Independent development of test systems can be seen as a challenging and motivating task for test engineers, it means much more responsibility than just running the test cases.

From table 8 can be found few things which are product owner's responsibilities to keep team motivated. Communication of clear vision for team deliveries is important to build common understanding why the team exists and what is its' target.

Working environment is something were product owner can help the team. For the tools needed in work team or developers should have autonomy to select and product owner can help to get them. Product owner can help also by providing needed materials and hardware. One important thing is that product owner has contacts or one can arrange contacts which are needed by the team to solve problems. It was already noticed that customer are not regularly participating to team work.

Technical coach for the team should be scrum master or equivalent person. Product owner can coach team in more leadership related issues. Giving positive feedback is one important thing to keep people motivated, not to mention that also negative things should be brought to table. Product owner can encourage team member to feedback internally in team, it is motivating to be respected by colleagues. If product owner is a master in agile methods, coaching can be offered also in that sense. Product owner should be able to provide decision making encouragement and coaching, and of course methods for it.

Table 9. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Team	Findings
A	Local work environment arranged for better conversation possibility, meaning work places are located to same area near each other. Daily meeting happens face-to-face, but via online conference tool because team is distributed on two sites. Common agile board application is used instead whiteboard.
B	The remote team has locally a daily half an hour morning meeting. Every Monday an online status meeting with product owner and the remote team is held. Occasional on-line meetings can be called whenever needed. Remote team of experienced developers works well, but few face-to-face meetings have improved the communication a lot. Common agile board application is used instead whiteboard.
C	Team is divided to two locations and on both sites they are sitting near to each other, which make it easy to have face-to-face conversations. Anyway the whole team has never had a common face-to-face meeting. Daily meetings are online meetings. Video is not possible with remote team even it could improve the level of communication. Common agile board application is used instead whiteboard.
D	Team is distributed to three different locations and face-to-face conversations are not possible. Daily meetings are on-line meetings, where you can't be sure how much everyone is focusing on the meeting and it is possible that some problems are not taken into discussion and made visible to everyone. It would be best to have all team members located in the same site and near each other. Once the whole team was working in the same location for few weeks and productivity was much higher on that time. Common agile board application is used instead whiteboard
E	All team members are located near to each other and around the scrum table. Conversation is natural and open. Common agile board application is used with customers.
F	Face to face is the best way to communicate. Whenever it is possible to work together in same site, it is very productive. The whole development team is located in same area to improve communication. But daily communication with product owner happens in on-line meetings, in which all team members are present. Daily meetings are used for status sharing and prioritizing the next features to implement. Ad-hoc online meetings can be arranged if there is some problem which needs to be discussed. Chat is also handy communication tool. Common agile board application is used instead whiteboard.

Findings in table 9 tell that there is a consensus between all interviewees about importance of face-to-face conversation. In all teams work environment is arranged so that all team members are located near each other. In one team where the scrum board was used all team members have easy visibility to that board.

The face-to-face conversation possibility is broken in the teams which were distributed to different sites. Even they are locally working together and near each other, the conversation between the other parts of the team can happen only via on-line conference tools and messaging tools. Video conference tools could be the best for common meetings, but those are not available exclusive for own use. Those have not been very reliable within

the context of this interview, but there is no doubt that a reliable connection between two sites in different countries can't be arranged, it is just a matter of investment and the feasibility of it. It seems to be that presence of the product owner in team local working environment is not so important. Regular online meetings with team are enough to keep work going on to right direction, in the case where the whole team is in one location. In case of a remote team and specially subcontracted team one need to have daily visibility to team progress. Product owner participation to daily meetings is not suggested in agile process, but in this case listening of daily status without too much managing the team is one good idea to follow-up progress of the work. A written daily report does the same thing, though.

Even having a few face-to-face meetings can improve the communication, by better knowing each other and understanding their background. Altogether distributing a team to different sites is not according to this particular agile value and causes information breaks and more management work.

Table 10. Working software is the primary measure of progress.

