HELSINKI UNIVERSITY OF TECHNOLOGY
Faculty of Information and Natural Sciences
Degree Programme in Information Networks

Niina Huttunen

Agile User-Centered Software Design Process for Outotec

Master's Thesis

Espoo, August 31st, 2009

Supervisor: Professor Marko Nieminen, D.Sc. (Tech.)

Instructor: Lauri Repokari, Lic.Sc. (Tech.)

HELSINKI UNIVERSITY OF TECHNOLOGY
Faculty of Information and Natural Sciences
Degree Programme in Information Networks

ABSTRACT OF MASTER'S THESIS

Author:	Niina Huttunen		
Titel:	Agile User-Centered Software Design Process for Outotec		
Professorship:	T-121 User Interfaces and Usability	**Date:**	August 31st, 2009
Supervisor:	Marko Nieminen, D.Sc. (Tech.)	**Language:**	English
Instructor:	Lauri Repokari, Lic.Sc. (Tech.)	**Number of Pages:**	5 + 106 + 21

This thesis considers development of software design process as a case study. The result is an agile user-centered software design process model. The research is done by following principles of design science in organizational development. In design science researcher develops knowledge which professionals can use to solve the problems in their field.

The target company wanted new software design process to be agile and user-centered. Agile software development methods are created to solve typical problems on software development projects. Slipping schedules, growing budgets and poor quality are challenged by increasing communication and collaboration and by decreasing planning and documentation. On the other hand, user-centered design is the solution for creating usable and pleasant products. These two design and development methods can be combined even though they differ from philosophy. One option is to make user information collection, user interface design, implementation and prototype testing parallel in cycles which are in different phases.

Combining agility and user-centered design was the starting point for the research. After that process end-users described current situation and their hopes and needs in interviews. They also analyzed interview results and designed solutions for recognized problems in co-design workshop. Finally they evaluated preliminary version of the process model which was based on the user needs in evaluation workshop. In other words, their continuous input was in essential role in all the phases of process design.

As a result of the research there is presented a new agile user-centered software design process model for the target organization. As a support for the model there are presented current situation of software development practices and process users' needs for process model. However, since the change would be too big as such there is also presented a step model for increasing agility and user-centered approach gradually. The next step is to research how this model, and also process models created with end-users in general, succeed in practice.

Keywords: user-centered design, agile software development, design science, process design

TEKNILLINEN KORKEAKOULU DIPLOMITYÖN TIIVISTELMÄ
Informaatio- ja luonnontieteiden tiedekunta
Informaatioverkostojen koulutusohjelma

$(-+,':	Niina Huttunen		
$820 0+ / +:	Agile User-Centered Software Design Process for Outotec		
"31)(44663+:	T-121 Käytettävyys ja käyttöliittymät	**"'+7'84:**	31.8.2009
%&.71,&:	Marko Nieminen, TkT	**+(.+:**	Englanti
! *,&&,&:	Lauri Repokari, TkL	**#+76 / ''3':**	5 + 106 + 21

Tässä diplomityössä kehitetään ohjelmistosuunnittelumalli asiakasorganisaatiolle. Tuloksena esitellään ketterä ja käyttäjäkeskeinen ohjelmistosuunnittelumalli. Tutkimuksen viitekehyksenä on sovellettu suunnittelutiedettä (engl. *design science*), jota voidaan yleisesti käyttää organisaatioiden kehityksessä. Suunnittelutieteessä tutkija kehittää alaan liittyvää tietoa, jota asiantuntijat voivat käyttää ratkaistessaan ongelmia kyseisellä alalla.

Kohdeyritys halusi tulevan mallin olevan ketterä ja käyttäjäkeskeinen. Ketterät ohjelmistokehitysmetodit on kehitetty ratkaisemaan tyypillisiä ohjelmistokehityksessä esiintyviä ongelmia. Liukuvia aikatauluja, ylitettyjä budjetteja ja huonoa laatua yritetään välttää lisäämällä kommunikointia ja yhteistyötä sekä vähentämällä suunnittelua ja dokumentointia. Käyttäjäkeskeisellä kehityksellä on puolestaan olennainen rooli käytettävien tuotteiden suunnittelussa. Nämä kaksi suunnittelu- ja toteutustapaa voidaan yhdistää keskenään, vaikka niillä onkin eroavaisuuksia. Yksi vaihtoehto on toteuttaa käyttäjätiedon kerääminen, käyttöliittymäsuunnittelu, ohjelmiston toteutus sekä prototyypin testaaminen rinnakkaisissa sykleissä, jotka ovat kukin eri vaiheissa.

Ketterän kehityksen ja käyttäjäkeskeisyyden yhdistäminen oli lähtökohta diplomityölle. Empiirisessä tutkimuksessa haastateltiin organisaation työntekijöitä eli prosessin tulevia käyttäjiä. Haastatteluissa selvitettiin organisaation nykyisiä käytäntöjä sekä tarpeita ja toiveita tulevalle mallille. Lisäksi tulevat käyttäjät analysoivat itsekin haastatteluaineistoa ja suunnittelivat ratkaisuja haastatteluissa havaittuihin ongelmiin. Loppuvaiheessa he arvioivat ja kommentoivat prosessimallin alustavaa versiota, joka oli suunniteltu käyttäjätarpeiden pohjalta. Toisin sanoen prosessin tulevilla käyttäjillä oli olennainen rooli prosessin suunnittelussa.

Tämän tutkimuksen tulos on kohdeyritykselle tarkoitettu ketterä ja käyttäjäkeskeinen ohjelmistosuunnittelumalli. Mallin tukena esitellään tämänhetkiset ohjelmistokehityskäytännöt kohdeyrityksessä sekä yrityksen työntekijöiden kehitystarpeet ja -ideat liittyen käytäntöihin. Mallin tuoma muutos saattaisi olla sellaisenaan kuitenkin liian iso. Tämän takia kohdeyritykselle on esitelty myös porrasmalli, jonka avulla ketteryyttä ja käyttäjäkeskeisyyttä voi lisätä asteittain ohjelmistosuunnittelussa. Seuraavaksi on tärkeää tutkia, kuinka diplomityössä kehitetty ohjelmistosuunnittelumalli toimii käytännössä.

