Interaction Designer in Agile Process

TAMK University of Applied Sciences
School of Art and Media
Interaction Design
Fall 2008 – Fall 2009
Vappu Leinonen

Acknowledgements

My thanks goes to:

- My colleagues at Nokia - some of you gave me the inspiration to write this thesis, some encouraged me and some made me laugh, even when I was tired.
- All my teachers - from the very first ones, Veikko and Anne, to the latest ones, Ari Närhi and Juha Ollikainen - you all have given me the inspiration to always learn more.
- My friends - you gave me something else to think about. Especially Krista - this wouldn't have been possible without your help and support.
- My family - the larger one, including relatives, because I wasn't raised only in the smaller one - knowing that you are there is enough and still you are giving me so much more.
- And last but definitely not least: Sami - thanks for all the love and support.

Opinnäytteen tiivistelmä

Vappu Leinonen
Interaction designer in Agile Process
Lokakuu 2009

59 sivua + liitteet
Tampereen ammattikorkeakoulu
Viestinnän koulutusohjelma
Vuorovaikutteisuuden suunnittelu
Lopputyön muoto: kirjallinen
Lopputyön ohjaaja: Ari Närhi
Avainsanat: ohjelmistotuotanto, agile, prosessimalli, vuorovaikutteisuuden suunnittelu

Ketterät metodit (agile) ovat nopeasti valtaamassa perinteisen vesiputousmallin paikan ohjelmistokehityksen alalla. On tiedetty jo pitkään ettei vesiputousmalli toimi muuttuvissa tilanteissa jotka ovat ominaisia ohjelmistokehitykselle. Ohjelmistokehittäjät loivat agilen vastaukseksi ja he ovat todistaneet että se todella toimii. Mutta agilea luodessaan he unohtivat yhden asian: suunnittelun.

Vuorovaikutteisuuden suunnittelu on tärkeä osa ohjelmistokehitystä, mutta agilen kuvailuissa ei edes ehdoteta mitään siihen liittyvää. Vaikka on selvää että ohjelmistokehittäjät loivat agilen toisille ohjelmistokehittäjille, on yhdistetyn prosessin tarve kasvamassa. Vuorovaikutteisuuden suunnittelun ja agilen yhdistelmä ei ole niin yksinkertainen kuin aluksi voisi olettaa. Molempien täytyy antaa periksi joidenkin periaatteiden osalta jotta voitaisiin saada aikaan jotain enemmän: parempi tuote loppukäyttäjälle.

Tämän tutkimuksen teoriaosa kuvaa vesiputousmallin ongelmia, sitä miksi agile luotiin ja mitä se on, mitä vuorovaikutteisuuden suunnittelu on ja miten vuorovaikutteisuuden suunnittelu voitaisiin yhdistää agileen. Yhdistelmästä ei ole olemassa monia kuvauksia vaikka juuri kuvaukset olisivat tärkeitä heille, jotka aikovat yhdistää vuorovaikutteisuuden suunnittelun agileen. Tähän tarpeeseen vastaa tutkimuksen loppuosa, jossa kerrotaan prosessimallista joka on käytössä yhdessä Nokian Devices-puolen tutkimus- ja kehitysprojektissa. Loppuosa sisältää myös tulokset kyselystä, joka tehtiin jotta saataisiin selville mitkä osat prosessimallista eivät toimi.

Thesis summary

Vappu Leinonen
Interaction designer in Agile Process
October 2009

59 pages + appendices
TAMK University of Applied Sciences
Media Programme
Area of specialisation: Interactivity Design
Type of Final Project: Written
Thesis supervisor: Ari Närhi
Keywords: software development, agile, process model, interaction design

Abstract:

Agile methods are quickly replacing the traditional waterfall process model in the software development industry. It has been a widely known fact for a long time that the waterfall does not work in changing situations, which are the essence of software development. Developers created the agile methods as an answer and they have proven that it works. But when they created agile methods, they forgot one thing: the design.

Interaction design is an important part of creating the software, but agile does not even suggest anything related to that in the process descriptions. While it is clear that agile was created by developers to developers, the need for combined process is growing. The combination of agile and interaction design is not so simple as it may sound to be. Both have to give up on some principles, in order to gain something more: a better product for the end user.

This thesis has a theory part which describes the problems of waterfall, why agile was created and what it is, what interaction design is and how interaction design could be combined into agile. Only few descriptions about the combinations of interaction design and agile exists. The descriptions would be important for those who are trying to figure out what kind of things they should take into account when starting to combine interaction design into agile. The case study part of this thesis answers to the need by giving a description of the process model, which is used by one project in Nokia Devices R&D. The project is called project A in this thesis. In addition, the case study includes the results of an Internet survey, which was conducted to find out which parts of the process were not successful.

Table of Contents

1 Introduction

Interaction designers have been one of the few groups in the software development industry who have had a pleasure to work without any specific knowledge of the development process. They have traditionally worked in the design phase of the waterfall process, having a specific timeslot for all work - including concepting and specifying. The concepting is almost always the easy part, but the specifying has been a problem. Interaction designers are frustrated with the amount of details which have to be specified, knowing that if they forget something, or think something is common knowledge, the final product won't be what they intended it to be. Some designers believe that if they some day produce a perfect specification, the final product will at least somehow reflect that. Others have already learnt the lesson: it won't. Specification is either not read through and understood by developers - or more commonly, some problems arise during the development and changes are needed.

Both - the interaction designers and the developers - have been discussing the possibility to work together. Somewhere they already are doing that in an unstructured way. In small projects that might work - just forget all processes and do what you want - and especially: how you want. But for bigger projects, working without any process causes huge issues. If the problems of the waterfall process model are reflected on the final product, what could be the answer? The development has raised the issue up and done something about it. They created agile methods as an answer to the need of change. They even have proofs that agile works; there are many books, studies and articles written about the subject and all those say the same: by using agile methods, the development is done faster, people are happier and changes are possible even in the middle of the project. However, one big thing is missing: they forgot the design.

In most of the projects, which use agile methods, developers do the design. That can also work in some cases, but most of the projects would definitely gain advantage of the proper design - done by interaction designers, graphic designers and usability specialist working together. The most beneficial combination would

combine all these professions into the same process with developers. However, agile methods do not take advantage of these professions; they seem to completely forget them. Of course, the agile methods were originally created by developers to developers, but now when they are spreading to the whole software development industry, the design should be also considered. This study gives an insight to the current reality - why agile methods were created and what they are, what interaction design is and how it could be combined into agile process. And even more importantly for interaction designers: could the agile methods be a solution to the problem where the final product does not reflect the original interaction design?

1.1 Research questions and objectives

This study aims to understand the effect of agile on interaction designers' daily work. In addition to that, the study explains some of the possible ways of combining interaction design into agile.

The study explores following questions and their possible answers:

- Why was agile created and what it is?
- What is different in the agile process compared to the traditional waterfall process?
- What are the common things in interaction design and agile? What are the differences?
- What kinds of processes have been used to create agile interaction design? What changes are the needed to the agile process and to interaction design?

The thesis consists of two parts: theory and case study. The theory part is written based on literature and online references. It explains the basics of agile and interaction design, compares them and also describes some projects which have combined interaction design into agile. The case study explores a Nokia project where interaction designers are part of the agile process. The case study tries to find answers to following questions:

- What kind of process is used in this project?

- Which areas of the design work are not successful in this process? Which areas of the design work should be improved?

The case study is not intended to give generalized answers; it only gives insight to one Nokia project. The results will be used only by that project for improving process areas, which don't seem to go well yet. The case study results - even though they can't be generalized - are still interesting to many others because there aren't that many studies done around the particular subject.

1.2 Limitations

This study can't cover all possible aspects of the researched area and all things related to that. The thesis focuses on interaction design and even though there are some references to graphic design and usability, those professions are so different from interaction design that they should be studied separately. The thesis includes only brief overviews of software development, the waterfall process model, agile and interaction design. There is a lot of information available about those subjects in other sources.

1.3 Main concepts and definitions

Agile

Agile is a word for combining all agile methods under the same title. Those methods can differ really much from each other, even though they are essentially based on the common ground. The common ground defines agile methods to be iterative and incremental process models. Agile is described in more detail in chapter 2.2.

Interaction design

Interaction design is a discipline, which mainly focuses on defining interactive products and services - especially how those products and services function and are structured. Interaction design is partly about usability, information architecture, graphic design, industrial design and other disciplines, but it has also

unique characteristics mixed into the combination. Interaction design is described in more detail in chapter 2.3.

Software development

Software development means all activities, which have to be done to create software products. Those activities include e.g. designing, implementing, marketing and maintaining the software product. Software development can also refer to phases after the design phase in waterfall process model - in those cases the design and development are separated. When creating software products, some kind of process model is used; traditionally the most preferred one has been the waterfall process model, but lately agile methods have gained much popularity.

Waterfall

Waterfall is a sequential process model, which has been used widely in the software development industry. Waterfall divides the development process into seven phases: requirements, design, implementation, integration, testing, installation and maintenance. The commonly known fact in software development industry is that the waterfall does not work; it causes projects to run out of time and budget. Waterfall is described in more detail in chapter 2.1.

2 Theory

Way back in time, the first automatic electronic computers were simple. Software was always designed for one computer and for one task. Software was a new thing and developed by the people who knew how to program. There was no process for creating software, because there was no need for that. Software was so small scale at first that one developer or a small team of developers could create the whole software without any special process. (Dijkstra 1972)

As computers' size got smaller, amount of memory bigger, performance faster and price cheaper, software became more complex. More computers were sold and reusable software was needed. Soon it became hard to manage a software project without a structured process. Software crisis came up when projects ran over budget and schedule; software was of low quality and did not meet the requirements. It was soon clear to everyone that some structured process was needed and the first software development processes were created. (Wikipedia: History of software engineering)

2.1 Waterfall process and why it doesn't work

One of the most used processes has been the waterfall model. The first time it was introduced - but not named - by Winston W. Royce in 1970. Royce explained how software projects could be managed. Ironically, he also stated that this kind of process model is risky and explained his opinion that it should be used only in very small projects. However, almost all of the large-scale software projects from 70's until this day have used the process model, which Royce described. At some point it was given the name 'waterfall'. (Wikipedia: Waterfall model)

2.1.1 Waterfall process model

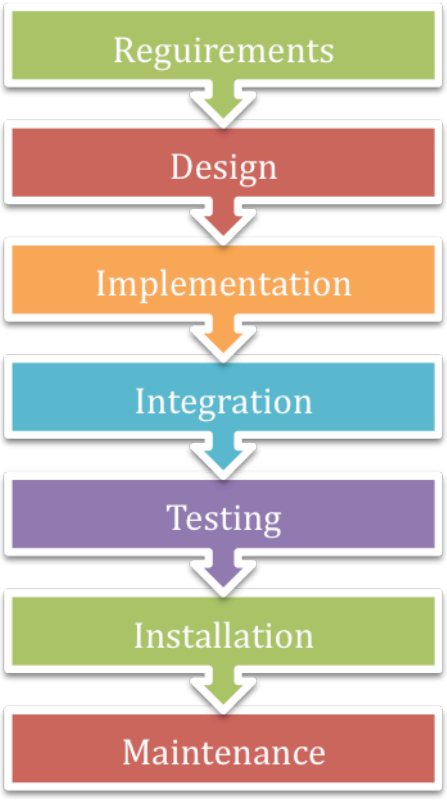

Figure 1: Waterfall process model (Wikipedia: Waterfall model)

Waterfall is a sequential process model, which includes seven phases: requirements, design, implementation, integration, testing, installation and maintenance. In waterfall, one can proceed to the next phase only when all the phases before that are finalized. The finalized phase is never touched again, e.g. design is frozen when entering to the implementation phase. (Wikipedia: Waterfall model)

Requirements is the phase where customer needs are gathered. Typically in this phase the most important question is: what is the problem that the software is expected to solve. During this phase, requirements specification is written. In *design* phase the requirements specification works as a starting point. Designing means defining architecture, interfaces, interactions, data, etc. Outcome of this phase are design specifications. *Implementation* phase means the actual

constructing of the software. Practically this means that programmers code the different parts of the software as specified in the design phase. In the *integration* phase all the different parts of the software are put together. The next phase is the *testing* of software; finding the bugs and fixing them. (Wikipedia: Software development process)

After the software is tested and all the bugs found have been fixed, it is time for *installation*. Installation includes all the necessary actions that are needed to make the software ready for the customers. After these actions, the software is ready and delivered. Of course, the work does not end on the delivery. Software always needs new versions because of customer requests or defects found while the system is in use. New fixed versions of the software are created during the *maintenance* phase. (Wikipedia: Software development process)

2.1.2 Why waterfall has been so popular?

Waterfall has worked in some situations, which have mainly been unchanging. The unchanging situation is ideal for the waterfall method because with fixed environment it is easy to predict the future and plan accordingly. The main idea behind the waterfall model is simple: time spent early on making sure that requirements and design are absolutely correct will save much time and effort later. The waterfall model is also called 'Big Design Up Front' (BDUF), because the main idea relies heavily on doing the design up front. (Wikipedia: Waterfall model)

2.1.3 Reality check with waterfall

Starting to use waterfall as a software development process was originally based on misunderstanding. Even Royce, who was the first one to describe waterfall process, though it is too simple to succeed. He was absolutely right.

Stanley J. Jarzombek's study has been widely used as an example of waterfall's failure. Jarzombek studied the software projects of USA Department of Defence (DoD) in 1995 to find out how many projects were successful. All studied projects used waterfall process, because they needed to follow DoD's standard 2167A. Results of Jarzombek's study (1999) show clearly that waterfall does not work:

29% of the software was not delivered, even though it was paid for, 46% was delivered but not successfully used, 20% had to be extensively reworked or was abandoned after delivery and 3% was used after doing some changes after delivery. This all leads to the fact that only 2% of the software was used as delivered. The study just nailed the commonly known fact: waterfall is not the right process for software.