Team	Findings
A	Software product has dependences to other entities which makes integration, testing and verification of official customer release quite tedious. Anyway customer can be provided with unofficial demo version for testing latest features.
B	Product can't be tested in all different setups, thus had to make a decision that only most common environments are covered. Testing the product takes as much effort as the implementation work. Error metrics are used as an indicator when the release is ready for customer delivery.
C	Bug fixing has highest priority and bugs are added to sprint backlog and prioritized higher than implementation tasks. Testing and verification happens all the time by nominated person. Official customer release requires more testing effort and overrides daily build testing. New test engineers need more attention from project manager (scrum master) to make sure they follow testing process correctly.
D	Working software is the only thing that matters to customer. Testing is not only a mandatory task, it is emphasized to ensure the quality. Team has test first thinking in development and all user stories have acceptance criteria included. Testing is automated and the test system and test cases are developed by test engineers themselves. Test system automatically creates test reports for easy follow-up for quality.
E	Automatic testing happens daily and generates a report for problems and for code complexity and testing coverage. All functionalities can't be tested by team itself, thus final verification happens factory production line. Official release is delivered after that final verification. End user feedback is usually delayed as factories are all over the world. Working software has more priority than documentation.
F	Making a verified official release requires two weeks of integration and testing effort in addition to daily automated test runs. Working software is available every day but not integrated to newest dependent software components. Every day a small amount of new functionality is added to delivery, ensures that we are doing the right thing and nothing else gets broken.

According to findings in table 10 all teams are investing on testing efforts. Automatic test systems ensure quality on daily builds. And the results from testing are available after test runs. This kind of daily testing is not comprehensive as it does not include latest deliveries from other dependent entities. In addition it is not possible for team to test delivery in final environment. Continuous integration and automated build and test systems make it possible to quickly provide the customer with draft version for evaluation. Guaranteed working software delivery is not easy to do and depending on team it will take time from couple of days to two weeks. Testing is valued amongst these teams and it is said that in some teams testing time is as long as implementation time. One team is targeting to develop test cases before implementation is started.

Table 11. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Team	Findings
A	Maintaining inherited code base can cause blocking situations which are difficult to solve and break the constant pace. This causes context switching. Following code complexity points out problem areas in source code.
B	Work-In-Process limiting is used to keep work amount in line with developers' capability. Due to several customers with different requirements and changing priorities the context switching can't be avoided and happens quite often.
C	Official customer releases cause temporary overload for testing. Context switching breaks constant pace, but it can't be avoided. Changes in depended entities cause priority changes for team development tasks. Work amount for each team member are kept sensible to avoid stress. Overtime is anyway sometimes needed and it is mainly compensated with time-off. When more resources are needed, problem needs to be communicated to management.
D	Technical skills vary so much inside the team that work amount can't be shared equally. A scrum master could help the situation by technical mentoring and coaching. Currently some developers have big and long lasting implementation tasks with strict schedules.
E	Product owner maintains constant pace by protecting the team against external pressure. That pressure has caused sometimes too early implementation and then lot of changes afterwards. Team tries to avoid single responsibility and aims to have at least two people who are capable for implementing the same feature. Thus workload can be shared evenly.
F	Subcontracted team home country regulation prevents too much overtime. Sometimes overtime is anyway needed and it is then compensated with time-off.

Findings in table 11 show that usual reasons to break constant development pace are sudden changes in requirements or priorities. This is likely to happen if team has several products to deliver and several customers. Consequence of these is context switching, which is frustrating or overtime working to get everything done in time. It is possible that lack of technical understanding causes overload, especially when the team has not been developing the product from the beginning and the source code is unfamiliar. Also if technical skills vary a lot inside the team, the more productive person need to support the others and at the same time hurdle to finish challenging tasks. It has happened that team has pressure from external stakeholder, high management or arrogant customer, to push through a requirement. Not to say it could be right thing to do for one party, but for team it causes overload for implementation and overload later on by means of bug fixing and late changes. Building the official release is in some cases overloading testing engineers.

There are few things that product owner can do to protect constant pace. Overtime can´t be avoided totally and when done it should be rather compensated with time-off, to allow people to recur. It is of course individual decision how to compensate overtime. Good thing is also to circulate different task so that at least two persons are always capable to

share the work load of certain implementation area in the product. Developers need encouragement and support for dividing their work to small pieces, this ends up to product owner responsibility if there is no scrum master or similar role in the team. Following code complexity measures helps to find out the risk areas in the code, if the project and the code are not familiar for the developers. If there is a constant need for overtime and the team is always overloaded, it is product owners responsibility indicate the need for more resources to management. It is also possible that the project layout is wrong and it is not possible to execute without overloading and burning out the developers.

Table 12. Continuous attention to technical excellence and good design enhances agility.