7&+04&0&5: käyttäjäkeskeinen suunnittelu, ketterä ohjelmistokehitys, suunnittelutiede, prosessin suunnittelu

Preface

This master's thesis would have not been possible without the effort of my present employer Design Agency Fusion Ltd. My careful wish was to start do the thesis in the beginning of the year 2009. I think it was something in the mid of February when I started the literature review. I was also asked what I would like to research even though it was clear from the beginning that this thesis will be done as a customer project. I made some suggestions. One was that I would like to something new: apply my knowledge about user-centered design with some new approach. That wish was definitively fulfilled and it made the thesis extremely interesting.

I was very lucky that the customer was found very quickly, thanks to Ville Kolehmainen from Fusion who made a great job in finding me a theme for the thesis. I want to thank Outotec Oyj for providing me the chance to do the research. Even more grateful I am for the smooth co-operation, especially with Matti Luukkonen and Ari Rantala. When I asked for a discussion meeting you always had time. When I decided to change the schedule of the interviews, I got 19 interviews with very short notice scheduled exactly to those weeks I proposed. I am also very happy that so many people showed interest by participating into workshops. This is possibly one of the rare thesis' which did not have any obstacles during the process. To be honest, whole process has been even too easy. Usually people are very happy and relieved after finishing their thesis. Due to positive experience I feel like I could still continue the research.

Multiple persons have given input for this thesis. I want to thank Professor Marko Nieminen for supporting my ideas and giving new insights in multiple discussions. You also had time every time I needed of which I am very grateful. Thanks for my instructor Lauri Repokari for giving valuable comments about structure and content. Once more I want to thank Matti Luukkonen, Ari Rantala, Ville Kolehmainen and Jarkko Malviniemi (Fusion) for supporting this work. Thanks for my parents who has encouraged my education during my life. Finally I want to thank my fiancé who is a great listener, good proofreader and who has participated in the simultaneous wedding organization project actively. I am very happy to merry you on 29th August 2009.

In Espoo, 27st August 2009

Niina Varonen

Table of Contents

1. Introduction

1.1. Background

Agile manifesto presenting the philosophy of agile methodologies was published in the beginning of 2000s (Agile manifesto 2002). Subsequently agile software development methods have caused a huge impact on software engineering (Dybå and Dingsoyr 2008: 834). However, they are gaining popularity in industry although they comprise a mix of accepted and controversial software engineering practices (Lindvall et al. 2002). This might be the consequence of agile methodologies' background: agile methodologies are developed to provide companies with solutions for challenges like slipping schedules, growing budgets and poor quality (Martin 2002). In contrast, user-centered design originates already to 1980s and it became widely used after publication of *User-Centered System Design: New Perspectives on Human-Computer Interaction* by Norman & Draper in 1986 (Abras, Maloney-Krichmar and Preece 2004). Nowadays user-centered design is widely considered as the key to product usefulness, usability and an effective approach to overcoming the limitations of traditional system-centered design (Mao, Vredenburg, Smith and Carey 2005).

A Finnish company called Outotec Oyj providing process solutions, technologies and services for the mining and metallurgical industries decided to start using agile software development methods. They also wanted to improve the usability of their products by implementing user-centered design. They wanted some expert to develop them an agile user-centered software design process model. This model development was done as a master's thesis by an external consultancy work.

There is an active R&D department at Outotec Oyj. It constantly develops new technologies and processes to increase the competitiveness of metal production processes (Outotec 2009). That requires also software development. They build user interfaces to be used in monitoring rooms and process driving. Those user interfaces can be located in the physical device or then the devices can be used via wireless connection. In that case interface can be for instance in normal web browser. However, these user interfaces are usually done by automation and software engineers who are also making the software and hardware. In addition, user interfaces do not

have any consistent outlook – every software engineer makes it to look like they see fit. Usually one developer is designing and developing software but not following any consistent software development process. Again here, everybody is doing as they see fit by themselves. End-users are practically never involved in the design. That is why Outotec Oyj has realized that there is a strong need to change standard of activities.

In general Outotec Oyj provides globally process solutions, technologies and services for the mining and metallurgical industries. It employs 2674 persons in (2008) on every continent. There are offices in 21 countries. Outotec Oyj has three business divisions: Minerals Processing, Base Metals and Metals Processing. Their technologies and services cover the whole production chain of processing minerals to metals. They are also used in the production of ferrous metals and ferroalloys, alumina and aluminum, copper, nickel, zinc, precious metals, niobium, synthetic rutile, certain industrial minerals and sulfuric acid. (Outotec 2009).

1.2. Research Aim and Research Questions

Outotec Oyj, which is referred as the target company in this thesis, decided to standardize their software and user interface development. Target was to develop a design process which can be followed in projects. In addition, the goal was to start using agile methodologies and make user interfaces more user-friendly. In other words there was a need for a software design process which is agile and user-centered. Model should assist the target company employees to design software and user interfaces considering agile and user-centered approach. The aim of this research is to find out what kind of agile and user-centered software design process model is suitable for the target company. This constitutes the main research question:

1. **What kind of agile and user-centered software design process model is suitable the target company?**

The level of agility and user-centeredness is specified along the research since it depends on the present state of the organization and employees' needs.

Since process model should be suitable for the target company, the research needs to focus on the target company instead of generalities. The target company employees, in other words

prospective process model users, are the most reliable source to discuss about their company and practices. Presumably they have development targets and ideas how to do software design better. That is why the process design is done by following principles of user-centered design. User-centered design considers user needs and requirements. First sub question is based on finding out user needs and requirements for the process model.

1.1. What kind of development targets and needs the target company employees have considering the software and user interface development?

Before improvement actions are started it is useful to study the present state in the target company first. First of all, researcher gets a good insight on earlier software development practices. Secondly, researcher familiarizes herself with use context of the process model. Present state in the target organization gives the researcher framework and starting point for improvement actions. Exploration of the present state creates the second sub question.

1.2. How software and user interface development is done in the target company at the moment?

Agile and user-centered approach of the software design was the starting point for this thesis. In order to develop an agile and user-centered software process model it is essential to have knowledge about those issues. This knowledge is gathered in the literature review. Target in the literature review is to find out what agile user-centered software design process is. In order to explain that it is important to know what agile software design process is and what user-centered design process is. Those three issues are discussed in literature review.

The biggest benefit for the target company is achieved with the answers to the research questions 1 and 1.1. With answer to the question 1 the target company is provided with a new agile user-centered process model. This was the hope of the target company since they wanted to start develop software and user interfaces with more agile and more user-friendly practices. With answer to question 1.1 the target company receives information of development targets considering software and user interface design. These development targets are constructed based on the organization and its employees' needs and they can be utilized also for other

purposes than this research. Some of targets were not considered in this research so they are issues for further researches.