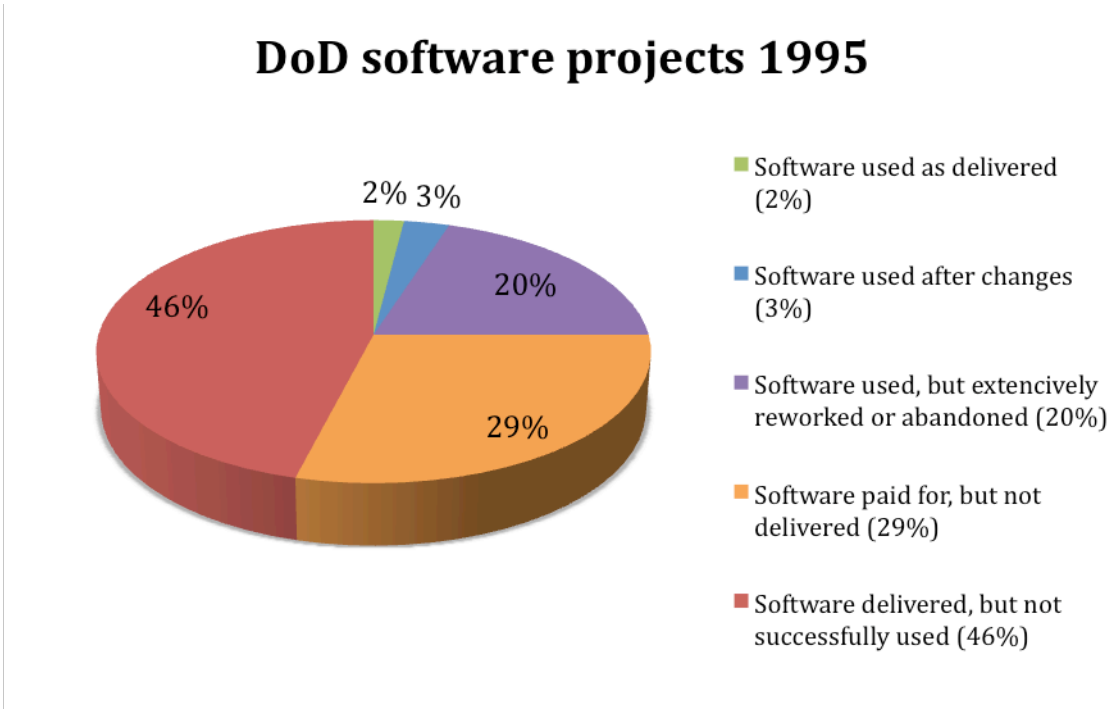

Figure 2: DoD software projects 1995 (Jarzombek 1999)

What are the reasons behind the big failure, which has caused doomed projects and wasting billions of money? Dean Leffingwell (2007, 20) has written a good list of assumptions, which are the essence of waterfall:

1. *There exists a reasonably well-defined set of requirements if we only take the time to understand them.*
2. *During the development process, changes to requirements will be small enough that we can manage them without substantially rethinking or revising our plans.*

3. *System integration is an appropriate and necessary process, and we can reasonably predict how it will go based upon architecture and planning.*

4. *Software innovation and the research and development that is required to create a significant new software application can be done on a predictable schedule.*

Leffingwell also writes about the reasons why these assumptions are incorrect. The first assumption is not true, because development team can't describe intangible requirements to customers in such a way that the customers would have the same vision as the team. Therefore the team will deliver software, which is not exactly what the customer had in mind. Also the delivery will possibly change the requirements.

The second assumption is even more incorrect. Leffingwell has no calibrated or absolute data to prove his experience, but he addresses a well-known fact: "-- *requirements do change while the system is being developed*." He also states that the combination of slow development and fast change create problems in the project.

Leffingwell also trashes the assumption number three by lesson learned: "-- *even a relatively thorough up-front analysis could not predict nor control the system integration process. The problem is too complex; change happens midstream; technologies evolve during the project and integration assumptions are often wrong and are simply discovered too late*."

As the first three assumptions are not true, how could the fourth one be? Maybe because everyone is predicting that some change will happen? Of course everyone has also left room to the schedule to make necessary changes. But even that hasn't been enough: making software is simply not predictable. (Leffingwell 2007, 21-26)

It has been widely known fact inside the software industry that waterfall does not work in practice. Most of the software projects have not used pure waterfall model, but some kind of modification of the waterfall. Perhaps the most known modification has been the sashimi model, which was originated by Peter DeGrace. In the sashimi model, different phases of the waterfall overlap. By overlapping, the

problems that are discovered in the next phase of the project can still be fixed in the previous phase because of the overlapping. However, even modified versions are heavy weighted, slow and unable to cope with big changes. (Wikipedia: Waterfall model)

2.1.4 Summary

The history clearly shows that some kind of structured process model is needed when developing software. The first and most used process model has been the waterfall model. In most cases waterfall hasn't been working as it was intended. The main problem with waterfall has been it's inability to cope with changes. Most of the software projects change a lot during the project's lifecycle and using the waterfall process model causes big issues in those cases. In some cases this problem can be handled with modified waterfall process, e.g. with sashimi model, where different phases of waterfall overlap. But big changes still remain as an issue.

As a lesson learned from history, the waterfall method shouldn't be used as a process model for software development if it seems that there is a possibility of changes happening during the project's lifecycle.

2.2 Agile, a way of coping with changes

When the waterfall model had failed several times, software industry needed something else to keep the projects on track. Lightweight methods started to emerge from multiple places at the same time in mid-90s. On February 2001, seventeen key people of lightweight methods met at Snowbird, Utah to discuss about the similarities of their methods. During the few days the group adopted the name 'Agile Alliance'. They also created the *Agile Manifesto*, where basic principles of agile are described:

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

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

That is, while there is value in the items on the right, we value the items on the left more. (Agilemanifesto.org 2001)

2.2.1 Agile methods

Alistair Cockburn (2007, 369-372) has explained a little more about the manifesto. He tells that lightweight methods – now known as agile methods – were not so much invented, but more like 'uncovered' through the real work in software development. Agile Manifesto has been constructed to show the problems of heavyweight methods. Last sentence of the manifesto is really important: agile is not throwing away all the items on the right; agile just appreciates the items on the left more.

For all agile methods, the four statements of Agile Manifesto are very much true. Some of the methods emphasize one or two items more, but all methods are based on the common ground. Scott W. Ambler (2008) has written that common ground to a simple form:

> *Agile is an iterative and incremental (evolutionary) approach to software development / which is performed in a highly collaborative manner / by self-organizing teams / with "just enough" ceremony / that produces high quality software / in a cost effective and timely manner / which meets the changing needs of its stakeholders.*

The simple common ground leaves two fundamental aspects of agile unmentioned: agile is also reactive and adaptive. Agile reacts to the changes and adapts to the changing circumstances. (Wikipedia: Agile software development)

Even with the common ground, agile methods are very much different from each other. Agile is not a method, one simple solution to a very complicated and changing problem. Agile is just a common ground for the methods under the title 'Agile methods'. Dean Leffingwell (2007, 8) lists those methods to include Dynamic Systems Development Method (DSDM), Adaptive Software Development (ASD), Crystal Clear, Scrum, eXtreme Programming (XP), Lean Software Development (Lean), Feature-Driven Development (FDD), Agile Unified Process (AUP).

Leffingwell's list - or any other list - is not complete. But complete list doesn't even make sense in the reality where everyone is usually adopting one method and then changing it to suit them better, either by taking some practises from existing methods or by creating their own practises. Not many projects out there are practising one pure agile method. Leffingwell (2007, 8) states that the most used methods are Scrum, XP and DSDM. DSDM is widely used in USA, but Scrum and XP are definitely the most used ones in Europe.

2.2.2 Scrum

Jeff Sutherland and Ken Schwaber developed scrum in mid-90s. Since then it has been widely used in many software companies and IT organizations. Scrum is based on few simple rules which help to produce incremental releases of a product in fixed period of time, prioritize the work, keep the work transparent and at the same time keep the team in control of their work. (Leffingwell 2007, 41-49)

The scrum identifies three roles: product owner, scrum master and team member. The product owner is representing the interests of the user and customer. He is doing that by managing the product backlog. The scrum master has two responsibilities: implementing the scrum practices in the team and helping the team to achieve the goals. The latter also includes the responsibility of removing any obstacles that are impeding the team. The team member is responsible for implementing the functionality of the product. The team is organizing and managing itself and ideally should be able to be cross functional, so that anyone can do whatever tasks are in the backlog. The ideal team size is seven, plus or minus two. If the team is bigger, it should be divided into separate teams. If the

person is not a product owner, a scrum master or a team member, he shouldn't be interfering during the sprint. (Schwaber and Beedle 2001)

The scrum project starts by the creation of the product backlog. Product backlog has all the features that have to be done before the product is ready. The features are arranged as items, which include the description and estimation. The product owner is responsible of the product backlog, especially prioritizing the items. The product backlog is the most important document in the scrum process, but it is constantly changing during the project because increasing knowledge of the project will affect the product backlog. (Schwaber and Beedle 2001)

After the product backlog is created and teams are set, it is time for the first sprint planning meeting. During each sprint the team creates an increment of usable software. The sprint is a fixed period of time, traditionally 30 days, but nowadays anything between 1-4 weeks. In the planning meeting, team members calculate how much they can do in this sprint as a team and then select enough - but not too much - items from the product backlog to be done in this sprint. Items with high prioritization are of course taken first. Selected items are moved to the sprint backlog and expanded by the team. Expanding means that the team members are dividing each item into tasks, which are so small that the team can describe them clearly enough to every team member to know what should be done to complete the task. The team also estimates the hours, which are needed to complete the task. It is important to notice that the team decides what will be done during the sprint, because the team has to commit on doing the sprint backlog items. (Leffingwell 2007, 41-49)

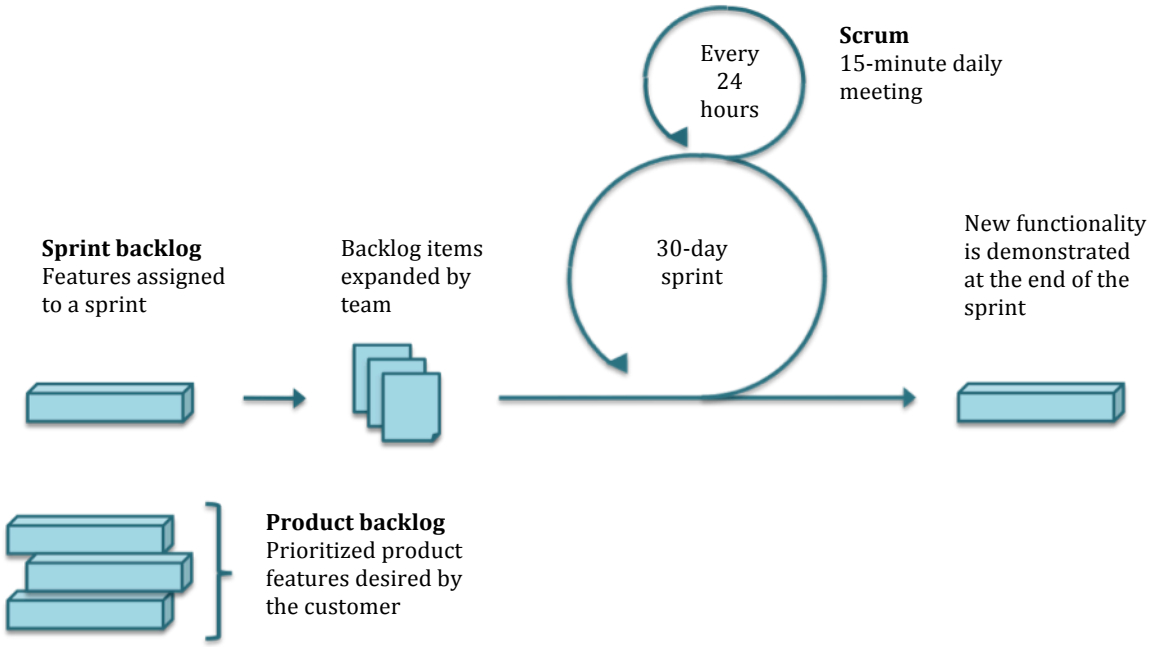

Figure 3: Scrum process model overview (Leffingwell 2007, 47)

After the planning meeting the real work starts. Every team member takes a task from the sprint backlog and starts working on that. Ideally remaining task hours are updated constantly, but in practice it is enough if the hours are updated before each scrum meeting. Scrum meeting is maximum 15-minute daily meeting where every team member answers the same questions:

1. What did you do since last scrum meeting?
2. Do you have any obstacles?
3. What will you do before next meeting?

The purpose of the daily meeting is to keep the work transparent to all other team members. The obstacle question is important because any obstacles should be removed as soon as possible to keep the work ongoing. The scrum master is responsible of removing any obstacle that occurs. The remaining hours are observed by watching the sprint burn down chart (Figure 4) in every scrum meeting. If it starts to seem that the team can't finish all tasks in the sprint backlog, the team can decide to drop some of the items or tasks back to the product backlog.

Those items or tasks are then taken into the next sprint backlog. (Leffingwell 2007, 41-49)

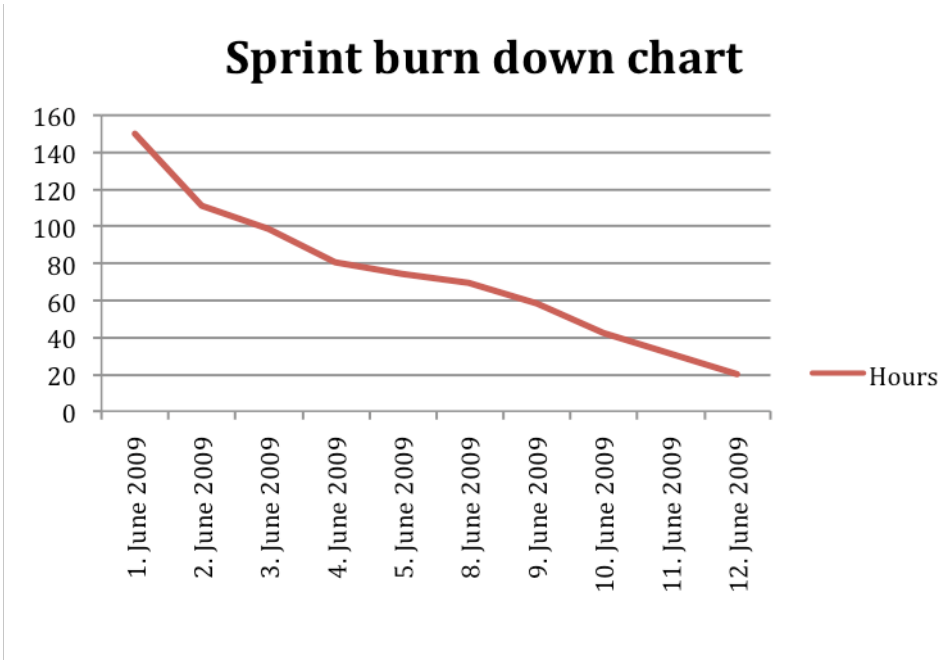

Figure 4: Example of the sprint burn down chart

The work continues like described above until the sprint ends. Each sprint ends with the sprint review where the new functionalities are demonstrated to the product owner and stakeholders. After the sprint review it is time for the next sprint planning meeting, sprint work, etc. Sprints continue until the stakeholders decide that the product is ready enough or until the time for that specific project is used. (Leffingwell 2007, 41-49)

The scrum is easy to adopt and it has also been successfully used in large organizations. It has clear advantages, like being transparent and easy to understand, even though it is flexible. The most important advantage of scrum is that the team is always committed to do the tasks and they always know what they should do.

2.2.3 XP

Extreme programming (XP) was developed by Kent Beck, who published his book *Extreme Programming Explained* in 1999. The word *programming* in the name

clearly defines what is the important driver in XP: providing explicit methods for programmers, so they can more confidently respond to changing requirements and still produce quality code. XP has many principles and practices that are like jigsaw puzzle pieces; individually they make no sense, but when combined together a complete picture can be seen. (Larman 2004, 137-171)

The XP team consists of five to ten programmers who work at one location with the customer on site. The key roles include the following: customer, programmer, tester, coach and tracker. The customer is responsible of writing the user stories and acceptance tests. She is also needed on site to answer the questions as soon as they arise. Programmers and testers write the tests, designs and code. They also identify tasks and estimate those. Tester also helps the customer to write and develop tests. The coach customizes the process as needed and teaches the process to others. The tracker collects the metrics, tells the progress and gives feedback on poor estimates. (Larman 2004, 137-171)

In XP it is more important to focus on key practises than to process lifecycle. Leffingwell (2007, 35-38) has listed 13 practices to be the essence of the XP:

1. **Sit together** means just what it says: the whole team should sit together in common workspace. This helps to build the sense of community and also drives the informal communication.
2. **Whole team** works only for one project at the time and an individual team member can't be part of more than one team at the time.