Team	Findings
A	Bi-weekly design meetings are held for ensuring the design quality. Especially when starting a new application development a lot of communication is needed inside the team to agree the main lines of the design. It is also good to have just enough documentation in the code to point out the decisions made in the implementation.
B	Target in implementation is to keep options open until the last decision point and to produce simple self-explaining code. Refactoring is used to improve design. Retrospectives are held whenever someone from the team sees the need for improvement, this one Kanban principle. Dependencies are hindering the optimal design approach, but those can't be totally avoided.
C	Developers have autonomy to make design decisions. Product owner can challenge decisions to start discussion and supports organizing design meetings. Dependencies to other entities are tried to minimize on design level. Senior level developers are usually quite enthusiastic to avoid bad design and technical debt is corrected as soon as possible.
D	Coding standard checking tool is used, but everyone is not committed to improving scores of it. Dependencies to other entities make it hard to maintain optimal design. Design meetings can be arranged ad-hoc, not regularly. Overall design and architecture is defined by development environment, thus there is not so much design work needed. Test case design is team own responsibility.
E	Good design is target for the team and in current setup technical excellence is learnt by doing and it is growing all the time.
F	Developers have the responsibility to follow coding guideline. Interface design agreements are respected to avoid problems in integration. Technical education and self-learning is emphasized by product owner.

From table 12 one can find few design related features. In the beginning of the project design issues and agreements require a lot of time and communication. It is important to respect agreements about interfaces. Design decision should be commented directly to code, it is not necessary to have separate design document. Technical debt can't be avoided, but teams are targeting to fix it as soon as possible by refactoring. When design decisions are made, it should be kept in mind that dependencies are not increasing, rather decreasing. Defining and applying coding standards ensures that everyone in the team can understand the code. It also enables the usage of code standard checking tools to follow up compliance of the standard. Technical education and self-learning is encouraged by product owner.

It seems that in most teams design decisions are made inside the team and it is not an issue for product owner to participate. Scrum masters and architect are natural leader for design development. For increasing technical excellence the product owner can support the team support the team by searching and providing information about training opportunities. And product can also arrange room learning in product backlog.

Table 13. Simplicity--the art of maximizing the amount of work not done--is essential.

Team	Findings
A	It is hard for some engineers to do only minimal necessary implementation and not to build everlasting solutions. Code complexity is a good measure to follow to keep code simplicity high enough.
B	Fast rough implementation of feature is often used to verify the feasibility. If the feature is seen as unnecessary, internally in team or by customer, it will not be implemented. Implementation is made in small increments which do not break the whole. The final product is self-guiding and there are no user manuals needed.
C	Implementing unnecessary features can be avoided by keeping product backlog features in priority order. Customer requirement meeting held before sprint planning to get latest changes on priorities. Incremental and frequent delivery to customer helps to find out what is the functionality they really want. Complicated parts of implementation are separated from optimal design.
D	Prototyping keeps options open until last decision moment, but it sometimes means unnecessary code. Tasks are often too big for iterative and incremental implementation. Customer requirements can include unnecessary features. Two separate product lines inside the team causes sometime less valuable features to be implemented, just to keep everyone busy.
E	Product owner needs to be aware of possible waste in software development and point out when there is a risk to produce waste
F	Simplicity in implementation is promoted, sometimes refactoring is needed to get rid of too complex solutions. Coming changes are tried to take into account in design.

In table 13 team C findings tell quite clearly a basic idea of the agile process. Product backlog is kept up-to-date and especially before sprint planning. It ensures the team selects right requirements and tasks for next iteration. And demonstration to customer before sprint planning gives valuable last feedback which can affect to priorities and sharpen the requirements in the backlog. This is optimal flow of process and truly helps to avoid implementing unnecessary features. And the shorter the sprint is and smaller increment the less wrong implementation can be done.

Prototyping is quite similar approach, but the prototype needs to be simple enough. Otherwise too much code will be trashed if the idea is not acceptable. If the requirements are not chopped to smaller task the iteration time may stretch to months and the developer loses the opportunity for valuable regular customer feedback.

In agile development it is important to deliver just what is enough for the customer, nothing extra which looks fancy or complicated code which requires high expertize to be written and read. Code complexity measures are good tool to follow up that code stays simple.

Table 14. The best architectures, requirements, and designs emerge from self-organizing teams.