This research does not discuss software implementation practices from the technology point of view. For instance different implementation methods are not compared to each other and tools or technologies are not even discussed. Instead, research concentrates on the software design process in general. Different roles, their tasks and co-operation are researched.

Project planning, documentation and software testing are part of the software design and development process. However, they were excluded from the research. During the empirical research it transpired that documentation and testing as such are too big issues to handle in this research. They require more specific background studies both so they are themes for separate researches. Naturally project planning can be done with the help of the process model. However, to guarantee consistent project planning there should be more instructions and project plan template. Instructions and template construction requires also a separate project.

1.3. Research approach

Literature review consists of three parts related to agile and user-centered software design. They are agile software development, user-centered design and their combination agile user-centered software design. Empirical research in this thesis was done by following principles of user-centered design. In other words, user-centered design also acted as empirical research framework. In general this research is positioned under design science since the target was to produce knowledge which is used to produce design solutions to field problems. According to van Aken (2005: 387), the mission of a design science is to develop knowledge, which the professionals of that discipline can use to design solutions for the problems in their field. Design science has also been used for organizational development (van Aken 2007). According to van Aken (2007) organizational change can be carried out with three stage process model in which organization employees participate in planning the chance. This organizational change improvement framework is applied in this research. Design science for organizational development supports user-centered design methods for developing the knowledge used in designing. Structure and content of this thesis is illustrated in figure 1.

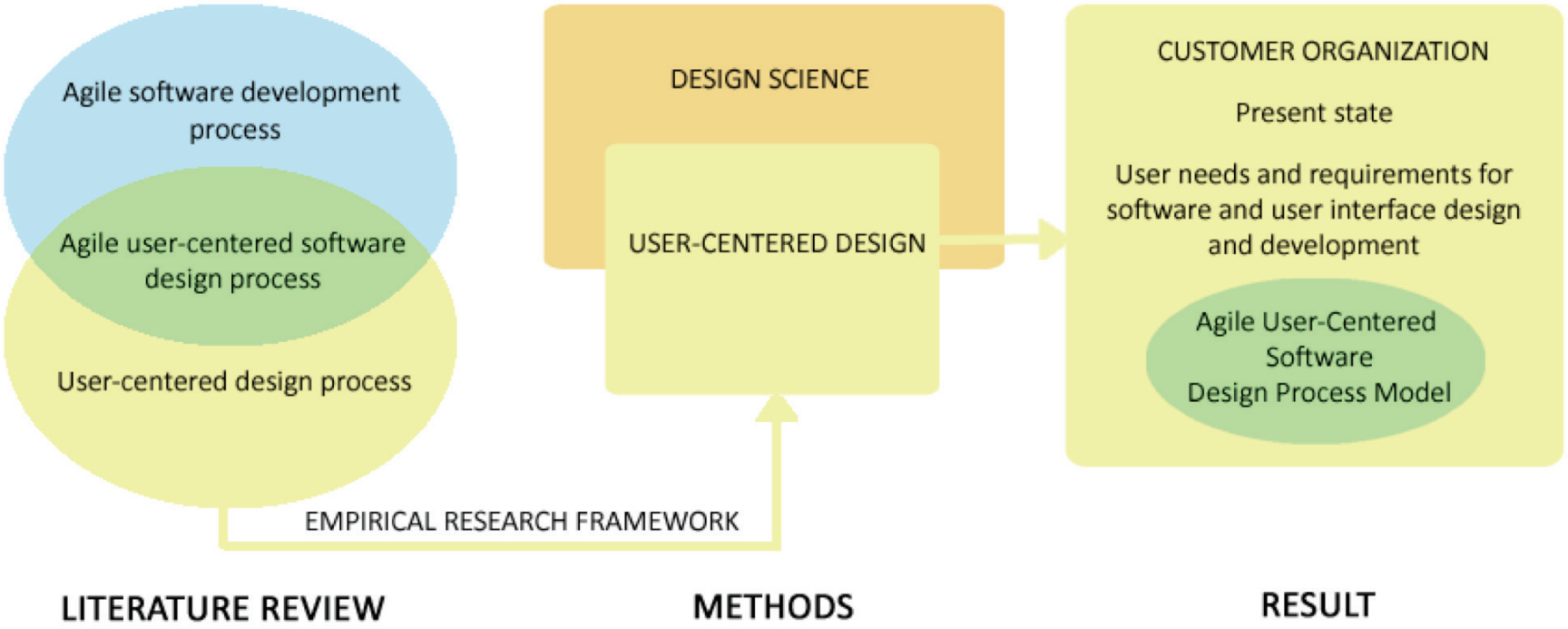


Figure 1 Master's thesis work structure

User-centered design methods used in the research were interview, co-design workshop and prototype evaluation. The target of the interview was to investigate the target company employees' needs and hopes. In addition to needs and hopes, also present state in the organization was studied. In a way, present state can be regarded as use context. Current software design and development practices and organizational factors like organizational culture describe quite well organization's present state from the software development point of view. Co-design workshop and prototype evaluation session provided the researcher with ideas and opinions about preliminary process models. Knowledge gathered during the user-centered design process was used in the design of the agile user-centered software design process model.

This master's thesis uses the term *software design* to describe the whole process from preliminary idea or need to the completed product alias software. Since it contains also the implementation phase it could be reasonable to speak about *software development* process. As a similar example product *development* process includes the whole process from tiny idea to ready made product. However, this thesis uses *software design* since focus is not on the implementation. Software implementation from technological point of view is beyond the scope of this master's thesis. No technologies, tools, programming languages or even programming practices are commented. In this case, software development would have referred too much to implementation phase. Instead of, *software design* includes also implementation phase since design and development are done in parallel in agile development. In addition, the term *design* reflects more about user-centered design and user interface design which are in relevant role in this master's thesis.

2. Literature Review

2.1. Agile Software Development

Agile methods are lightweight software development methods which consist of short iterative cycles, involving customers to establish, prioritize and verify requirements and prefer team knowledge instead of paper documentation (Boehm & Turner 2004: 32). Earlier software development methods like XP, SCRUM, Crystal and Feature Driven Development (FDD) were called lightweight because they have only a few rules and practices which are easy to follow. Later, due to the negative connotation of the term lightweight, these methods have been called as agile (Constantine 2002: 2).