3. **Informative workspace** helps to keep everyone on track what the team is doing and how the progress is going. Easiest way to handle this is to use the common workspace. Usually there are user stories in the right spots on the walls so that everyone can see what is currently worked on, what has been done and what will be done in the next iteration.

4. **Energized work** means that a team member shouldn't be working more than 40 hours a week. Working in XP is intense and require that people work closely together. Everyone is tired after eight hours of intense work and they also need some privacy after being together for those eight hours.

5. **Pair programming** is done by two people with one computer. Those two people are collaborating, discussing, coding and testing the same piece of code at the same time. This working method is increases the quality of code and is one of the most interesting practices of XP.

6. **User story** is the unit of functionality in XP. User stories are written by the customer as things that the system needs to do. User stories should only provide enough detail to make an estimate of how long the story will take to implement. When it is time to implement the story the customer will give a detailed description of the requirements.

7. **Weekly cycle** is the iteration cycle in XP. User stories should be so small that some number of them can be finished in one iteration. With weekly cycle the progress can be seen weekly and also accomplishments can be appreciated weekly.

8. **Quarterly cycle** is the most natural fit for larger amounts of work, such as releases or major external deliverables.

9. **Slack** helps to keep the humanity in the system. The slack can be arranged by many ways, but the most important thing is that it is arranged somehow. Otherwise the constant highly productive work will get the team frustrated.

10. **Ten-minute build** is one of the goals in XP. The team should be able to automatically build the system and run all the tests in 10 minutes.

11. **Continuous integration** is achieved by integrating and testing all changes every few hours or in the worst-case scenario, daily.

12. **Test-first programming** means that a failing automation test is written before the code writing. This is done because it improves the quality of the code.

13. **Incremental design** is also part of XP. The team design a little every day and when they discover extra complexity, they refactor the code and the

tests. This means that architecture emerges over time as the team members discover it.

Figure 5: XP Lifecycle (Larman 2004, 142)

Graig Larman (2004, 137-171) has defined the lifecycle of the XP process. The XP process starts with exploration where enough user stories for the first iteration are written on cards and they are given the estimates. Feasibility should also be checked in the exploration phase. In the planning phase the date and the stories for the first release are agreed. The customer and developers are also completing the needed story cards and estimates. In the next phase are the actual iterations. Iteration planning game is held when starting iteration. The customer selects the stories to be implemented and developers break those stories into short, estimated tasks. After the tasks are estimated the selected stories might change because the team can't do all selected stories in one iteration. After the stories, tasks and estimates are done, developers start the implementation. When needed, they also collaborate with the customer. If the product is not finished for the release in the end of iteration, the team starts again from the iteration planning game.

2.2.4 Summary

Lightweight methods started to emerge from different places in mid-90's. They were created because the waterfall process model had been proved to be unsuitable for the software development. In 2001 the key people of lightweight methods met and adopted the name 'Agile Alliance'. They also wrote the *Agile Manifesto*, which gathers the basic principles of different methods. After the meeting, lightweight methods have been grouped together under the title agile. The basic principle of all agile methods is that software is in iterative fashion and in each iteration an increment of software is finished.

Different agile methods remained separate and also new methods were formed. The most used methods today are scrum and extreme programming (XP). Scrum is a simple frame for keeping the work focused and transparent. It has been used in large projects, also for other kind of projects than strictly just development. XP is more like a set of practices and is more focused on making the development work easier. Although a project can adopt one of the agile methods, in most of the projects, the decision has been to create own process by using the most suitable parts of existing methods.

2.3 Interaction design

Dan Saffer (2007, 9-18) has defined a brief history of interaction design in his book *Designing for interaction*. He clarifies the history by dividing it into two parts: before and after computers. Before computers, machines were responding to human input, but not in a sophisticated way. Machines did not have any awareness of that they were used. However, inventing the machines such as telegraph, telephone, radio and television required the creators to design an entire system of use. This design work is actually the starting point of interaction design.

When the first computers were invented, the creators were still focusing on the machines, the hardware. Humans still had to adapt to using them. The focus started to change towards people when new inputs such as control panels were created in 1960s. At the same time the first experiments with monitors, mouse and

light pen were starting. The focus shifted even more towards people when software became more important than hardware. In 1970s even people who weren't especially taught to use computers could use the command line interface and new software. Apple introduced the first graphical user interface (GUI) in early 1980s to the mass audience. At the same time, dial-up modems introduced new possibilities such as bulleting board system (BBS) where users could leave messages to others and read the messages. This invention started the human-to-human interaction through computer interfaces. (Saffer 2007, 9-18)

In early 1990s the interaction design started as a formal discipline when the World Wide Web (WWW) and mass adoption of e-mail brought the need for better design forefront. 1990s was also the invention time for merging technologies, for the first time computer technology was used in cars, stereos, dishwashers etc. Other new inventions were made in the world of entertainment and mobile phones. Someone also needed to design the interaction with those merging technologies and new inventions. That increased the importance of newly born interaction design quickly. (Saffer 2007, 9-18)

2.3.1 Definition of interaction design

Interaction design was not the only discipline trying to answer huge design needs. Other disciplines trying to fulfill the need included e.g. user-interface engineering, usability engineering and information architecture. It took some time to get all the disciplines to a state that they could be described. Meanwhile the Internet matured and became more about doing things than just reading content. Technologies were evolving so quickly that everyone was just using the term, which they had heard from somewhere or just made up one when they needed it. No one had time to carefully think through who should design what and how. This quick evolution confused and still confuses many of those who were or still are somehow involved with the interaction design work.

Today there are few explanations of how all those disciplines relate to each other. Almost everyone agrees that user experience (UX) design is the big umbrella, which includes almost all other disciplines. Describing the overlaps is not easy,

even Dan Saffer who published his figure originally in 2007 has redrawn it and still admits that it doesn't cover all disciplines involved. Saffer's new figure (2008) - although not perfect - contains some valuable information. It shows that interaction design overlaps with many other disciplines. Saffer has even named the overlapping parts in the new figure.

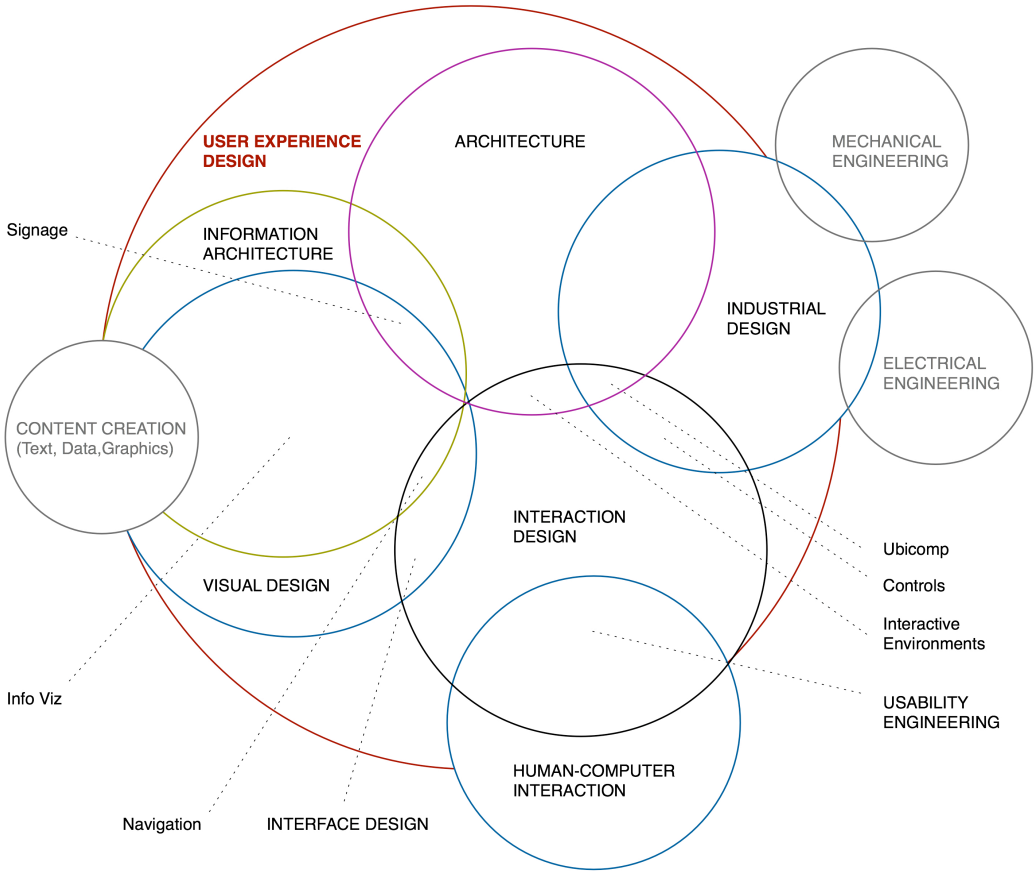

Figure 6: The Disciplines of User Experience (Saffer 2008)

By looking at the Saffer's figure, interaction design seems to include interface design, navigation, interactive environments, controls and usability engineering. But that is not all. How should the blank area be filled? What definition could describe that?

Jesse James Garrett (2003) starts by describing the whole user experience in web environment. First he defines the five planes of user experience (Figure 7). Garrett states that design process is about moving from top to bottom - from abstract to

concrete - in the plane figure. He splits his planes down the middle because in web environment things are easily split into two different objectives: web as a software interface and web as a hypertext system. Software side is mainly about the tasks and the hypertext side is about the information. After splitting the figure shows how different planes break down into component elements (Figure 8).

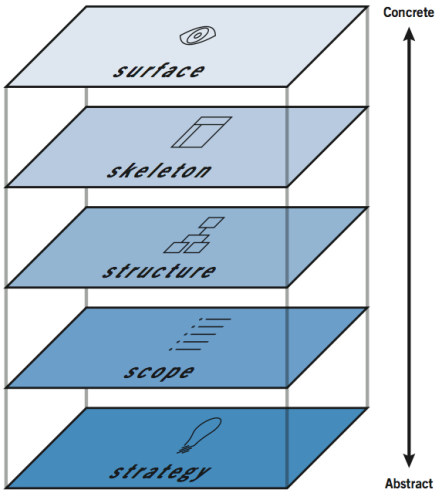

Figure 7: Planes of user experience (Garrett 2003, 24)

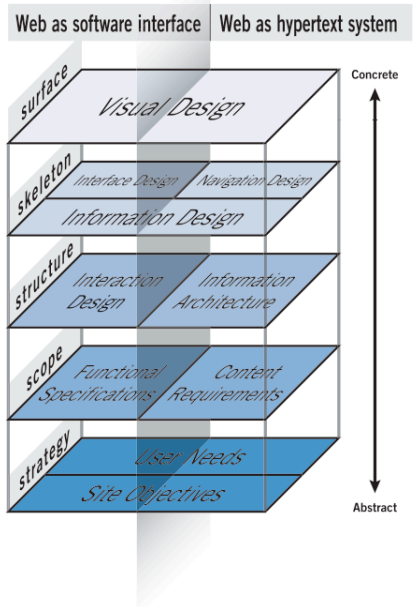

Figure 8: The elements of user experience (Garrett 2003, 33)

The figure shows interaction design as one part of the whole user experience. Garrett (2003) explains that interaction design defines how system behaves in response to the user. However, he divides many design areas out of interaction design, which is not reflecting the commonly accepted fact that each of those areas overlaps interaction design. In many cases the same person is working on many or even all of those areas.

Neither of above definitions - Saffer's and Garrett's - catches the essence of interaction design. Interaction design seems to be something broad, which is hard to describe by only focusing on specific areas. Something always seems to be left out, even when it is the essential part of interaction design. Interaction Design Association (IxDA) has a broader definition:

> *Interaction Design (IxD) defines the structure and behaviour of interactive products and services. Interaction Designers create compelling relationships between people and the interactive systems they use, from computers to mobile devices to appliances; Interaction Designers lay the groundwork for intangible experiences.* (IxDA 2009)

IxDA seems to be on right track. The definition is not too limited, gives room for different variations and states clearly that interaction design is about structure and behaviour.

2.3.2 Interaction design process

Usually interaction designers have worked inside the waterfall process; their work is done in the design phase. But what kind of process is used inside the design phase? Some definitions exists, those are usually describing the moving from abstract to concrete. Saffer, Garrett and many others have described the process, but the simplest and most truthful description comes from Jonas Löwgren and Erik Stolterman (2004). Their terms of describing the different parts of the process are vision, operative image and specification.

The design work starts immediately when the designer thinks about the project for the first time. At that point designer gets to know - or starts to find out - the

background material, the problem, the requirements, etc. The next thing that happens is the vision, which forms from designer's thoughts. The vision is kind of a strategy, only existing in the mind of the designer. It is definitely nothing concrete - it can be hard to explain - but it is the direction where the designer will be going. After the designer has the vision, the operative image - also known as the concept - starts to take form. At first, the operative image is high level, but soon it will get more detailed. After the operative image is detailed enough, the designer moves to specification phase, where everything from high-level operative image to the smallest details is written into a specification. The specification is used in the next phases of the project as a definition of what should be implemented. (Löwgren and Stolterman 2004)

Löwgren and Stolterman also remind that while the process might seem to be a straight line from abstract to the concrete, it definitely isn't so in the reality.

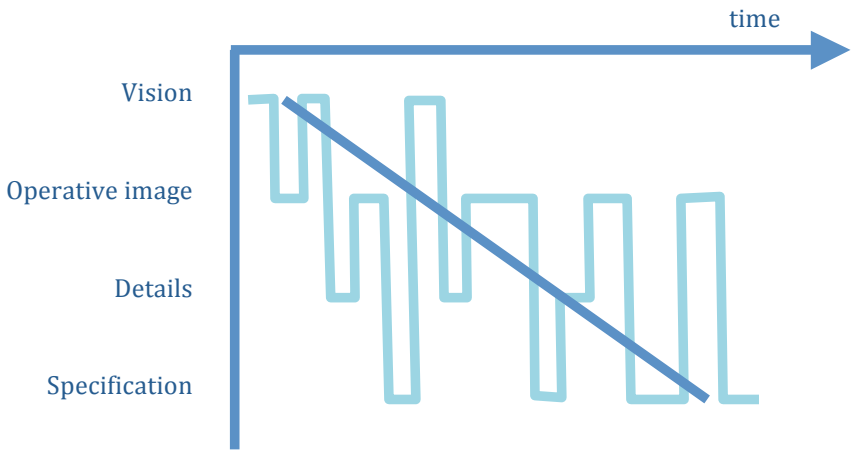

Figure 9: The design process is not linear. (Löwgren and Stolterman 2004, 25)

The figure above describes the truth of the design process: every step will affect to previous and next ones and moving between them will be happening through the whole design phase. Or as Löwgren and Stolterman describe:

> *One of these fundamental aspects is the recurrent leaping between details and the whole, or between the concrete and the abstract. -- It is then necessary to move rapidly and repeatedly between the world of ideas and the concrete*

> *reality of the design situation. Sometimes, this process can seem cumbersome and obstacle to creating a good design process, but it is in fact a necessary and natural part of design work. Accepting this reality and dealing with it is better than hiding behind a model of design work that appears rational or logical.* (Löwgren and Stolterman 2004)

Designers have known the reality so well that they haven't tried to force some logical process inside the design phase. The designer is always responsible to use whatever techniques they need in the design process, it is one critical part of designer's professional skills.