Team	Findings
A	Team work is managed by project manager and development tasks are assigned by project manager. Project manager handles the customer interface as well. Anyway autonomy in handling the tasks is emphasized.
B	If project manager is too controlling, developers feel themselves as coding machines. Team has responsibility and power to do development according to its' own decisions, while high level target is common with project management. Sometimes coaching is given for team leader to solve problems on personal or technical level. Authoritarian thinking of the leadership has been removed and trust build on that.
C	Team has autonomy to plan sprint contents. Product owner guides the planning by prioritizing backlog and justifying the priorities. Quality targets are set by team itself. Product owner gives to team a clear vision what is expected from them and from their product. With remote team self-organizing is a challenge. Usually subcontractors do not have all information available and their autonomy is restricted. They are also used to top-down management culture, thus product owner (scrum-master) must encourage them or just assign the tasks to them.
D	Architecture is top-down driven, not much room for own ideas. Currently increments are too big and cause challenges in integration to external systems. Product owner emphasizes decision making inside the team. Tasks need to be assigned to developers. Technical coaching is needed in many cases. Team structure and deliveries are too complex to create an environment for self-organizing team. A scrum master with enough time for coaching and equal task assignment would be needed. Especially with remote subcontracted resources.
E	Product owner supports team members in decision making by giving ideas. Team members select their tasks independently and estimate their own work effort for it. That sometimes requires encouragement from product owner. Product owner gives coaching on agile process issues or at least assists to get coaching.
F	Communication inside team is open and support for problem solving and decision making is available. Senior developers are coaching and mentoring new resources. To be a self-organizing team requires time and experienced team members before it can really work.

Analyzing the findings in table 14 shows that self-organization comes from autonomy. Team must have responsibility and power to select and commit to tasks it's going to implement in next iteration. Important part is also mentoring and coaching inside the team from experienced team members to new resources. Product owner should avoid managing the team. Product owner can encourage and support the team by coaching in managing problems, decision making, own work effort estimation and agile processes.

From table 14 it can be seen, that self-organizing is not happening if the team is divided to different locations. There is still only one scrum master or team leader driving the agile process and the other sites are not equally positioned with local team. In some of the teams there are no team leaders and it is product owner responsibility to handle those duties. It means overload on product owner workload. Technical coaching and agile

process controlling is scrum masters or team leaders task and product owner should concentrate on customers.

Subcontracted teams seem to have a problem with authoritarian work model. They are used to do what they are told and not think that much by themselves. This requires extra effort from product owner or team leader to encourage them to make own decisions. It is also a fact that subcontractor do not have same information available than local employees, which make it even harder to make decisions.

In the findings there are comments that it has taken years to change the attitude in the subcontracted teams. Also the experienced team members are mentioned to be role models for self–organizing.

Table 15. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Team	Findings
A	Monthly retrospectives were used earlier. Now bi-weekly design meetings include the retrospective if needed. Discussion about problems and failures are an opportunity for everyone to learn how the system works.
B	Team does continuously process improvement and the product optimization. Retrospectives are held whenever someone from the team sees the need for improvement, this one Kanban principle.
C	Retrospectives are held in the end of every sprint. Long term (half a year) retrospective also held. Subcontractors need encouragement to participate into improvement of methods and processes.
D	Retrospectives are not working within on-line meeting. Face-to-face would bring better results as everyone gets involved. Scrum master for facilitating retrospectives could help the situation.
E	After team and projects have been running for years, not so much to improve. Still retrospectives are held, but results are quite light
F	Retrospectives are kept from time to time when someone finds out a problem. In the beginning of the project, when there was a lot of tuning needed in the processes, it was good to have a bi-weekly retrospective meeting.

On table 15 interviewees tell how their teams are using retrospectives. Retrospective is a meeting where team takes a look at the past and reflects how to improve their processes to become more effective and how to improve quality.

In two of the teams where sprint cycles were complied, also retrospectives were held in the end of the sprint. If the project has been running for a long, like several years, there is not so much to improve anymore, in the beginning of the project these meetings have valuable. Also it can be seen that subcontractors do need encouragement to participate into process improvement.

Most of the projects have been running already for a long time and those are used to have this kind of improvement meetings only when a member of the team indicates that things are not going on smoothly.

3.4.4. Problem areas against agile working methods

This chapter provides information about good work practices as product owner or project manager should apply in their work. This information is valid in general for teams which were studied.

- Mixed and distributed team structure

When team is constructed from both local and outsourced resources, it causes many problems against agile principles. Face-to-face meetings are not possible, the tasks are not distributed evenly and a team leader can’t give individual support for remote team members.

- Broad customer base

Having several customer causes continues discussion about priorities between the customers and their requests. If the customers are not very active, it requires additional work effort from project manager to communicate and get the decision about the conflicting priorities.

- Broad product base

When the team several products to deliver it is obvious that priorities between those will change. There is also a risk implementation of specific product turns out to be individual developers responsible. In that situation tasks will not distributed evenly and there will be differences in workload between team members.

- Subcontractors and customer co-operation

Subcontractors may not have access to all possible information needed for decision making. Thus it is mandatory to have a local project manager or technical expert participating to these discussions.