2.1.1. Philosophy of Agile

Many of software development projects have failed due to lack of practices to guide it (Martin, 2002: 3). Slipping schedules, growing budgets and poor quality are normal to make customers and developers desperate. Agile software development gives one possible solution for improving software projects.

Essential story of agile software development started in February 2001 when a group of lightweight software development leaders, even competitors for each other, gathered together to find out alternatives to documentation driven, heavyweight software development processes. They named themselves as *Agile Alliance* and created *Manifesto for Agile Software Development.* (Agile Manifesto 2001). Agile manifesto is based on twelve principles describing targets and goals of agile software development (attachment 1). 17 members of Agile Alliance have subscribed the following manifesto.

> 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. (Agile Manifesto 2001).

Individuals and interactions over processes and tools mean that people are the most important ingredient of success. The project can fail if internal communication or team work is not working, regardless of the how good and expensive working tools are provided for the workers. Also a good process will not save from the failure if the members in the team are not strong players. However, team does not need excellent programmers. A strong player in a team might be an average programmer working well with others by communicating and interacting. Teams consisting of average programmers communicating together are more likely to succeed than a group of strong experts without any communication. Building a good team is much more important than building good facilities for the physical working environment. (Martin 2002: 4-5).

Working software over comprehensive documentation means that documentation should not be intentionally avoided but documentation should be kept as minimum as possible. For example new members in the team are not briefed into the project by documents. Instead of, information transferring is done with code and communication in the team. Code tells exactly what has been done so far. In addition, when people are sitting together they make newcomers part of the team through close training and interaction. Documentation should be used for communicating the rationale of design decisions and structure of the system but all the insignificant details should be left out. (Martin 2002: 5).

Customer collaboration over contract negotiation objects the traditional way of producing customer satisfaction. Customer telling his requirements, paying the money and then waiting for the result in a certain schedule is a chain which often fails in software development. Instead, customer should be included into the development process without making any strict up-front contracts about schedule and software content before the actual process. Software should be developed step by step and after every step output should be evaluated with the customer to guarantee the right direction. If direction is wrong then it needs to be changed. (Martin 2002: 5-6).

Responding to change over following a plan means that projects should be able to react to the changes. Both business environment and customer requirements are likely to change during the process and development. Ability to respond to change has a huge impact on the success and is strongly related to the planning of the project. Strict scheduled development plans do not guarantee any success since, besides dates, plans can undergo changes in shape. Project shape is changed every time some feature or task is dropped out or added in during the process. That is why detailed plans should be done for the next two weeks, very rough plans for the next three months and extremely crude plans after that. It is enough if developers know what they are doing during next two weeks and what the direction is after that, with only a vague idea about the system after a year. (Martin 2002: 6).

2.1.2. Agile Method

Chapter 2.1.1 presented philosophy behind the agile methodologies. However, it did not describe how agile works in practice and how the method is used in software companies. Use in practice is described with the help of two agile methods. Extreme programming, XP, is the most famous agile method (Martin 2002: 11) and it is described in detail. The second method SCRUM, presented in the following chapter, is also widely used and it has interestingly organized details. Since comparison of different agile methods with strengths, limitations and feasibility studies is beyond the scope of this thesis, this chapter only presents these two methods and their working in practice.

2.1.2.1. Extreme Programming

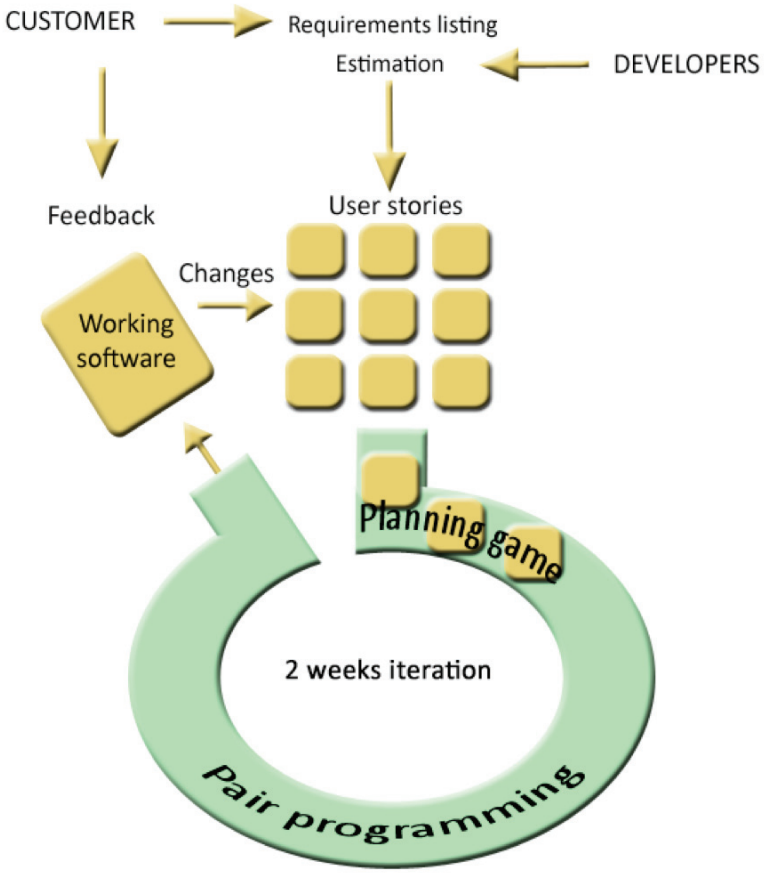


Figure 2 Extreme programming (XP)

Customer is in a big role in Extreme programming (XP) throughout the process (figure 2). XP defines customer as a person or a group who defines and prioritizes features for the software. This customer can be a representative of business analysts, marketing specialists, end-user or in fact a paying customer. In XP developers and customer work close each other as a team. Then they can communicate about the problems all the time during the process. They should work as a real team so working in the same building is recommendable. The best case is working in the same room. (Martin 2002: 12).

In XP up-front planning is replaced with rough *requirements listing*. In practice customer writes few words about each requirement on an index card. These requirements are discussed a bit more in detail in the team, and then developers give them an *estimation* about its development. After discussion and estimation requirements are called *user stories* which help to identify them once the requirement implementation is scheduled. Customer makes this scheduling based on user stories' priority and estimated cost. (Martin 2002: 12).

XP team believes that the specific details are likely to *change* and become more exact with time, especially when the system starts to come together from single features. That is why detailed requirements gathering long before implementation is regarded as premature focusing and a waste of time. (Martin 2002: 12).