2.3.3 Interaction design techniques

If part of interaction designer's professional skills is to be aware and be able to use whatever interaction design techniques they need, the important question is still not answered: what are those techniques? The question hasn't normally been answered in interaction design literature, although many writers are describing some of the techniques.

The designer's work consists of research, analysing, defining, brainstorming, solving problems, creating alternative solutions, evaluating, specifying and of course redesigning. None of these activities are not finished until the design is ready and design is not ready until the final product is ready. Designers are constantly using all of these techniques during the design process. (Wikipedia: Design; Wikipedia: Interaction design)

Research techniques include interviewing and observing users and their environment to find out about the current situation and the problems in it. This part of the research is about finding out the user needs. Benchmarking is also one research method what designers often use. It is done to find out more about the business goals and opportunities. Benchmarking means that designer analyses competitors in order to find out the common solutions, pros and cons of each competitor design and what is needed to be able to compete with best ones or beat them in the market. The results of different kinds of research have to be analysed

in order to get the most useful information into the design work. (Wikipedia: Design; Wikipedia: Interaction design)

Designer also has to know more about the project than just what research analysis reveals. The client is always interviewed by the designer to define the project more clearly. The designer needs to know how the client sees the business goals and the users, and also the different practicalities of the project: who is doing what, what is designer's role, what is the schedule and budget, how the cooperation and communication is arranged, etc. These are not part of the actual design work, but they affect to the design work so much that the designer needs to know them. (Wikipedia: Design; Wikipedia: Interaction design)

The vision should be in the designer's mind when she has done enough research and analysis. After the designer is aware of the problems and the possible direction where she is heading, she starts the creative process of problem solving. In the creative process the time is important: the designer needs to have enough time to think about all the requirements, needs, goals and possible solutions to those. It is also important to notice that designers look for a solution also outside the current design space if they have enough time. These ideas are often the breakthroughs in the industry, simply because none of the competitors have not looked at the problem from that perspective before. However, usually the design time is so limited that designers are using some of the known creativity or problem solving techniques to quickly create enough possible solutions to be worked on. Perhaps the most used technique is brainstorming with specific group of people, usually with other designers. (Wikipedia: Design; Wikipedia: Interaction design)

To find out the best solutions multiple rounds of brainstorming, discussions and refinement of the problem is usually needed. When the designer has enough refined problem and the possible solutions, she starts to create alternative solutions. Alternative solutions are descriptions of concepts and/or prototypes. They can contain designs e.g. as wireframes or flow diagrams. The designer is constantly evaluating the alternative solutions and getting feedback from other persons involved such as clients, other designers and development team. By evaluation and feedback alternative solutions can merge, some of them can be

dropped and new solutions can appear. The alternative solutions gradually get more details and the designer starts to realize which one of the alternatives she should move specification phase. Before the specification phase, more detailed evaluation is usually done through usability testing or expert evaluation. Interaction designer can do usability testing or expert evaluation, but it is usually recommended to be done by someone else than the designer. The designer might be too inside the design to be able to notice the problems in it. (Wikipedia: Design; Wikipedia: Interaction design)

The usability evaluation gives information to the designer: is she going to the right direction or not, what should be redesigned, what is working well, etc. If the direction is right, the designer can start the design specification. The design specification is a large document describing the whole interaction design, even the smallest details. In the specification phase the designer also needs to evaluate the design all the time. Detailing the design often reveals some new design problems, which then need to be solved. Those problems can also be so huge that they affect to the vision and concept and the designer needs to redesign and possible re-evaluate almost everything. The design specification is mostly done for the development team that builds the product after the designer has specified it. Usually even in the later parts of the projects (e.g. in development phase) the specification is still changing. (Wikipedia: Design; Wikipedia: Interaction design)

2.3.4 Summary

The history of interaction design started when new inventions such as telegraph, telephone, radio and television required the creators to design an entire system of use. Interaction design as a discipline however started when computers evolved to the point when human-to-human was possible through computers. At that point masses started to use computers and more sophisticated interaction methods were needed. Interaction design was not the only discipline trying to answer to that problem and quickly there were so many disciplines existing that none could define what those all were about and how they related to each other. Nowadays when those disciplines have existed for a while, there exist some definitions for each discipline.

Interaction Design Association (IxDA) has defined the interaction design perhaps the most truthfully:

> *Interaction Design (IxD) defines the structure and behaviour of interactive products and services. Interaction Designers create compelling relationships between people and the interactive systems they use, from computers to mobile devices to appliances; Interaction Designers lay the groundwork for intangible experiences.* (IxDA 2009)

The interaction design process is also not so well defined, but that is because the process it so clear even to interaction designers. The process has basically three steps; Löwgren and Stolterman have named them as vision, operative image and specification. The vision is just a direction to which the interaction designer knows she is going, the operative image can already be documented, it is more complete description of the main parts of the project. The specification is the document where the whole design is described, including the smallest details. The process is not so simple that those steps would be done one after each other, but the design is almost always so complex thing that the designer moves between those three steps through the whole project.

The interaction design techniques include research, analysis, defining, brainstorming, solving problems, creating alternative solutions, evaluating, specifying and of course redesigning. Almost always all of these techniques are used in every step of the interaction design process.

2.4 Agile and interaction design

When thinking about all what is written above it is no wonder that agile and interaction design haven't been combined in many projects. There exists many differences between agile and interaction design, but there are also similarities. The main similarities can be found from the basic values: both agile and interaction design are highlighting the collaboration.

Why is it so difficult to merge interaction design into agile? What are those huge differences that make it difficult? First of all, it is important to remember that agile methods were created by developers for developers. Creators of agile methods didn't think about the interaction design as a part of the process, because software projects didn't either have interaction designers or they were working separately from developers at that time. The worst threat of the interaction design professionals is really much true today: agile is not even suggesting anything related to interaction design in the process and agile methods are taking over the whole software development world. Just when everyone started to realize that interaction design is important when creating products for users, almost the whole user experience is left without any support from process side.

When agile methods are referring to design, they are usually talking about the code design, done by the developers. Some references to user interface (UI) can be found, but in those cases it is suggested that developers do the creation of UI inside iterations. Of course, that approach works well in some projects, but most of the software projects would advantage of the proper interaction design. To be able to merge interaction design into agile, it is important to see the main differences.

2.4.1 Differences between interaction design and agile

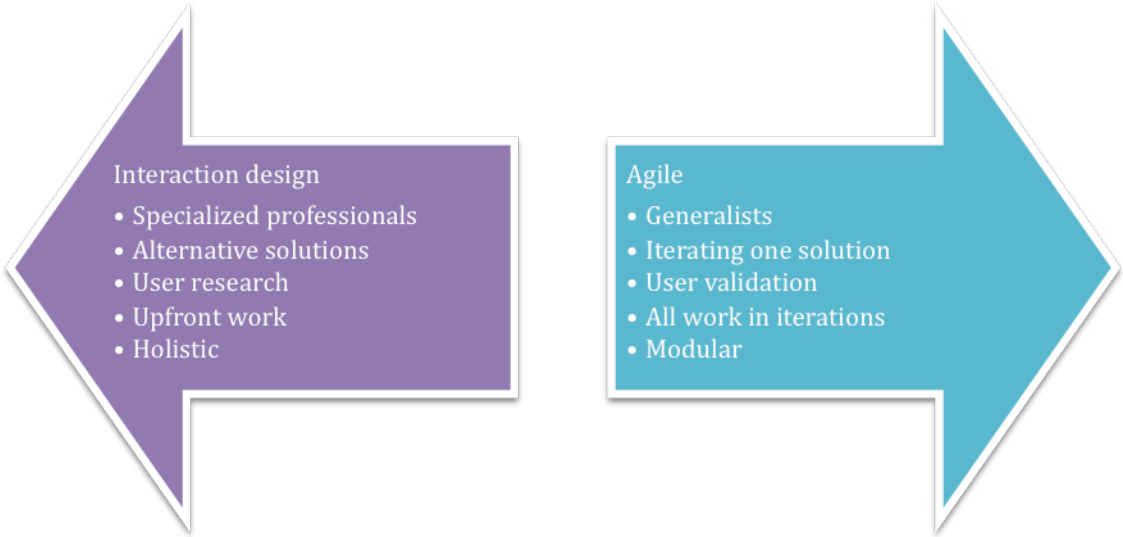

Figure 10: Differences between interaction design and agile

There are many differences between interaction design and agile, but the most important ones are the five main conflicts. First one is about the roles: while interaction design needs a specialized professional, agile says that all team members should be generalists. The second conflict is about the alternatives: interaction design uses alternative solutions as a basis for design work, agile iterates only one solution. The third conflict is about usability: agile recognizes the importance of user validation but doesn't give any time for user research which is one fundamental aspect of interaction design. The fourth conflict is about history: for interaction design it is natural to do at least some of the design up-front, but agile specifically tries to avoid that by doing all work in iterations. The last conflict is about the way of thinking: interaction design starts from holistic point of view, but agile uses modular thinking.

The last one is the most interesting difference. John Armitage (2004) has illustrated that difference and how it can affect to the final product. He first shows example progression images of waterfall and agile approaches (Figures 11 and 12). In waterfall, the design defines the holistic overall picture - the sketch - first and then proceeds gradually to higher fidelity by specification and then to final product by implementation. The agile approach is different, in ideal case the progress is seen by the series of finished pieces, which together create the final product.

Figure 11: Staged progression from low to high fidelity. (Armitage 2004)

Figure 12: The ideal XP result: a series of finished pieces assembled serially into a cohesive whole. (Armitage 2004)

The waterfall however has few risks. First one is shown in Figure 13: The design is done well, but for some reason or another, implementation is not. The result is distorted final product. The second risk is illustrated in Figure 14: The design is not finished so the implementation isn't either. The result is unfinished product, which can't be used. Agile has also a risk, illustrated in Figure 15: The holistic view is missing; so individual pieces don't fit together well. The final product is not what users and customers want it to be. (Armitage 2004)

Figure 13: One risk of phased processes: Design is accurate but implementation is poorly crafted or unreliable. (Armitage 2004)

Figure 14: Another risk of phased processes: Design never ends and implementation is never finished. (Armitage 2004)

Figure 15: The risk of the XP approach: Individual pieces have value, but the larger system is disjointed. (Armitage 2004)

Figure 16: The answer: Split your time investment between overall design and detailed, iterative development. (Armitage 2004)

Armitage also states the answer to these problems; it is illustrated in Figure 16. The answer is to split the design time between the holistic overall design and detailed iterative design. This answer shows a promise that there could be a way to merge interaction into agile.

2.4.2 Combinations of interaction design and agile

Currently there exists few descriptions of projects which have already combined interaction design and agile. All descriptions have something in common: they use similar kind of modified agile process. They also state some other suggestions than just process descriptions, mainly about the values and working habits.

Maria Giudice (2008) asks an interesting question with her presentation *Can't we just all get along? Human-centered design meets Agile.* She has understood that it is important for both developers and to designers to create a product that is appreciated by the users. She states that we can get along and proposes the following process:

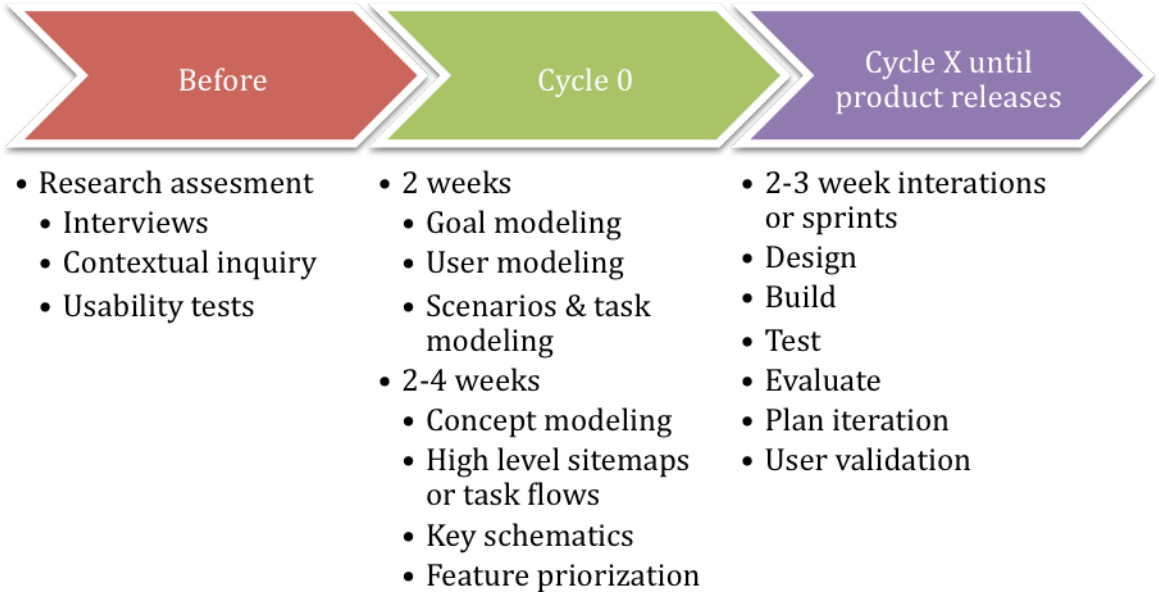

Figure 17: Human-centered-Agile (Giudice 2008)

Giudice's process starts even before cycle 0 by doing the research. Cycle 0 is dedicated for the holistic overall designs like modelling goals, users and concepts. In cycle 0 also the feature priorization is done. After that starts the detailed iterative design in cycles which continue until the product release. Notable thing is that Giudice suggests that user validation - usability testing - is also done inside cycles. Giudice states clearly that all stakeholders and team members - project lead, client, user experience, visual design and engineering - should be participating through the whole process.

Jakob Nielsen and Chris Nodder (2008) have done two rounds of research: a survey of 105 design and development professionals and more in-depth case studies from 12 companies and their Agile projects. They have valid data of the current situation about the projects which have merged usability and agile. The study is not about the interaction design with agile, but Nielsen and Nodder have concentrated to both usability and design so the results and suggestions are good material for interaction design too.