- External pressure directly to team

Sometimes customers try to push their priorities and schedules directly to developers. This can happen even more easily within subcontractor and customer direct communication, due to nature of subcontractor role and cultural aspects. Project manager responsibility is to protect the team from this kind of activity. Agile processes should reveal the work effort not planned to iteration.

- Mixed roles

In agile methods there are usually few specific roles defined. If one person is acting in several roles in team it will cause conflicts or at least too high work load. For example in Scrum the Scrum master can partly act like a designer, but Product owner and Scrum master shouldn’t be the same person. That will be constant turning of the table. Also the setup where developers are working for more than one team is problematic and against agile principles.

- Waste in implementation

Long implementation time of too big tasks and rare demonstrations to customer causes unnecessary implementation included into delivery. Same applies to product backlog priorities validity, project manager duty is to co-operate with customers to keep information updated. The priorities are there to be respected in task selection for implementation. Prototyping can cause unnecessary implementation, when executed in too detail level.

- Offshore subcontractors and self-organization

Self-organization is not subcontractor behavior by definition. They may live in an authoritarian culture and expect that the manager gives the orders and assigns tasks. They may hesitate to make decisions and criticize decisions on the implementation and processes. Situation is far more better from project manager point of view it they have a local team leader. Still it is not sure how self-organizing they are. To change the attitude in the team takes long time and a lot of effort from project manager to encourage them and arrange the needed tools and information to enable self-organization and decision making.

3.4.5. Good work practices to success with agile working methods

This chapter provides information about good work practices project managers or product owners should apply. This information is valid generally for teams which were studied.

- Continuous integration and daily builds

Latest version of the product is always available for customer or other stakeholder review. It may not be fully functioning but it has the latest feature implementation included and feedback can be gathered. It helps to avoid unnecessary implementation and may be used to demonstrate optional solutions for problematic issues.

- Centralized team structure

Development team and its leader or scrum master needs to be located in the site and near to each other. Otherwise interaction and effective communication can't take place. Project manager does not need be located so close to team, but good connection to team is needed, for daily follow-up with team leader or scrum master and for frequent meetings with whole team.

- Support for team success

Project manager is the coach for agile team by encouraging for internal constructive and positive feedback within team. Encouragement for decision making and supporting decision making by providing tools and information needed decision making is responsibility of the project manager in agile development environment. Support is also needed in setting targets and estimating work efforts. Encouragement is required by team for initiating and implementing process improvements.

- Agile boards, physical and virtual

A common whiteboard illustrating team's development current state is very is handy, especially when all team members have easy access to it. It is important to keep it updated at least daily. In addition a common

application for agile board is required to provide status information to customers and other teams which are not co-located. It is important to have agreement how and by whom physical and virtual agile boards are synchronized.

- Test automation

Automated tests within continuous integration gives frequent feedback about the quality of the team current implementation. Usually automated tests are not comprehensive and finally the delivery will be integrated with other software deliveries for further integration. Anyway, it is important to be sure that the latest delivery of team is working seamlessly with current environment.

- Clear roles

Project manager or product owner handles the general information sharing and gathering with customers and other related stakeholders. Product owner's contacts can be utilized whenever deeper co-operation with team and customer is needed.

Development team has the technical excellence and responsibility deliver what they have committed to implement. Customer and project manager should not direct what or how the team will do. They just need to trust the team.

- Product backlog validity

Frequent communication with customers is needed to keep product backlog updated. The team should be able rely on backlog requirements and their priorities.

- Code complexity following

Code complexity measuring tools are easily used within continuous integration environment provided by company, as well as other code quality measurement tools. It is easier to share implementation tasks inside the team when source code is easy to read. Common coding guideline makes code more understandable when applied diligently.

- Continuous improvement

Having a retrospective meeting and discussing process improvement might sound frustrating when the project is mature and it feels like there is not that much to improve anymore. But if the team wants to be agile it needs to be agile also in processes. When a project has moved to a maintenance phase it could be a good idea to think about other agile methods or Kanban process to serve customers better in changed environment.

4 DISCUSSION

4.1. Research results

The first research problem was 'How the agile values and principles are applied in project management'. The interview result analysis chapter 3.4.3 collects the related findings to each agile principle. And those are discussed in detail for every agile principle.