Iterations in the XP project last two weeks. In the end of the each iteration or cycle *working software* is demonstrated and delivered for the customer to collect their feedback. Each iteration starts with *planning game* in which developers give customer a budget based on how much they were able to get done in the last iteration. Customer chooses user stories to be implemented during the iteration but do not exceed the budget. After the stories are decided and iteration has been started customer is not allowed to change the definition or priority of the stories in that iteration. Instead, developers can freely cut the stories up in to tasks and develop the tasks in the order that makes the most technical and business sense. (Martin 2002: 12).

Besides iteration plan for the following two week, also release plan is done for the next six iterations by the team. There is the same kind of a planning game session in which developers give budget based on the previous release and the customer decides stories and their development order under budget limits. However, releases are not cast in stone. Customer can cancel, add new or change story or its priority any time. (Martin 2002: 12-13).

Programming is done in *pairs*. Programmers work together at the same workstation. One programmer drives the keyboard and types the code while the other member of the pair watches the code being typed looking for errors and improvements. Pairs are changed at least once a day and inside the pair roles are changed frequently to avoid situations where the driver

gets tired or stuck. Since programmers work very closely with many kinds of specialists, pair programming increases the spread of knowledge through the team. (Martin 2002: 13).

2.1.2.2. Scrum

Scrum method consists of *Product Backlog, Sprint Backlog, 30 days Sprints* and every day status meetings called *Daily Scrum* (figure 3).

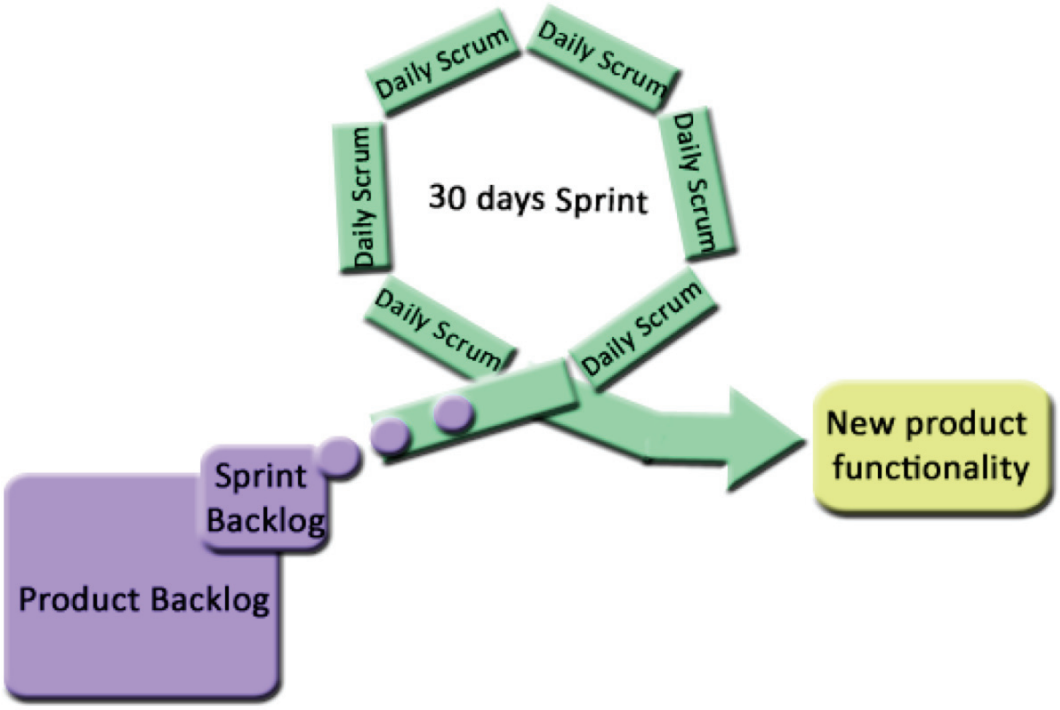


Figure 3 Scrum process

All the features, functionalities and technologies that system should use are listed in the Product Backlog. It is never finalized, instead of it evolves during the process. Content to the backlog can come from users, customers, sales, customer service and engineers but only a Product Owner can prioritize the backlog. Prioritizing decides the order in which the system is built and usually the desired features are prioritized high. (Schwaber & Beedle 2001: 7-10).

There can be many parallel *Scrum Teams* which are responsible for the development part. Each small and cross-functional team independently chooses from the Product Backlog as much work

as they think they can survive during the next 30 days sprint. Scrum teams are self-organizing and fully autonomous. It is up to team to decide how the product backlog will be turned into product. Every team has a Scrum Master who is representative of management. However, his task is to take care of the facilities. He enforce scrum practices, help in decision making, acquire resources as needed and remove possible impediments in progress. (Schwaber & Beedle 2001: 7-10).

Every day a team has a 15 minutes lasting status meeting called Daily Scrum in which their progress is reviewed and possible impediments are identified by the Scrum Master. In Daily Scrum, Scrum Master asks all team members to answer three questions: What have you done since last Scrum? What will you do between this moment and the next Scrum? What got in your way of doing work? (Schwaber & Beedle 2001: 7-10).

In the end of the sprint team will deliver *new* executable *product functionality*. Team gets together with management at a Sprint Review Meeting in which the product increment is inspected. Naturally there is a possibility that built product increment is useless if it does not satisfy stakeholders. However, thirty days Sprint duration ensures that the worst that can happen is the work lost only from those thirty days. After product increment inspection the management usually rearranges the Product Backlog based on the previous Scrum. Then Scrum Team again chooses tasks for the next iteration. These iterations are continued until the product is deemed potentially releasable. In the end of the process there are Release Sprints which prepare the product to release-readiness. (Schwaber & Beedle 2001: 7-10).

2.1.3. Benefits and Limitations of Agile Software Development

It would be easy to say that agile processes mitigate risks which use to occur in traditional software processes. To avoid slipping schedules, growing budget and poor quality software companies have changed from traditional software processes to agile methods. However, the trustful evidence for existence of benefits is lacking. Studies present that very few empirical research efforts have been conducted to study benefits of using agile methodologies (Erickson, Lyytinen & Siau 2005: 89) or the strength of evidence in these studies is low (Dybå & Dingsoyr 2008: 853). To enhance understanding of the benefits and limitations of agile software

processes more empirical data comparing the effectiveness and limitations of agile and non-agile approaches is needed (Turk, France & Rumbe 2002). Analytical studies based mainly on expert opinion or lessons learned and are not reliable enough for agile evaluation in this thesis. They were excluded from the following literature review.