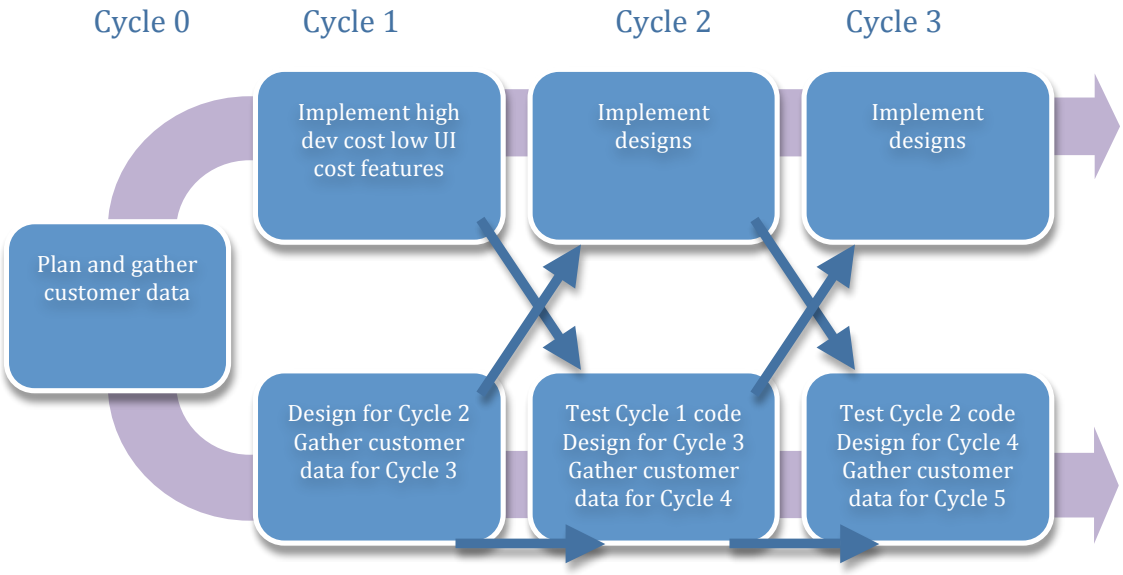

Figure 18: The cyclic parallel track process (Nielsen and Nodder 2008)

Nielsen and Nodder describe almost the same process as Giudice, but they propose to make everything in parallel. This means that design works separately from development. Process would go so that cycle 0 is used for planning and gathering customer data. In cycle 1 development starts by implementing low UI cost features so that design has time to do the design for cycle 2 and gather customer data for cycle 3. In next cycles development implements designs done in previous cycle. Design focuses on three things: testing code from previous cycle, designing for the next cycle and gathering customer data for the cycle after next one. Nielsen and Nodder remind that even without using parallel tracks, staying ahead of the development team is beneficial.

Cennydd Bowles (2008) has written an article where he states that now it is the time to get real about agile design and prove that design can adapt. In the article he describes the "best practice" of agile design:

> *"Best practice" suggests that designers should research iteration n+2, design iteration n+1, support iteration n and review iteration n-1.* (Bowless 2008)

When Bowles talks about research, he means usability testing. Bowles also reminds that the best practice shouldn't be taken too literally and that designers

should use their experience when deciding what should be done when. Some stories are way too complex to be designed in one iteration.

Bowles suggests that interaction designers should get themselves involved in writing the user stories. By doing that designers can make sure that the user stories constitute a chain - called journey - which is doing the same as the traditional vision; making sure that the design is going to one direction, not to many. This way the final product should be more consistent. Bowless has found out that some agile teams use iteration zero not only before starting the actual iterations, but also in the middle of the project. The mid-project iteration zero is basically an iteration where no user stories are delivered. Instead, it is used for tidying up the code and planning the next steps. For interaction designers it is a good time for doing research, revisiting the vision and making sure that the design is coherent.

Agile is light on documentation, so the designers don't have to do heavy specifications. Bowles recommends using lo-fi sketches and paper prototypes as a mechanism for sharing design ideas. Designers can iterate their work easily by sketching lots, quickly and often. Bowles reminds also that working together is one of the key elements of agile, but it doesn't mean that specialists wouldn't be needed. Design shouldn't be dictated by designers - nor should it be a democracy. Shared ownership of the design should be fostered, because everyone in the project has the same goal: making sure that clients and users get the best product the team can create.

2.4.3 Advantages of the process change

Studies and process descriptions are important when considering the process change, but they do not give much inspiration to interaction designers and developers. Inspiration is definitely one of the key issues for any change - process change is not an exception. Individuals have to want the change, otherwise the change resistance will appear and possible even cause the failure of the change.

So what is so much better when interaction design is merged into agile? The best descriptions of the advantages come from the people who have been part of the

combined process. Paul McInerney and Frank Maurer (2005) have done a case study, where they interviewed three user centred design (UCD) specialists who were working on their first agile project. The results of the case study were positive:

> *"-- all our UCD practitioner reports were positive. Certainly, work-life aspects were positive - they felt actively engaged in a common goal. Our participants said the resulting UI designs may or may not be better but are certainly no worse. These case studies can be seen as interim reports: Agile methods are still maturing, and no case study participant had yet shipped their first agile project."* (McInerney and Maurer 2005)

Also Giudice (2008), Nielsen and Nodder (2008) think that for interaction designer, working in agile is really much different but definitely no worse than working in waterfall. In agile, designer doesn't need to get the design right at first. As long as the direction is known, everything can still change. This principle gives a lot more time to nail the design. In waterfall, designers often have a thought at some point that if I would do that again, it would be so much better. In agile, designer can do the whole thing again if needed. Less documentation means different things, but for designers it means more time to think and less time used in writing - that can't be a bad thing. Working constantly with other people is more inspirational, discussions help to do the design. When sketch and conversation are used to communicate the design, development can ask the questions. This means that they will clearly understand what the design is all about. And when they understand that, they will start to embrace design by themselves. Gradually the whole team will be committed to design issues because also developers will feel that it's their design, not just designer's design.

2.4.4 Important things to consider before diving into agile world

There are many good things in agile way of doing the design, but there are also things that the interaction designer should know and consider before diving into agile world. Guidice (2008), Nielsen and Nodder (2008) have also listed some of the considerable things.

Agile process is much easier when you have experienced professionals in the team and the managers and stakeholders are also committed to change the ways of working. Experienced professionals don't have to know anything about agile, but being experienced in own work gives more time to focus on the process for a while. When starting with agile, it can be a good thing to have a coach to teach the basics to the whole team. The coach has to know something about interaction design and how to do it in agile. Otherwise it might be harder for the designer to try to convince the team that because of interaction design, strict agile process is not possible. The interaction designer should be able to propose changes to the process and those changes should be done - after all agile is supposed to be flexible. (Giudice 2008; Nielsen and Nodder 2008)

Interaction designer should have a possibility to concentrate to the vision first, for example in the iteration zero before the real iterations start. The vision should be kept in mind all the time when doing the modular design inside iterations. Even the vision can change, but then already implemented parts should be checked again. Getting ahead implementation should be done also in iteration zero. While agile doesn't require heavy documentation, the design work is much easier if previously done designs are available somewhere. Documentation can be done e.g. in wiki or in some other flexible tool instead of traditional heavy specification tools. The most important thing is to remember to add changes to the documentation constantly. When changes are needed, the designer must be able to do quick decisions. Building better communication between the whole team and all stakeholders is important. (Giudice 2008; Nielsen and Nodder 2008)

In agile, communication is one of the key values. Communication should happen from early on, one of the designer's responsibilities should be to guide developers out of the feature discussions to the user goal discussion. To be able to do that, the designer has to be part of the team and let the design be cooperation. It is hard to give part of the responsibility to others, but the designer must be able to trust her team enough to do that. Even though part of the responsibility is on the whole team, the designer must always remember that she owns the design, so she has to make the final decisions. Keeping in touch with other interaction designers can be a good thing for getting help and comments for the design work, but designer's

main responsibility should always be the agile team. The team will most probably need designer's help through the whole development, so designer should always be available for the team. Co-location is the most effective way to be available and part of the team. (Giudice 2008; Nielsen and Nodder 2008)

2.4.5 Summary

Agile and interaction design have differences even in their main principles. This is one of the reasons why interaction design hasn't been combined with agile often, but the main reason is still that the agile methods do not give any instructions how the combination could be done. The differences can be tackled by compromising both interaction design and agile a little bit and by remembering that both are essentially created for one purpose: to be able to create the best possible product for the end user.

All existing descriptions of combined processes seem to follow the same principle: some time is used for concepting before the implementation starts and after that the detailed design is done in iterations (sprints, cycles). In iterations, the design stays a little bit ahead of the implementation, so that there is enough time for the research needed and other design activities. The interaction designer also has to support ongoing iteration and review the last iteration done. All studies also remind that the concept (operative image) should be kept in mind all the time when focusing on the small pieces of the whole product.

The basic iterative process is not enough for projects which want to combine interaction design and agile. Also some working practises and basic principles need to change dramatically when compared to the traditional waterfall process. The main difference is that interaction designer should be working as a part of the agile team, not so much as a part of the design team. The highlighted practises and principles also include constant communication, light documentation, defining the process details for each project individually, ability to change the design in the middle of the project, etc. These practices and principles are the key point to the successful combination of interaction design and agile.

3 Methods of research

The intention in this study is to find out what effects the agile process has on interaction designer's daily work. The focus is on the practice: the aim is to describe history and current reality so that the study gives real information which any project could use when starting to combine interaction design into agile.

The study starts with the theory part that explains the basics of agile and interaction design: why they were born and what they are. The theory part also goes through the differences and similarities of agile and interaction design, as well as summaries studies already done of projects, which have merged those two.

The second part is a case study about one Nokia project. It gives a detailed overview of one specific project where interaction design is part of agile process. The aim is to describe the used process detailed enough, so that anyone can understand how the combination of agile and interaction design can be done in practice. This does not of course mean that the process that is used in this project would be even near perfect. To highlight that fact the case study also has the results of a survey, which was conducted to find out the areas of interaction designers' work, which are not successful in this particular process.

3.1 Justification of methods

The main focus area of this study is the theory part. It is based on literature and many online references, including studies already done about the subject. The theory works as a basis for suggestions, which are gathered together in conclusions part of this study.

The aim of this study is not to create generalizations, because it is quite clear that generalizations won't work in most of the cases, which are much different from each other. Instead of generalizations, this study gives suggestions based on the current knowledge on this subject. Other studies and literature try to look for generalized, one for all process. This study recognizes that such process does not exist: every project should create their own process. In order to do that, it is useful to know about the previous projects that have already been using new process

model for some time. The case study tries to answer to that need by giving a detailed description of one specific project. In addition to the description, the case study also includes the results of a survey, which gives quantitative data of interaction designers' work in the current process model.

3.2 Data collection and analysis

As described before, the theory part of this study is based on literature and online references. The data for the case study has been gathered by interviewing and by conducting an Internet survey. The case study project is a Devices R&D project in Nokia.

3.2.1 Nokia

Nokia is a Finnish multinational Internet and communications corporation. Nokia is world's #1 manufacturer of mobile devices, with estimated 39% share of global device market in 2008. Nokia had over 128 000 employees in 120 countries at the end of 2008. In 2008, Nokia reported net sales of 50.7 billion Euros and operating profit of 5.0 billion Euros. (Nokia 2008)

Nokia has strong R&D presence in 16 countries. In 2008, there were over 39 000 employees in R&D and Nokia's R&D investment was 6.0 billion Euros. In Nokia organization, Devices is responsible for developing and managing Nokia's device portfolio. (Nokia 2008)

3.2.2 Project A

The studied project is a Devices R&D project in Nokia. In this study it is called Project A. Project A aims to create a new mobile product by designing both hardware and software. As in most Nokia projects, there is no obvious client and development is oriented towards a broad market. Project A is quite large scale but still focusing totally on creation of one design centric product. The new agile process model has been defined for project A because it was clear from the beginning of the project that changes will happen through project's lifecycle. It was

also clear that more traditional agile processes wouldn't work because project A is so heavily design centric.

3.2.3 *The interview*

The project description has been written based on the interview of Kimmo Tuomainen, who has done most of the work on defining the process model for project A. The semi-structured interview has been on going in spring 2009. The semi-structured interview was used because it gave an opportunity to have a series of discussions, which raised more issues up than structured interview would have done.

3.2.4 *The Internet survey*

The Internet survey was conducted in April-May 2009 with interaction designers working in project A. The aim was to collect quantitative data and analyze it to find out which parts of interaction designer's work are not working well in the current process model.

The Internet survey was created with Nokia's internal online survey tool. Invitations were sent to all interaction designers working in project A. Out of 22 interaction designers, 13 participated the survey.

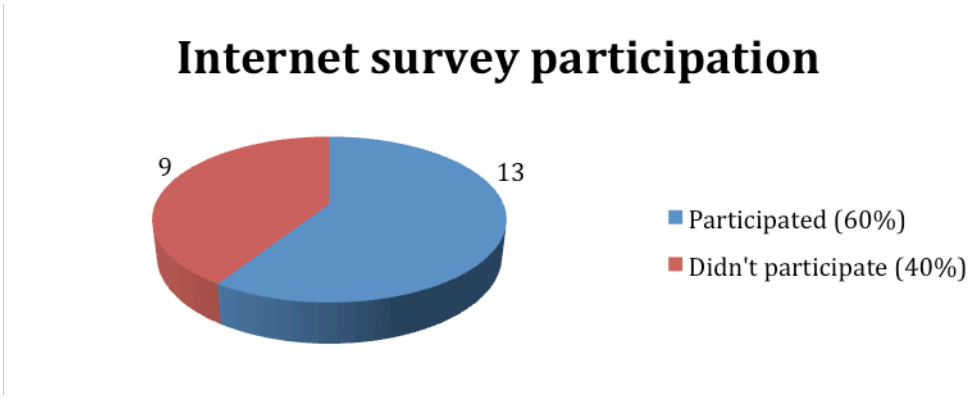

Figure 11: Internet survey participation

The questions of Internet survey were formed so that it was possible to use Likert scale or simple yes/no answers. On the other hand, the intention was not to limit

answers so strictly, so also comment fields were available. The questions were divided into six pages and the whole survey was nine pages long.

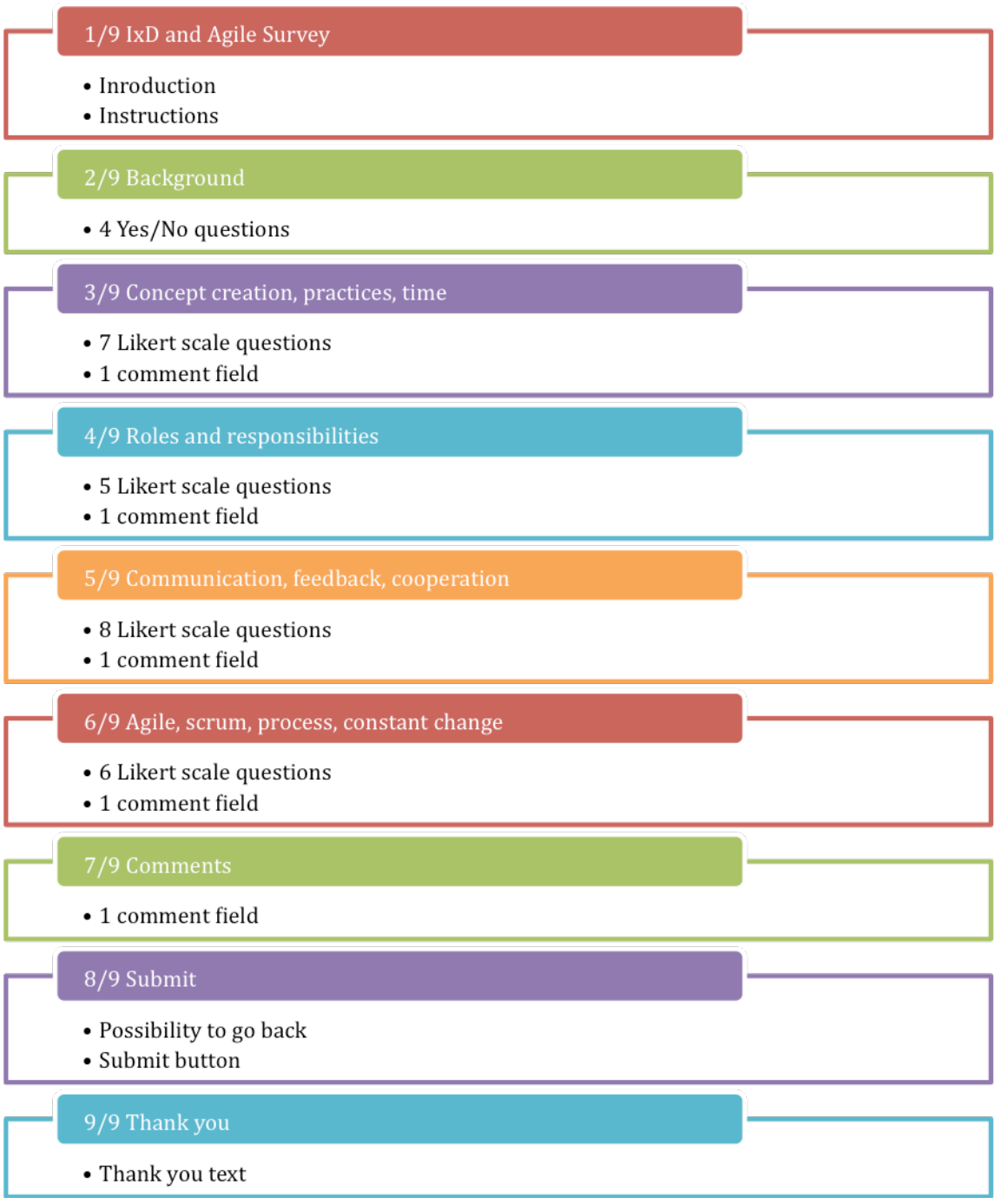

Figure 12: Internet survey structure

The 7-point Likert scale with additional 'Don't know' option was used in main question pages. In background questions only simple Yes/No options were offered.