The second question was 'What are the management practices suitable to particular agile software development method'. This was a disappointment for researcher as there was so much variation and freedom in applying a certain agile method that it was not possible to define a method specific guidance for management practices. Presumption was that most of the teams are applying scrum method but the reality was something else. Still there is nothing wrong in tuning the process the team is using, but for this part of the study it just caused too much challenges. Instead of method specific guidance the interview result analysis provided enough data to define success factors and obstacles for executing according to agile principles within the teams under study. Those results are explained in chapters 3.4.4 and 3.4.5 and will be communicated to interviewees.

4.2. Reliability and validity

Reliability means according to Hirsjärvi etc. that when we are studying the same person on two different times we are able to get the same answer or results. The reliability can also mean that two interviewer will end up to same results with the same interviewee. Reliability can also be understood so that with two different research methods can give the results from same target group. (Hirsjärvi etc. 2004, 186)

It is likely not possible to get exactly same answer from the person after some time, as the people change during time and circumstances change and thus affect to people view of the issue. (Hirsjärvi etc. 2004, 186)

It could be possible that the same results would have been reached regardless of interviewer. Researcher itself is not experienced as an interviewer, thus there would have been differences compared to very experienced interviewer.

It is not likely that any other method could give us same results as people behavior depends on the context. (Hirsjärvi etc. 2004, 186)

Reliability can also relate to researcher actions on analysis of the results, like is all data transcript same way and all data littered correctly (Hirsjärvi etc. 2004, 187). In that sense study can be said to be reliable.

It was obvious that after interviewing few people, researcher could not resist to lead the interviewee, when they had difficulties to find their point. It was still kept in control and used only for the cases where interviewee had problems to understand the issue of the discussion.

Validity can be statistical, structural, internal or external. None of these can really be used with qualitative research methods (Hirsjärvi etc. 2004, 187-188). Though it can be said that structural validity of the research method is true but the results are valid only for that group of people at the time research was executed and on their current environment.

Validity in qualitative research can be addressed by member checking. It means that interviewees have a possibility to check and validate the littering and researchers interpretations (Hirsjärvi etc. 2004, 186). In that sense the validity is ensured in this research.

4.3. Reflection on research process

This study was started on December 2012 by planning this research, but the actual work started on September 2013. That long silent period leaded to fact that the whole work needed to start from the beginning. It is really important for this kind research that one can work continuously towards final target. It is important when one is studying the theory, that all related information is documented immediately, otherwise it needs to be studied again. That happened to researcher in theory studies while there was almost nine months break in the process. Analyzing the empirical research result is also a task which requires focusing and concentration. One can't control large amount of data for long time period. If there are breaks in analysis, one needs to digest the whole information again. Researcher learned the lesson from theory studies and focused on interview result analysis intensively.

Interviewing was a new experience for researcher. It can't be avoided that every interview changes the attitude of the interviewer to the issue under discussion. It is also hard to resist helping and guiding the interviewee when one has difficulties to find answers. There is a big possibility to lead the thoughts to a direction that interviewer itself prefers.

The research question about the agile method would have been easier to clarify with just clear questions about the characteristics of the teams' way of working.

4.4. Possible future studies

It was seen in the results that many teams were not able to execute in the most efficient way when those had distribution in the team structure or project layout.

The project managers who were in the target group of this study were not in a position where they can decide the resourcing and distribution of projects and resources. Thus it would be a good idea to study the understanding of agile development principles also within higher management. Same kind of interview as in this study for them could refresh their thinking of agile development and guide to decisions supporting agile development.

Subcontracting and offshoring with agile development could also be one idea for a study. When scoping only to that area the theory of subcontracting and offshoring can studied more thoroughly than in this study and possible better guidance to those situations can be reached.

Scheduling and planning was not studied in this research thoroughly. Anyway there seemed to be quite many different techniques for planning and work effort estimation, for example in the book by Larman. (Larman 2007)

5 CONTRIBUTION

The results of this research are tightly related to the teams that were the target group. This research revealed problems in some teams which were causing challenges to effective execution. Already after interviews before this report is not finished corrective actions were made in some teams. The problems were probably already known but for sure this study gave some extra boost to execute corrective actions. It was also heard from the free comments during interviews that target person were seeing that discussion interesting and good opportunity to stop and think how they have been leading their project.

When these research results are published to target audience it will give them more guidance about what are the things they should maintain or what are the things they could improve. The findings in this research are also good to remember when new projects are established. Especially team structure and project lay-out are the barebones for succeeding. Agile development idea enables adaptation in processes, but it can't change persons and environment. Those fundamentals are important to communicate to instances which make the decisions have power to influence.

The researchers own expertise of leading agile teams has grown a lot during research. It was not a direct target of this research, but unavoidable side-effect. Researcher can now better understand what went wrong in the past trial of applying agile method.