Dybå and Dingsoyr (2008) studied what is currently known about the benefits and limitations of agile methods and the strength of evidence for those results. Out of 1996 they found 36 studies which had acceptable rigour, credibility and relevance (Dybå and Dingsoyr 2008: 853). Studies were divided into four thematic groups: introduction and adoption, human and social factors, perceptions of agile methods and comparative studies of agile and traditional methods. Results of benefits and limitations in each group are presented in table 1. However, table 1 should not be regarded as a guide for deciding whether the agile method is appropriate for some project or not. There are two reasons for that. Firstly, table 1 summarizes results from only one research (Dybå & Dingsoyr 2008). Unfortunately at the moment this study seems to be the most inclusive from the studies done during last years. More empirical studies are needed to make more comprehensive comparisons. Secondly in most of studies (76 %) Dybå and Dingsoyr (2008) researched development was done with XP and some of them were lacking appropriate recruitment strategy to ensure unbiased comparison. That is why these results should not be generalized until additional studies are done.

Table 1 Benefits and limitations of agile software development (Dybå and Dingsoyr 2008: 850)

	Benefits	**Limitations**
Introduction and adoption	In general agile development practices are easy to adopt and work well.	Difficult to introduce in a complex organization.
	Customer collaboration is emphasized.	Pair programming is seen inefficient and it is claimed to work best with experienced development teams.
	There exist work processes for handling defects.	Lack of attention to design and architectural issues.
	People are learning in pair programming.	
	Thinking ahead for management.	
	Engineers are focusing on current work.	
	Estimation is easier.	
Human and social factors	Thrive in radically different environments: from organizations having hierarchical structure to little or no central control.	Good interpersonal skills and trust are important characteristic for a successful XP team.
	Customer involvement and physical settings can vary and the process can still be successful.	
	Communication creates awareness within teams and organizations.	
	Developers in team have faith in their own abilities and preserve quality of working lives.	
	Big responsibilities, like a high level of individual autonomy, can be balanced successfully with high level of team autonomy and corporate responsibility.	
Perceptions of agile methods	Customers are satisfied with the opportunities for feedback and responding to changes.	The role of on-site customer can be stressful and cannot be sustained for a long period.
	Developers are satisfied with agile methods: their job and the end product.	Pair programming is regarded as exhausting practice due to heavy concentration.
		University students regarded pair programming difficult because of large skill differences between the members of the pairs.
Comparative studies	Changes are incorporated more easily and business value is demonstrated more efficiently by agile project management.	Team members are less interchangeable in agile teams which affect project management.
	Using XP increases productivity in terms of LOC/h.	Productivity can be just an image: sometimes developers believe that they are more productive and efficient just because they are using agile methods.
	It is possible to combine agile project management with overall traditional principles.	

Dybå and Dingsoyr (2008) recognized many benefits in researches they studied. Partly these benefits support also agile manifesto. Impact of communication was recognized in their study, and customers were pleased to have the chance to be involved in the process and response to the changes. Agile manifesto also highlights responding to the change. According to the research (Dybå and Dingsoyr 2008: 850) agile project management usually handles changes better compared to the traditional software development methods.

Nevertheless, also limitations of agile have been recognized during different studies researched by Dybå and Dingsoyr (2008). That is why agile should not be an undisputed target in software development. Every project's needs, resources and possible compatibility with agile should be considered carefully before any tools are decided.

Constantine (2002) has not done empirical studies about limitations and benefits of agile methodologies but he points out some interesting facts based on experience. Dybå and Dingsoyr (2008) mentioned that university students regarded pair programming difficult due to differences in skills. Constantine (2002: 4) highlights this by pointing out that agile methods work best with first-rate, versatile, disciplined developers who are highly-skilled and highly motivated. In addition, they need to have exceptional discipline willing to work with someone sitting beside them watching every move. Constantine (2002: 4) also mentions that it is hard to find good and skilled enough persons to lead the team.

2.2. User-Centered Design Process

User-centered design exists in this thesis in two contexts. Firstly, the result of this diploma thesis work, agile software design process model developed for the target company, will be user-centered. Secondly, the software design process model is designed following user-centered design principles.

The research result is an agile user-centered software design process model. It is user-centered since users will be involved in the process once some software is designed in the target company. In other words, end-product users will be part of the design process. Their opinions and needs are considered in the design and development.

The second context refers to empirical research. User-centered design principles are used as part of the theoretical framework for the empirical part of the thesis. In other words, users are involved in the process design. In that case user is referred as a person who will use this software design process in their future work. In other words users are process users. User-centered design framework is also utilized when data is analyzed and a process model is created iteratively.

2.2.1. ISO 13407 Process

There are couple of process models which are totally related to the user-centered design. Presumably ISO 13407, called Human centred design processes for interactive systems, is the most well-known and it is presented in this research. Other user-friendly models are for example Usability Lifecycle and Lucid. They both mention the importance of understanding users and they both create prototypes which are tested or evaluated before production (see more for example Nielsen 1993; Kreitzberg 1998).

ISO 13407 is a process model for designing systems which achieve high usability. User-centered design methods are incorporated throughout the system life cycle. Process model specifies activities which are performed during the development of an interactive system but it does not specify or recommend any particular method or technique to be used. However, these techniques and methods are presented closely in the next chapter.

Process starts with process design which defines needs for user-centered design. Actual iterative process consists of four phases

- Understanding and specifying the context of use
- Specifying the user and organizational requirements
- Producing design solutions
- Evaluating designs against user requirements (ISO 13407 1999)

Phases constitute iterations as presented in figure 4.