For unstructured answers a comment field was also available in each main question page and additionally in the separate Comments page.

The results of the Likert scale questions are combined in this study. The following chart shows how the combination has been done.

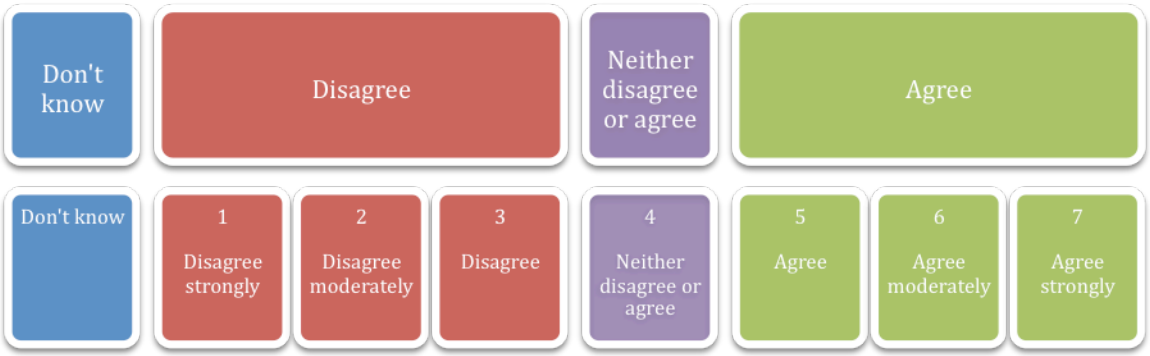

Figure 13: Combination of Likert scale answers

For the whole survey and questions, please refer to Appendix 1: Internet survey. For all comments received in the comment fields of the survey, please refer to Appendix 2: Survey comments.

4 Case study: Project A, Nokia

The case study describes the project A's process model and the survey results. The process model has been gradually taken into use when different parts of the project A have been starting. The process model has been changing and clarifying through the project A's lifecycle and it still continues to change. Those changes are done based on the learned lessons, but the main structure of the process hasn't been changing much.

4.1 The process model

The process model has been defined based on the Nokia level UI guidelines, Project A UI guidelines and two facts: project A is doing new UI from the scratch and there are no unlimited resources available. The process model is build on top of the commonly known scrum process, which is described in chapter 2.2.2. Whole project A is huge and had to be divided into several asset teams, each of which is taking care of one part of the whole project. Asset teams also form scrum teams. All asset teams have their own schedule; so one team can still be in concepting phase while the other has already done development for a while.

4.1.1 Roles and responsibilities

The traditional scrum process suggests that there shouldn't be any specific roles inside the scrum teams. This works well when only development - architecture, code and testing - is done inside the scrum team. When design is added to the scrum, some modifications were needed. In modified process there are three special roles in the scrum team: interaction designer, graphic designer and usability specialist.

As described above, interaction designers work as specialists inside the scrum team. Most of interaction designer's time is spent inside the scrum team, but interaction designers also work with other interaction designers to take care of the high-level concept and all dependencies. There are also dedicated senior interaction designers for that work, but they can't do everything.

Each scrum team has to cope with shared resources; interaction designer works typically in 1-2 scrum teams, graphic designer in 2-3 teams and usability specialist supports usually 4-5 teams. This fact of shared resources has to be though when deciding several things in the scrum teams. Typically scrum teams need to consider together e.g. meeting times and when they need more support from shared resources than usually. These can't overlap much, so that shared resources could be part of all scrum teams as much as needed. Of course that is not always possible, and those cases are solved so that interaction designer is mainly responsible of only one project and can then prioritize the work based on that.

Each scrum team is responsible of creating their own part of the project A. Development work is done together inside the scrum team, but the design work is divided between different designers and specialists. Interaction designer is responsible of the interaction design: mainly meaning specifying the logic and flows into the specification. Graphic designer is doing the final layouts and is responsible of the layout document. Interaction and graphic designers work together to define views and metaphors. Usability specialist is responsible of giving usability feedback about the whole design and arranging usability tests.

4.1.2 Concepting phase

The process model concentrates on the development phase of the project, so it does not define the concepting phase. Concepting can be done using the agile process or it can follow whatever suitable process. Important thing to notice is that if agile process is used, the process should bend so that the designers can do the concepting freely enough.

The high-level concept defines the vision of the project A and it is created in concepting phase. Each asset team has their own high-level concept, where the task flows for main use cases, main views and layouts and main dependencies are defined. The usability tests are done to ensure that the high-level concept is mature enough to be used as a basis for detailed design. The feasibility is also checked by the development, so that there are no major obstacles coming in implementation when the development starts.

4.1.3 The sprint flow

The main difference with the traditional scrum process is that the design work is done one or more sprint(s) ahead of implementation. This violates the basic principle of scrum where everything related to one backlog item should be done in one sprint. The modified process model however recognises that the design needs more time than just part of one sprint. The basic rule in the modified process is that the interaction designer is at the same time preparing for the next sprint by writing the specification, supporting the current sprint by detailing the specification and checking the work done in the previous sprint.

Each sprint should start with the traditional sprint planning meeting. In addition to that, there should be a meeting held about the sprint's designs. In that meeting the design should be gone through and discussed so that e.g. open issues come up early enough and not in the middle of the sprint.

During the sprint the traditional daily scrum meetings are held, but shared resources may not be able to participate in each of those. Each scrum team can decide how often the shared resources should be present. Otherwise the work inside the sprint follows the traditional scrum process. Also the sprint demo and retrospective follow the traditional scrum process.

4.1.4 Product backlog

The product backlog is the most important document in the scrum process. It includes all needed features as items in priority order. Everything which is done in sprints comes from the product backlog. In the modified scrum process, project A uses *user story* based items in the product backlog. Even though the user stories are originally coming from the XP, they are spreading to other agile process models. The user story is easy way to ensure that the backlog items are not just features, but something that gives real value to the end user. User story is written by completing the following sentence:

As a user, I want to … so that I …

The design needs own items in the backlog so that those can be taken into the sprint backlog before the development. This is again violating the basics of the scrum process and the user stories, but it is mandatory so that the design can stay ahead of the development.

4.1.5 Other documentation

Agile and scrum do not appreciate heavy documentation, but all projects can't be run without documentation. Especially large projects - like project A - will have serious issues if the whole project would just get rid of the documentation. As described before, project A relies first to the concepting phase where the high-level concept is created. Each asset project creates one concept document that clearly shows the maturity, feasibility and usability of the design. When creating the concept document, also all dependencies are checked and the design is aligned with other asset projects.

After the concepting phase starts the development, where the main document for each asset project is the specification. As high-level concept has already been created and documented, specification is the place for detailed design descriptions. The specification is not written at once as traditionally in waterfall process. In project A, writing is done one piece at a time during the sprints. This helps significantly designer's workload on the early phase of the project and gives also the development team a way to see how design is progressing. As the specification is done one piece at a time, changes are not so difficult to handle: the designer is already in the team and concentrating on the same project as the development. The specification remains to be draft until the development is ready. After that point the specification reflects the created product and is final.

Most important documents for interaction designers are the concept document and specification. Of course project A has also lots of other documentation, including different kinds of guidelines, architecture documents, usability documentation, meeting minutes, etc. These documents are created and used because they are needed, not just because everything should be documented.

Important thing to notice is that while project A has individuals working around the world, the aim is to keep all documentation available for everyone all the time.

4.1.6 Issues with the process model

When the process model was first defined, it was already known that some issues would arise. Those issues have been tackled after they have been identified. The new arrangements have been done where needed; basically the process model is just the backbone for asset projects and every asset project should define the details by themselves. The team knows best what works for them and also knows when they need to change something related to the process. The asset teams have had a coach available when first starting the scrum process and then on need basis. When there have been problems, the coach has e.g. suggested the practise to be changed to be more like some other agile method than scrum. That kind of modifications can be done as long as the main process stays the same for all asset projects.

Smaller issues have been rising because of the limitations of project A: only limited resources are available, the asset team members are not always co-located, the process and roles are not always clear for everyone, etc. All of these obstacles could have created big issues, but they were realized soon enough and they have been tackled at least somehow. For example, the team does not necessary need to be co-located if the team can and is willing to use different sorts of virtual meeting tools.

4.2 The survey results

As described in chapter 3, the Internet survey was conducted to find out the areas of interaction designers' work, which are not successful in this particular process.

The survey participants were first asked few background questions to find out some basics of the current situation. The intention was to find out if the process model was used, if the interaction designer was working as a part of the process, if the interaction designer was working in several teams at the same time and if the interaction was co-located with the team.

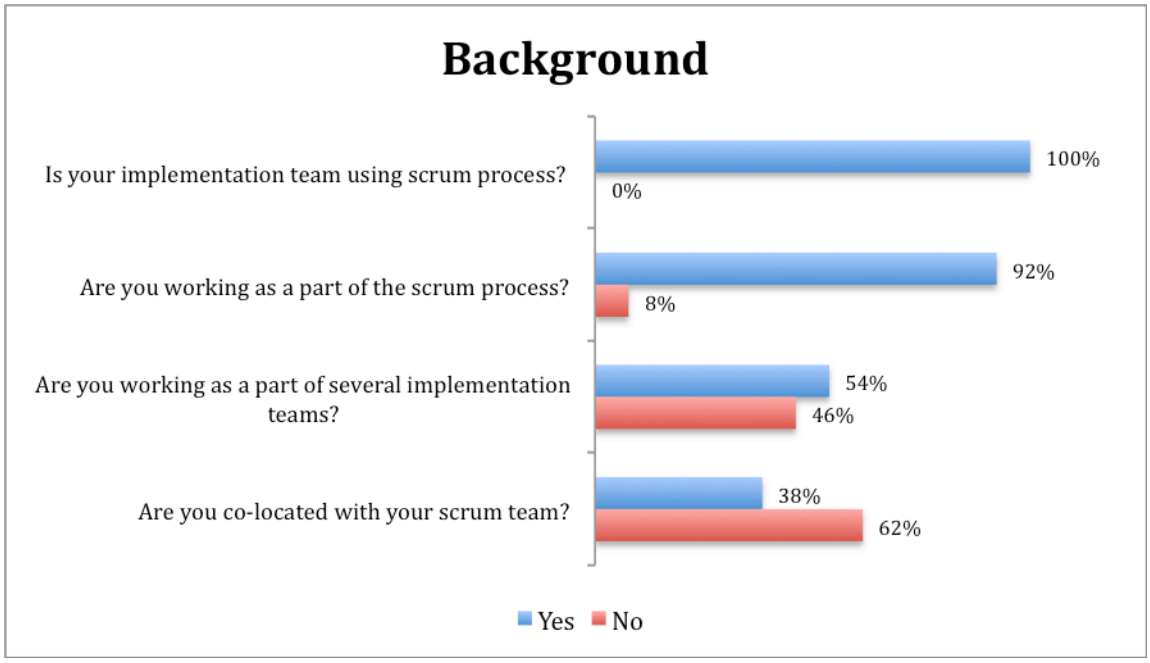

Figure 14: Background questions

All of the implementation teams were using the modified scrum process. 8% of the interaction designers are not working as a part of the scrum process. Over half (54%) of the interaction designers are working as a part of several implementation teams. Almost 2/3 (62%) of interaction designers are not co-located with their scrum teams.

The background questions already reveal some of the problems. All implementation teams use the scrum process, but not all interaction designers are working as a part of the process. About half of the interaction designers have to divide their time between several implementation teams and co-location is not common in project A.

The first actual question page concentrated on three things: concept creation phase of the project A, process practices and time limitations.