REFERENCES

Agile Manifesto 2001. Ref. 05.11.2013. http://www.agilealliance.org/the-alliance/the-agile-manifesto/

Anderson D. 2010. Kanban: Successful evolutionary change for your technology business. USA WA : Blue HolePress.

Auer A., several other authors. 2013. Ketterää Kehitystä. Suomi: FinnLectura

Cockburn A. 2007. Agile software development : The cooperative game. USA, Boston : Pearson education

Equinox blog 2011, Ref. 13.01.2013, http://blog.equinox.co.nz/blog/Lists/Posts/Post.aspx?ID=155

Boeg J. 2012. Priming kanban. Ref. 9.12.2013. http://oledru.free.fr/agile/kanban/PrimingKanban-JesperBoeg-Version2.pdf

Hackett M. 2013. Top Ten Risks When Leading an Offshore Test Team (Part 1 & 2). Ref. 20.01.2014. http://www.logigear.com/resources/articles-presentations-templates/620-top-ten-risks-when-leading-an-offshore-test-team-part-1.html and http://www.logigear.com/resources/articles-presentations-templates/622-top-ten-risks-when-leading-an-offshore-test-team-part-2.html

Hirsjärvi S., Hurme H. 2004. Tutkimus haastattelu. Teemahaastettelun teoria ja käytäntö. Helsinki: Yliopistopaino.

Haikala I, Märijärvi J. 1997, Ohjelmistotuotanto. Jyväskylä: Gummerrus kirjapaino Oy

International Institute of Business Analysis (IIBA) 2011. Agile Extension to the BABOK Guide. Ref. 1.10.2013. http://austin.iiba.org/download/presentations/The%20Agile%20Extension.pdf

Larman C. 2007. Agile and iterative development: a manager's guide. Westford, Massachusetts, USA: Addison-wesley.

Liker J., Convis G. 2012, Toyotan tapa Lean johtamiseen. Hämeenlinna: Kariston kirjapaino Oy.

McConnell S. 2002. Ohjelmistotuotannon hallinta. Helsinki: Edita.

Niiniluoto I. 1997. Johdatus tieteenfilosofiaan: käsitteen- ja teorianmuodostus. Helsinki: Otava.

Poppendieck M., Poppendieck T. 2003. Lean software development: An agile toolkit. USA : Addison Wesley

Royce W. 1970. Ref. 24.11.2013. http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf

Software Testing Class 2012. Ref 05.11.2013. http://www.softwaretestingclass.com/wp-content/uploads/2012/05/v-model-stlc1.png

Turialspoint 2013. Ref 24.11.2013. http://www.tutorialspoint.com/management_concepts/agile_project_management.htm

Kainulainen A. 2008. Agile Menetelmät. Jyväskylän Ammattikorkeakoulu

Koivusalo M. 2012. Measuring progress of agile transition. , University of Turku: Department of information technology

Yeatts D . 1998. High-performing self-managed work teams: A comparison of theory to practice. USA: Sage Publications.

Agile method properties (Kainulainen, 2008)