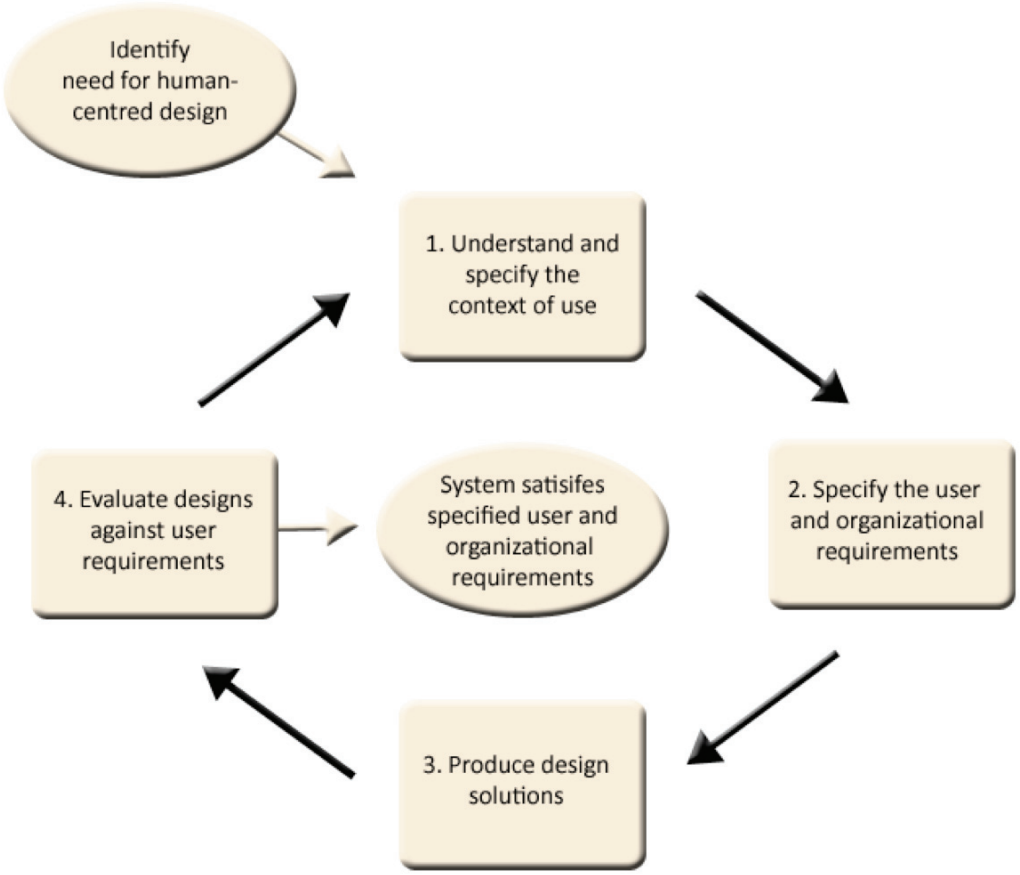


Figure 4 Human centered design processes for interactive systems (ISO 13407 1999)

Process starts with planning: what is done and how in order to involve users into design process. In the context specification (phase 1) the target is to find out who are the users, what kind of tasks they want to accomplish with the system and in what kind of physical and mental environment system is used. Physical and mental environment is regarded as context. (ISO 13407 1999).

In phase 2 outcome from the phase two is generated into user and organizational requirements. In phase 3 design solutions are generated considering the requirements created in the phase 2. (ISO 13407 1999).

In phase 4 designs are evaluated against user requirements specified in phase 2. If the system meets all the specified requirements, cycle ends and design can be produced (ISO 13407 1999). In computer-based systems this usually means that implementation phase can be started. However, it should not be forgotten that there can be different requirements in software development. User requirements should not be mixed up with functional and technical requirements which also need to be generated before the actual implementation.

If user requirements are not met with the produced design the context and user requirements should be considered once more. That way cycle starts again. (ISO 13407 1999).

2.2.2. User Information Collection

Designing high level usability and user experience requires clear understanding of the user, their needs and the use context of the product or the service. When designer has a good knowledge about the end-user and the use context they have a much better starting point for the design. After good design the end result will contain less expensive mistakes or problems.

User information is collected in user studies. In other words, user studies are done to find out user needs and study use context. In ISO 13407 user study covers phases 2 and 3 in which use context and user requirements are specified. User study reminds an ethnographic study in which researcher familiarizes themselves with the user, their habits and working or living context. However, unlike in ethnography, researcher rarely has an opportunity to spend weeks or months with the user. Instead they spend hours or days with user and ask questions like in an interview.

The method for performing user study is not specified. It can be observation, personal interview, focus group interview, (shadowing) observation, informal chat, questionnaire, artifact collection, probe or some kind of a mixture of those. The only requirement is that researcher gets a lot of valid information about the user and their life. Different user study methods are presented in table 2.

Table 2 *Basic user study methods*

Method	Description
Personal interview	Information about user, their work and their working environment is gathered just by talking with users. Also other people, like supervisors and managers, whose knowledge, attitudes and insights may be valuable in planning and designing your product, can be interviewed. (Hackos & Redish 1998: 135).
Focus group interview	Focus group is a group meeting in which there are typically 8-12 representative users (usually chosen to match previously identified demographics) and facilitator as a moderator. Facilitator is skilled in eliciting attitudes, opinions, preferences, and reactions to new product ideas. Facilitator also follows a script or a plan laying out the issues to be covered or the questions to be addressed in the group interview. It is important that no single person dominates the discussion. (Hackos & Redish 1998: 145-146; Cooper & Reimann 2003: 52).
Observation	User data can be collected by observing user while they are working. It is important to observe how the physical, social and cultural environments affect to their work. With observation the researcher wants to understand what user tries to do and what the user's goal is. (Hackos & Redish 1998: 139). However, usually this requires also questions and discussion. Contextual interview is presented later.
Shadowing	Observation can also be done as shadowing in which researcher closely follows a member of the organization (user) over an extended period of time. For instance, if the person being observed goes to another department or meeting, the researcher follows them. (McDonald 2005: 456).
Survey	Direct (e)mail questionnaires, telephone surveys and surveys conducted by fax or through the Web are broadcast techniques designed to gather information from a large group of users. Surveys can help to decide if circumstances observed at site visits are representative of a wider user population. However, conducting a survey requires good expertise in survey design and data analysis. Otherwise the result can be poor and useless. (Hackos & Redish 1998: 148).
Artifact collection	Researcher should collect artifacts from observations and interviews. When user shows, uses or talks about something, researcher should ask if the object can be given for them or photographed. An artifact might be for instance hand-done manual people created for them or forms and reports used or created during the process. In fact it can be almost anything related to research target. (Hackos & Redish 1998: 139).
Probe	Probes are explorative and design oriented self-documenting tools. They aim at understanding human phenomena and exploring design opportunities. Usually probe is some kind kit which contains different tasks which are designed to reveal users' perspectives. Users have a lot of time to do tasks and unlike in interviews, there is usually a possibility to return to tasks if something comes up later. (Mattelmäki, 2006).