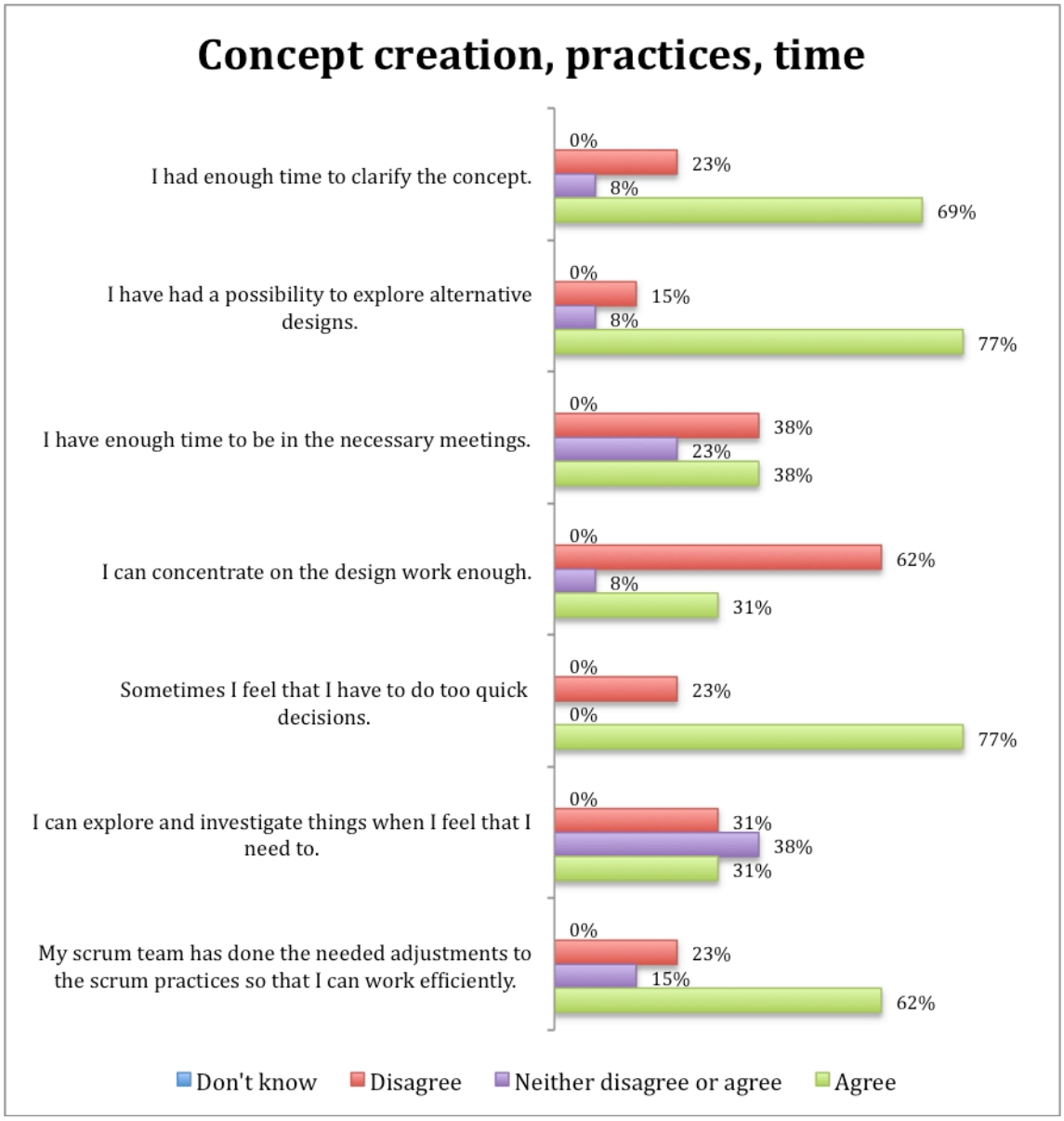

Figure 15: Concept creation, practices, time

Most of the interaction designers agree that they have had enough time to clarify the concept (69%) and they have had a possibility to explore alternative designs (77%). Having enough time for necessary meetings seems to divide opinions - 38% disagrees as well as agrees. Possibility to concentrate on the design work seems to create some issues (62%), as well as the feeling that decisions are made too quickly (77%). Exploring and investigating things divides opinions again, this time 31% disagrees as well as agrees. The needed adjustments to the scrum practices in the scrum teams are happening (62%), but not in every team (23%).

Having enough time is a clear issue to the interaction designers. It seems that there is not enough time to concentrate to the design work and some of the designers are struggling to have time to be in the necessary meetings or explore and investigate things. Designers also have to do too quick decisions, but one designer commented that it has been the case always and just has not changed when using scrum process. Some designers have issues with their scrum team's practices and team hasn't done needed adjustments. However, most of the teams have done needed adjustments.

The second question page concentrated on roles and responsibilities.

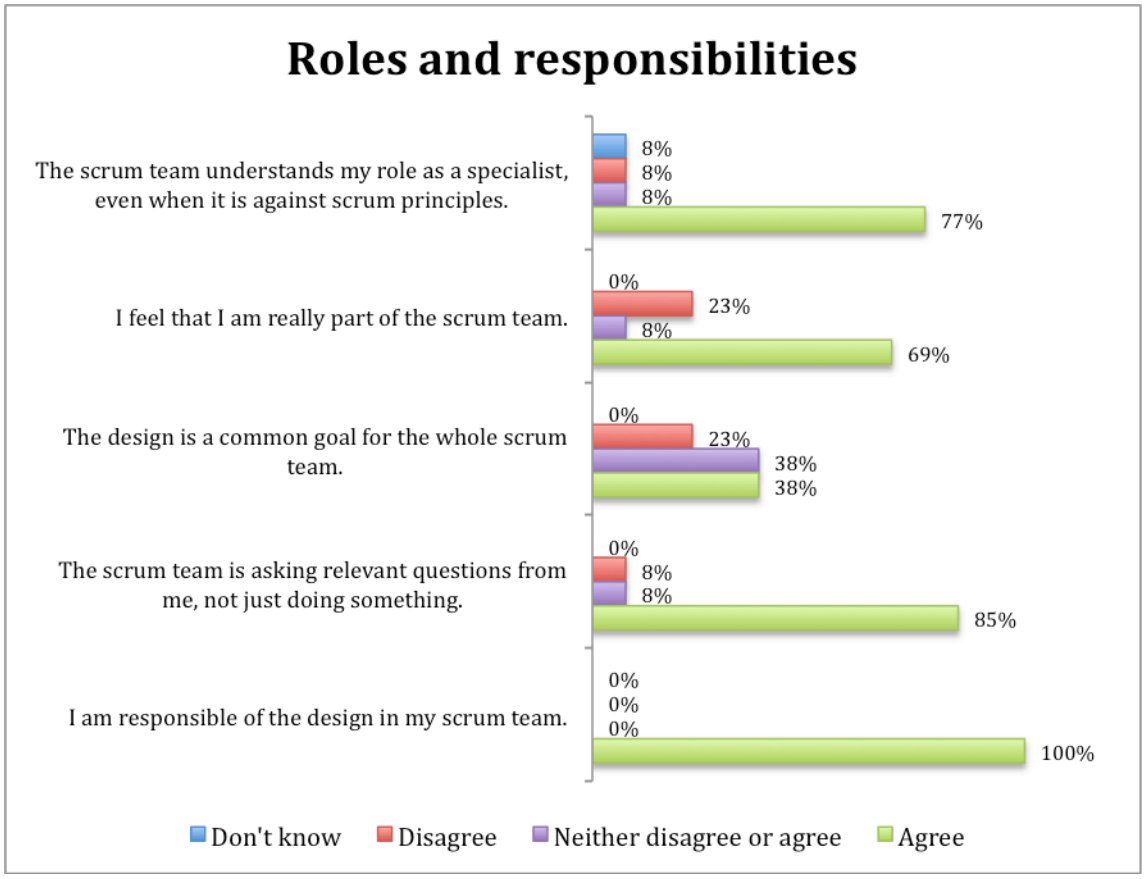

Figure 16: Roles and responsibilities

Almost all (77%) scrum teams understand the interaction designer's role as a specialist. 69% of interaction designers feel that they are part of the scrum team, but 23% are not agreeing. It seems that the design is not often a common goal for the whole scrum team, only 38% is agreeing. Most of the designers (85%) agree that their team is asking relevant questions from the designer, not just doing

something and all designers (100%) feel that they are responsible of the design in their team.

While many of the designers feel that they are part of the scrum team, the minority who feels opposite is not so small that it could be ignored. Also the design seems not to be a common goal for the most of the teams, or then designers are not that much involved with their teams that they would know the answer. One designer also commented not having enough visibility to the team. Otherwise roles and responsibilities seem to be clear to the designers and their teams.

The third question page was about communication, feedback and cooperation.

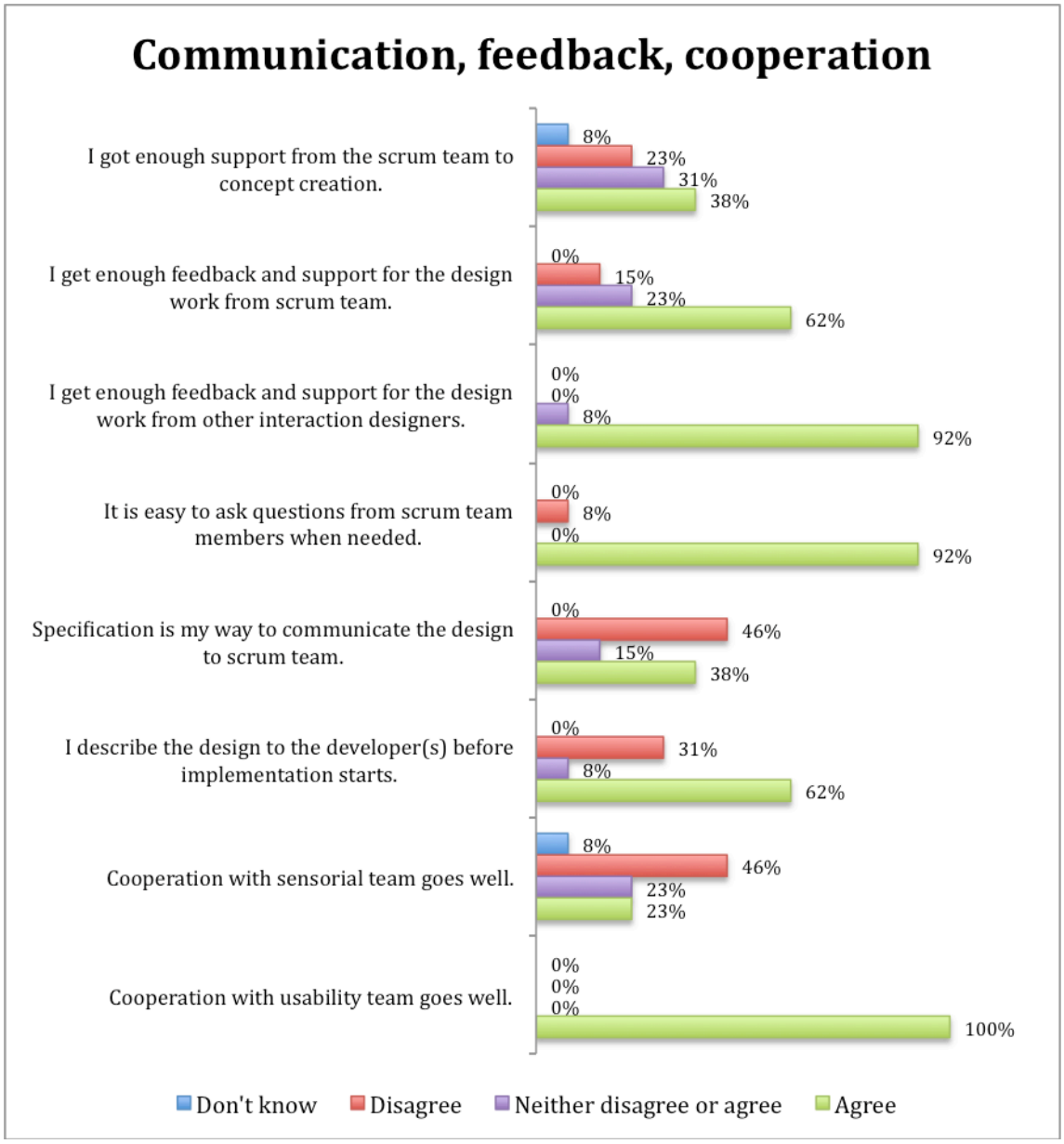

Figure 17: Communication, feedback, cooperation

It seems that the support from scrum team has been missing at the time of concept creation, only 38% of the designers got enough support, while 23% didn't get enough support. For the general design work, designers are getting enough feedback and support from other interaction designers (92%). Only 62% of the designers agree that they get enough feedback and support from scrum team, while almost all designers (92%) claim that it is easy to ask questions from scrum team. Specification as a designer's way to communicate the design to scrum team divides opinions - 46% disagrees and 38% agrees. 31% of the designers do not describe the design to the developer(s) before implementation starts, while 62% of

the designers do that. Almost half (46%) of the designers feel that cooperation with the sensorial team is not going well, while only 23% do not have issues. All designers (100%) are happy with the current state of the cooperation with the usability team.

The issues with the lack of support from team to the designer in concept phase might be a bigger problem than it seems from the figures. Those issues often cause problems later on and the problems get huge when all designers are still not happy with the amount of feedback and support in later phases of the project. It seems that asking questions is easy, so the hard part might be about getting the answers or just about having to ask everything. More troubles can be seen when about a third of designers are using the specification as a way to communicate the design to the team and do not describe the design to the developer(s) before implementation starts. There were positive comments about using something simpler than specification as a way to communicate the design to the team. There are also significant problems in cooperation with the sensorial team and one designer has commented that the problems are caused by the lack of resources.

The last question page asked about agile, scrum, and process in general and about the constant change.

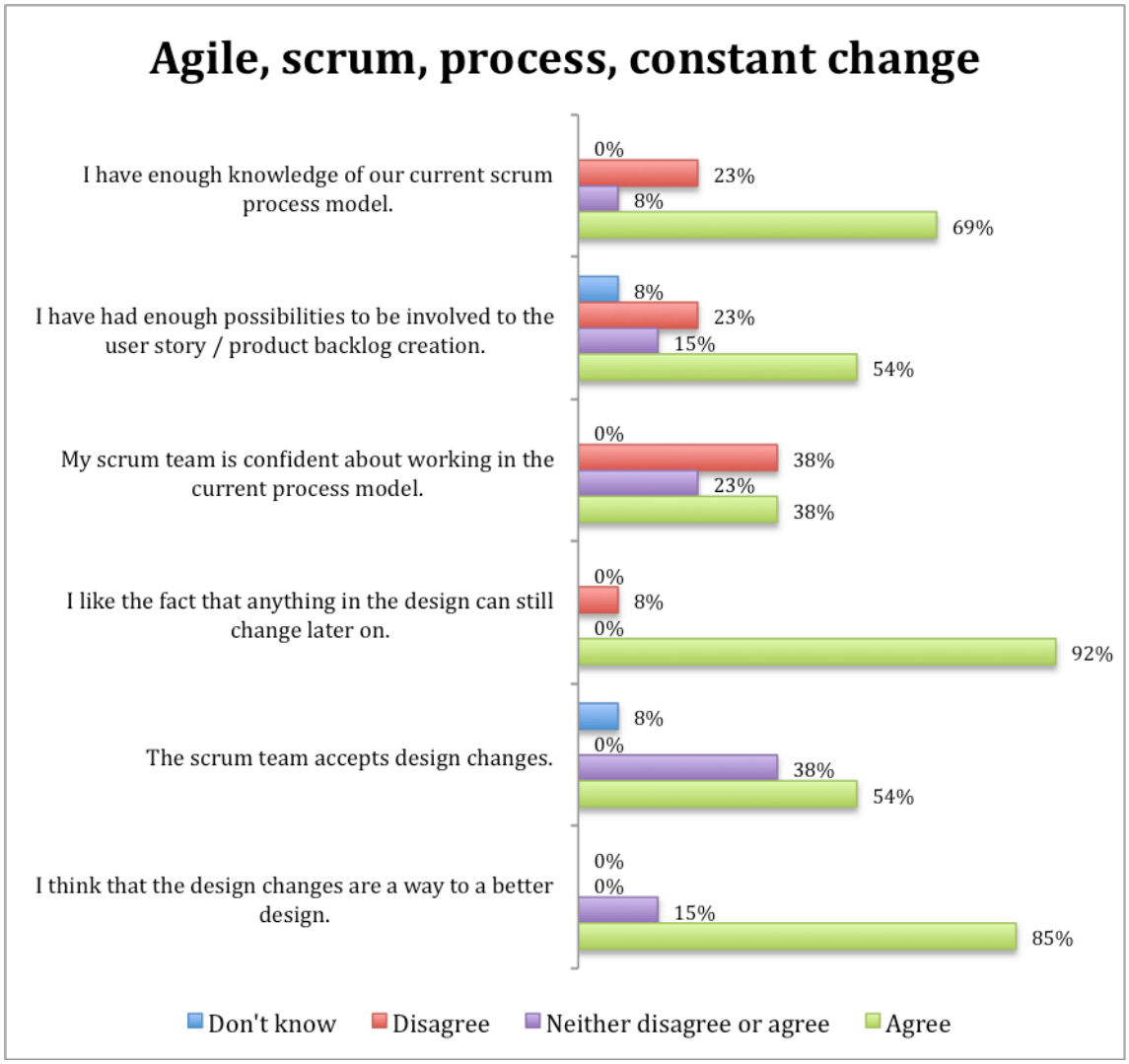

Figure 18: Agile, scrum, process, constant change

While most of the designers (69%) feel that they have enough knowledge of current scrum process model, significant minority (23%) feel opposite. About half of the designers (54%) have had enough possibilities to be involved to the user story / product backlog creation, while 23% feel that they have been left out. The teams confidence level divides opinions: 38% both disagree and agree that the team is confident about working in the current process model, while 23% neither disagree or agree. Almost all designers (92%) like the fact that design can still change later on, but only about half (54%) agree that the team accepts design changes. None is disagreeing, but 8% don't know and 38% neither disagree nor agree. Majority (85%) thinks that the design changes are a way to a better design.