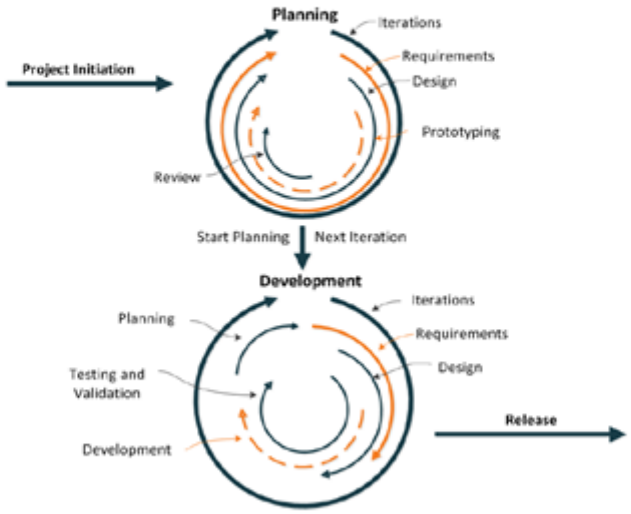
	Scrum	XP	ASD	Crystal	FDD	DSDM
Suunnitteluvaiheen laajuus	Osin määritelty, lyhytkestoinen, mutta riittävä	Osin määritelty, mahdollisimman lyhytkestoinen, mutta riittävä	Tarkoin määritelty, kattava ja pitkäkestoinen	Ei määritelty	Tarkoin määritelty, melko kattava, melko pitkäkestoinen	Tarkoin määritelty, kattava ja melko pitkäkestoinen
Iteratiivisuus	Kevyt suunnittelu, toteutusvaihe iteratiivinen	Kevyt suunnittelu, toteutusvaihe iteratiivinen	Jaksottainen suunnittelu-vaihe, toteutusvaihe iteratiivinen	Heikosti määritelty, ainakin toteutus-vaihe iteratiivinen	Jaksottainen suunnittelu-vaihe, toteutusvaihe iteratiivinen	Jaksottainen suunnittelu-vaihe, toteutusvaihe iteratiivinen
Iteraation pituus	30 päivää	2-4 viikkoa	4-10 viikkoa	2-4 kuukautta	2 viikkoa	ei määritelty
Asiakkaan osallistuminen	Suunnittelu, Iteraatioiden suunnittelu, tulokset	Päivittäinen, tiimin aktiivinen jäsen	Suunnittelu, iteraatioiden tulosten katselmointi	Ei tarkoin määritelty, mutta aktiivinen	Suunnittelu, iteraatioiden osalta ei määritelty	Suunnittelu, iteraatioiden tulosten katselmointi
Asiakkaalta vaadittu tekninen osaaminen	Ei erityistä vaatimusta	Korkea	Ei erityistä vaatimusta	Ei määritelty	Ei määritelty	Ei erityistä vaatimusta
Prosessin määrittely	Vaiheet tarkasti määritelty, työskentelytavat vapaat	Vaiheet ylimalkaisesti määritelty, vaihetuotteet ja työskentelytavat määritelty yksityiskohtaisesti	Vaiheet määritelty, vapaat työskentely-tavat	Ei määritelty	Vaiheet ja vaihetuotteet yksityiskohtai-sesti määritelty, työskentelytavat vapaat	Vaiheet ja vaihetuotteet yksityiskohta-isesti määritelty, työskentelytavat vapaat
Vaihetuotteiden määrä	4	6	5	Ei määritelty	5	7
Soveltuva henkilömäärä	Ei rajoituksia, skaalautuvuus hoidettu	Pienet, alle 10 hengen projektit	Ei rajoituksia, skaalautuu hyvin	Eri prosessit erikokoisille projekteille	Ei rajoituksia, suunniteltu suuremmille projekteille	Ei rajoituksia, käytetään yleensä suuremmissa projekteissa

KUVIO 14 *Agile-menetelmien ominaisuuksia*

Interview support material

1

Project life cycle



2

Agile 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

Interview support material

3

Agile principles

- 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.

4

Lean software development

	Lean Principle	Description
D	Deliver Fast	**Deliver value to the customer quickly**, rapid delivery; high quality; low cost **Queuing theory** to Limit Work in Process (WIP) and context switching **Manage workflow is easier than managing schedules,** using repeatable workflow
E	Eliminate Waste	Waste is anything that does not add value to the customer. The three biggest wastes in software development are: 1. **Building the wrong thing**: building features that aren't needed 2. **Failure to learn:** policies that interfere with our ability to learn 3. **Thrashing**: anything that interferes with smooth flow of value
L	Learn Constantly	**Predictable performance is driven by feedback:** rapidly respond to change **Maintain options**; keep code change tolerant, minimise irreversible decisions Defer commitment, schedule irreversible decisions to **Last responsible moment**
I	Build Quality In (Integrate Quality)	**Final Verification should not find defects!** Prevent with executable requirements **Mistake proof your process with test first development** to establish correctness **Break dependencies:** architecture should support addition of any feature at any time
V	Optimize the Whole (Value the Whole)	**Focus on the entire value stream** from customer request to deployed software **Deliver a complete product**, a complete team delivering not just the software **Think long term** rather than local optimization
E	Engage Everyone	**Autonomy**: Empowered self-organising feature teams with effective leadership **Mastery**: Provide challenge and environment which enables people to grow **Purpose**: Tie the work to value and a common vision
R	Keep Getting Better (Relentless Improvement)	**Failure is a learning opportunity**: investigate and correct them as they occur **Standards exist to be challenged and improved** **Use the scientific method** Plan-Do-Check-Act process

Interview support material

5

Management

- Self-managed teams
  - Support for decision making
  - encouragement for setting goals
  - Assistance
  - Coach
- Management skills development

6

Outsourcing/offshoring

- Work measuring, confidence to effort of other company workers
- Visibility to daily work
- Contact point
- Plans for downtime
- Access to our network and services
- Can you trust your partner?
- Attrition
- Culture/personality/communication
- Time for communication
- Language barrier