There could be more information offered to the designers about the current process model. The designer should always have a possibility to be involved to the user story / backlog creation, and that is not happening yet in every team. Half of the designers do not seem to know if the team is accepting the design changes or not. One designer commented that the changes could also be a sign of inability of strong decisions, while most of the changes are just for creating the better design.

The last question page had only one comment field for any comments that were not written before. The designers commented the following issues: support for decision making is needed, physical co-location is absolutely important, small modifications had to be done to process model to suit the team, the developers do not always understand that the whole design can't be done in sprints, time for specification writing is lacking.

4.3 Summary

The project A's process model is created on top of the scrum method which is described in chapter 2.2.2. The defined process model concentrates on the development phase of the project; the concepting can be done using any suitable process. The most significant differences related to the interaction design include the interaction designer's specialized role in scrum team and the separation of design tasks in the product backlog so that the design can be done ahead of the implementation. Other notable things are dividing the whole project into smaller asset projects, usage of shared design resources and the fact that asset projects can define the details of the process by themselves. The problems are mostly tackled individually in the asset project instead of changing the whole project A's process model.

Based on the survey results the problematic areas for interaction designers seem to be following:

- The lack of time when interaction designers need to concentrate on the design work or make decisions.
- The interaction designers are not yet an integral part of the asset teams.

- The communication between interaction designer and the asset team is not yet happening as much as it should.
- There are some issues in cooperation with the sensorial team.

All the problems mentioned above seem to arise from at least one underlying issue: dividing working time with many teams is not the best choice when teams should work strongly together, as in scrum and in agile in general. It should be clarified to the designers and to the rest of the team that interaction designers should be an essential part of the scrum team and work in the same process with the team. That can't be forced - the feeling should come from the team, not from outside. The possibility to unify the team can however come from outside, the form can be e.g. co-location or at least giving a possibility to meet face-to-face regularly and use virtual meeting tools that allow everyone to see each other, not just hear.

5 Conclusions

The history and previous studies show clearly that waterfall process model does not work in the software development industry. The need for changes trashes the waterfall method completely and the development side has already created agile methods as a working solution for the need of change. However, when different agile methods were created, design was not fully considered. This has left interaction designers hanging without any support from process side. The need for interaction designers is not gone; in fact it has even grown because software is changing to be even more complex than ever. Software development needs a process where interaction design is combined into the existing, already used agile methods. The studies have shown that it can be done, but it needs time and effort. In software development industry, every project is unique; so one process model does not work for all. Every project should create their own process, but when creating one, lessons should be learned from others who have already defined a process model and used it to see what works and what not.

Not many descriptions exist of the combinations of interaction design and agile, while those would be important for those who are trying to figure out what kind of things they should take into account when creating their own process model. The case study part of this thesis answers to the need by giving a description of the process model which is used by one Nokia Devices R&D project, called project A in this thesis. Project A uses scrum method as a basis for the process, but has done some modifications. The most significant differences related to the interaction design include the interaction designer's specialized role in scrum team and the separation of design tasks in the product backlog so that the design can be done ahead of the implementation. The case study also includes the results of an Internet survey, which was conducted to find out which parts of the process were not going well. The survey results show that while interaction designers don't have big issues with the process model, they are too busy to fully concentrate on the design work and they feel that they are not yet part of the scrum teams. The divided working time between many asset teams seems to cause issues, but it might get easier when the teams have worked together longer and changed some

parts of the process to suit them better. The case study was done in the early phase of the project A and it would be interesting to redo the survey when the project has been going on for a while. In the meantime it is important to focus on the issues that came up in the survey results. As mentioned before, in software development industry every project is much different from each other, so the survey results can't be generalized.

While agile methods might seem threatening from interaction designer's point of view, they are proved to work much better than waterfall process model. Agile methods are giving something also for interaction designers, there are many small positive aspects and there exists a promise of one bigger one also: with agile it is possible that the final product will actually be what interaction designer intended it to be.

References

Armitage, John. 2004. *Are Agile Methods Good for Design?* Interactions, January-February 2004, pages 14-23.

Cockburn, Alistair. 2007. *Agile Software Development: The Cooperative Game.* 2. edition. USA: Addison-Wesley Professional.

Cohn, Mike. 2004. *User Stories Applied For Agile Software Development.* 1. edition. USA: Addison-Wesley Professional.

Dijkstra, Edsger W. 1972. *The Humble Programmer*. Communications of the ACM, Volume 15, Issue 10. Pages 859-866. USA: ACM.
[Source: http://portal.acm.org/citation.cfm?id=361591]

Garrett, Jesse James. 2003. *The Elements of User Experience: User-Centered Design for the Web*. 1. edition. USA: New Riders.

Jarzombek, Stanley J. 1999. *The 5th Annual Joint Aerospace Weapons Systems Support, Sensors, and Simulation Symposium (JAWS S3) Proceedings*. USA: United States Government Printing Office.
[Source: Larman 2004 and
http://www.stsc.hill.af.mil/crossTalk/2002/11/gilb.html]

Larman, Graig. 2004. *Agile & Iterative Development: A Manager's Guide*. 1. edition. USA: Addison-Wesley Professional.

Leffingwell, Dean. 2007. *Scaling Software Agility: Best Practices for Large Enterprises*. 1. edition. USA: Addison-Wesley Professional.

Löwgren, Jonas and Stolterman, Erik. 2004. *Thoughtful Interaction Design: a Design Perspective in Information technology*. 1. edition. USA: The MIT Press.

McInerney, Paul and Maurer, Frank. 2005. *UCD in Agile Projects: Dream Team or Odd Couple?* Interactions, November-December 2005, pages 19-23.

Nielsen, Jakob and Nodder, Chris. 2008. *Agile Usability: Best Practices for User Experience on Agile Development Projects*. USA: Nielsen Norman Group. [Source: http://www.nngroup.com/reports/agile/]

Saffer, Dan. 2007. *Designing for Interactions. Creating Smart Applications and Clever Devices.* 1. edition. USA: New Riders.

Schwaber, Ken and Beedle, Mike. 2001. *Agile Software Development with Scrum.* 1. edition. USA: Prentice Hall.

Online

Agilemanifesto.org. 2001. *Manifesto for Agile Software Development*. [http://agilemanifesto.org/] (Retrieved on 4.10.2009)

Ambler, Scott W. *Agile Software Development: Definition*. [http://www.agilemodeling.com/essays/agileSoftwareDevelopment.htm] (Retrieved on 4.10.2009)

Bowles, Cennydd. 2008*. Getting Real About Agile Design.* [http://www.alistapart.com/articles/gettingrealaboutagiledesign] (Retrieved on 4.10.2009)

Giudice, Maria. 2008. *Can't we just all get along? Human-centered design meets agile.* [http://www.slideshare.net/mgiudice/humancentered-design-meets-agile-development-presentation-625465] (Retrieved on 4.10.2009)

IxDA. *What is interaction design?* [http://www.ixda.org/] (Retrieved on 4.10.2009)

Nokia. 2008. *Nokia in brief*. [http://www.nokia.com/NOKIA_COM_1/About_Nokia/Sidebars_new_concept/Nokia_in_brief/InBrief_08.pdf] (Retrieved on 4.10.2009)

Saffer, Dan. 2008. *The Disciplines of User Experience*. [http://www.kickerstudio.com/blog/2008/12/the-disciplines-of-user-experience/] (Retrieved on 4.10.2009)

Wikipedia. *Design*. [http://en.wikipedia.org/wiki/Design] (Retrieved on 4.10.2009)

Wikipedia. *History of software engineering*. [http://en.wikipedia.org/wiki/History_of_software_engineering] (Retrieved on 4.10.2009)

Wikipedia. *Interaction design*. [http://en.wikipedia.org/wiki/Interaction_design] (Retrieved on 4.10.2009)

Wikipedia. *Waterfall model*. [http://en.wikipedia.org/wiki/Waterfall_model] (Retrieved on 4.10.2009)

Appendices

The thesis includes following appendices:

Appendix 1: Internet survey

The whole survey, all questions and instructions included.

Appendix 2: Survey comments

All comments received from the free text comment fields in the survey.

Internet survey

Page 1/8: IxD & Agile Survey

This survey is about interaction designers in agile process. Results will be used in thesis work as well as in our team's process work. Intention is to gather data about the current situation, how well the process is going right now and what are the areas that should be improved.
Your personal information is not asked or used anywhere and the results are stored safely in the Nokia server.
Instructions
Scrum process means our modified scrum process, not traditional scrum.
Scrum team means the scrum team you are mostly working with. It is important that you are answering the questions so that you think about only one team, even though you would be working in several teams.

Page 2/8: Background

		Yes	No
1.	Is your implementation team using scrum process?		
2.	Are you working as a part of the scrum process?		
3.	Are you working as a part of several implementation teams?		
4.	Are you co-located with your scrum team?		

Page 3/8: Concept creation, practices, time

1 = Disagree strongly
2 = Disagree moderately
3 = Disagree
4 = Neither agree nor disagree
5 = Agree

6 = Agree moderately
7 = Agree strongly

		Don't know	1	2	3	4	5	6	7
1.	I had enough time to clarify the concept.								
2.	I have had a possibility to explore alternative designs.								
3.	I have enough time to be in the necessary meetings.								
4.	I can concentrate on the design work enough.								
5.	Sometimes I feel that I have to do too quick decisions.								
6.	I can explore and investigate things when I feel that I need to.								
7.	My scrum team has done the needed adjustments to the scrum practices so that I can work efficiently.								
8.	Comments about the topics on this page.	[Free text]							

Page 4/8: Roles and responsibilities

1 = Disagree strongly
2 = Disagree moderately
3 = Disagree
4 = Neither agree nor disagree
5 = Agree
6 = Agree moderately
7 = Agree strongly

		Don't know	1	2	3	4	5	6	7

1.	The scrum team understands my role as a specialist, even when it is against scrum principles.								
2.	I feel that I am really part of the scrum team.								
3.	The design is a common goal for the whole scrum team.								
4.	The scrum team is asking relevant questions from me, not just doing something.								
5.	I am responsible of the design in my scrum team.								
6.	Comments about the topics on this page.	[Free text]							

Page 5/8: Communication, feedback, cooperation

1 = Disagree strongly
2 = Disagree moderately
3 = Disagree
4 = Neither agree nor disagree
5 = Agree
6 = Agree moderately
7 = Agree strongly

		Don't know	1	2	3	4	5	6	7
1.	I got enough support from the scrum team to concept creation.								
2.	I get enough feedback and support for the design work from								
	scrum team.								
	other interaction designers.								

3.	It is easy to ask questions from scrum team members when needed.								
4.	Specification is my way to communicate the design to scrum team.								
5.	I describe the design to the developer(s) before implementation starts.								
6.	Cooperation with								
	sensorial team goes well.								
	usability team goes well.								
7.	Comments about the topics on this page.	[Free text]							

Page 6/8: Agile, scrum, process, constant change

1 = Disagree strongly
2 = Disagree moderately
3 = Disagree
4 = Neither agree nor disagree
5 = Agree
6 = Agree moderately
7 = Agree strongly

		Don't know	1	2	3	4	5	6	7
1.	I have enough knowledge of our current scrum process model.								
2.	I have had enough possibilities to be involved to the user story / product backlog creation.								
3.	My scrum team is confident about working in the current process model.								

4.	I like the fact that anything in the design can still change later on.								
5.	The scrum team accepts design changes.								
6.	I think that the design changes are a way to a better design.								
7.	Comments about the topics on this page.	[Free text]							

Page 7/8: Comments

1.	**Please write** **- comments** **- concerns** **- problems** **- good things** **- lessons learned** **- etc** **about the current process here, if you haven't written those before.**	**[Free text]**

Page 8/8: Submit

You can return to the question pages by selecting the page numbers below. The number in brackets shows how many unanswered questions there are on that page. After reviewing, return and click Submit Survey.
Page 2: (x) **Page 3: (x)** **Page 4: (x)** **Page 5: (x)** **Page 6: (x)** **Page 7: (x)**
The survey is complete.

If you are satisfied with your answers, please click the Submit Survey button.

[Submit]

Survey comments

Comments were first asked in each actual question page and then in separate comments page. In here the comments are separated under survey page titles.

Page 3/8: Concept creation, practices, time

“Decisions are sometimes too quick, but that's always the case, not just in scrum.”

“Generally there is enough time to clarify concept, but not enough to cope with all the dependencies if other projects are delayed.”

“I would like to discuss with developers and test engineers what is exactly needed in UI spec every sprint otherwise the spec will be unnecessary long for them. UI designers can spend rest of time for other tasks such as user test and UI framework.”

Page 4/8: Roles and responsibilities

”I don't have good enough visibility to the development to be completely sure of question 4.” (Question 4: The scrum team is asking relevant questions from me, not just doing something.)

Page 5/8: Communication, feedback, cooperation

“The problems with the sensorial team are mainly results of the lack of resources in the graphical design front, no problems in actual relationships between people. Also there is have been a few communication issues which hopefully are solved when we are soon part of the same team.”

“Specification is not the way to communicate, and should not be. Face to face communication and quick drawings works so much more efficiently. Testers might need and want specification, but implementation does not need it nor want it.”

“Ui specification is quite cumbersome when discussing or presenting the design to the team. Simple pictures and slides work much better.”

Page 6/8: Agile, scrum, process, constant change

”Sometimes changes in design are a sign of inability of strong decisions, but mainly of course changes are made towards a better design.”

Page 7/8: Comments

“We need more support from our senor staff, for example, making decisions.”

“After trying scrum, I'd not like to work in traditional waterfall fashion. Scrum is a demanding way to work, sometimes very stressful, but when it works it's very direct and rewarding way to work. Physical co-location is absolutely important though, no chance to make a 'tele-scrum' team work.”

“Our Agile UI process does not completely fit the process our asset project is using. Small modifications have been enough to get things done.”

“We have just run to agile without understanding how to create the whole UI from scratch. I feel the biggest flaws and misunderstandings have been caused because especially new developers do not understand how to make UI design in scrum. They just want to start coding and assume that the whole design can be done in sprints. Better communication of processes from Quality & Process team would have helped a lot.”

“At the moment, I don't have sufficient time to write UI specifications. Material for User test and interaction communications between other application designers takes a lot of my time as many of my design decision affect other applications.