User study methods can be combined. Beyer and Holtzblatt (1998) describe in detail user study method called contextual interview which is a mixture of the observation and interview. Method

is very simple: go where the user works, observe the user as she or he works and talk to the user about the work (Beyer & Holtzblatt 1998).

In practice contextual inquiry is usually carried out with the master-apprentice model. User is a master and teaches the researcher (apprentice) by doing work and talking about it while working. By talking he can describe exactly what he is doing and why. Apprentice is allowed to ask questions about something the master did if she needs details or clarifications. However, observation interspersed with discussion requires a bit extra effort from both master and apprentice. (Beyer & Holtzblatt 1998: 42-46).

One of the strengths of master-apprentice model is its naturalness. Usually people who have worked for a long time cannot describe their working in detail since they are not aware of everything they do. Some actions are the result of years of experience and have subtle motivations while others can just be habits. These are revealed best with real actions in real context. In master-apprentice model it does not matter if master is not a good teacher by nature. Master does not need to prepare any presentations or otherwise prepare for the situation. Instead of, everybody can talk what they do as it unfolds. In addition, when the work is right there even the details which people do not normally pay attention to are available for the study. (Beyer & Holtzblatt 1998: 42-46).

2.2.2.1. User study analysis

Typical way to analyze user study data is to construct an affinity diagram from data observations. Every interesting detail and observation from the interview, observation, artifact collection or probe is written into post-its. In general they are called observation regardless of the source. Observations are organized and grouped into reasonable hierarchical groups which help to constitute a big picture of the findings in the research. Grouping is done bottom-up by first figuring out which observations are related together and then inventing them a common topic. Subsequently observation groups with a topic are grouped together and given higher level topic. (Beyer & Holtzblatt 1998: 154-163).

User study results are usually constructed to user personas. One user persona describes one typical user with all interesting elements related to the research. For example, depending on the

focus of the research, sex, age, job, typical working day, interests, disabilities and hobbies can be described. In addition, user persona is usually given a name to create a feeling of the real person. However, user persona is always a fictive person. Its features are based on the real users who have been studied but it is never a description of only one real user. User personas are used in designing phase. Designers have something concrete to refer and real users stay always in mind with the help of the compact persona description. Due to concreteness, it is easier to design for 36 years old Joe with two teen-aged daughters than for a man with children. (About user personas see more for example Guenther 2006; Pruitt & Adlin 2006; Cooper 1999).

2.2.3. Designing With Users

User study is done before the design is started. However, users are not forgotten either after that. In contrast, end-users should be involved in the designing until the design phase is ready. They should participate either as evaluators of design mock-ups or even designers. In ISO 13407 Human-centred design process model designing with users can be part of the phase 3 (produce design solutions) and phase 4 (evaluate design solutions against user requirements).

After user study is done, designers prepare preliminary design of the user interface. Usually this preliminary design is an initial paper prototype or wireframe model which expresses only structure of the system without any concentration on precise layout, icons or decoration. Then users will give their feedback about the prototype. This feedback collection can be done as an informal discussion during system walkthrough with prototype or as a usability test. However, this kind of a usability test does not measure efficiency of user's performance. Instead, it is done in order to find out if the design is in the right track or if a better possible system structure exists. (See more Beyer&Holtzblatt 1998: 370-377).

Since prototyping is an iterative process, evaluation with users can be done after each iteration cycle if there are enough resources. Iterative evaluation does not need to be time-consuming process: it is possible to evaluate multiple prototypes and try out many different ideas during one week. One day designer prepares a prototype and next day they test it with users. Again next day after that designer makes a new version based on the users' feedback and so on. (Beyer & Holtzblatt 1998: 376-377).

Besides acting as an evaluator user can also participate in designing. Depending on the origin, this can be called as participatory design, co-design or collaborative design (see more Schuler & Namioka 1993). However, it should be noticed that co-design and collaborative refers also to other issues. Collaborative design may refer to computer-supported collaborative design, in other words to designing which is done with the help of computer systems, regardless of the physical distance of the designers (Kvan 2000: 409). As for hardware/software co-design means meeting system-level objectives by exploiting the synergism of hardware and software through their concurrent design (De Micheli & Gupta 1997: 349). In this thesis co-design is used to describe any kind of design action in which user is involved. Users and other stakeholders can participate in co-design workshops where the target is to be creative and push the design process forward by prospective users. Ideas created in the workshop do not necessarily need to be final features in the product. However, workshops give designers important information about user needs. That helps them in designing and users' ideas can act as design drivers or even become new innovations as such.

There are several methods which can be used in co-design. Beyer and Holtzblatt (1998: 376-377) have used prototypes as communication tools and to build trust with customers and stakeholders. According to them users are excited about prototype interviews since then they can see the progress, they can talk directly to developers about their ideas and hopes, and they can see how their responses shape the design. Real prototype of the system in co-design session enables pretending to do real work. When designers and users discuss about the system interaction along working, there will come out issues that would otherwise remain invisible. (Beyer & Holtzblatt 1998: 371-377).

If co-design session is organized in early stage of the design, prototype does not necessarily exist. Then users and stakeholders have to be motivated some other way. Traditional way is to use user personas which summaries results of the use study. Users are asked to create concepts for the personas. However, it is also useful to have some other materials to activate users. For example Vaajakallio and Mattelmäki (2007) have used "Make tools" to amplify ageing workers' creativity and to enable the enactment of use scenarios. Basic idea of make tools is to allow people to construct design representations through visual elements and at the same time express their needs (Vaajakallio & Mattelmäki 2007: 225). However, make tools can vary from

visual collages to three-dimensional artifacts (Vaajakallio & Mattelmäki 2007: 225). According to Sanders (2006), these kind of visual toolkits work as scaffolds for experiences that support the creativity of everyday people. She emphasizes that toolkits helps people to express their ideas and feelings.

Besides visual toolkits also design games are used. Brandt (2006:57) defines design game as a framework for organizing participation in participatory design projects. According to her, design game is a play with props following specific rules. There can be an element of competition between players but the target is not to win, instead, the target is to combine strengths of different players in order to be creative in design (Brandt 2006: 57-58). Design game can use different elements: collages make tools, cards, words etc. but practically elements, rules and target are always designed independently for different situations.

2.2.4. Use Context in Organizational Process Development

In product or software design use context is easy to define: physical and mental environment in which the product will be used. For example Schilit, Adams and Want (1994: 1) approach the definition of context with questions: where you are, who you are with, and what resources are nearby. Dey (2001) argues that definition to be too specific. According to him, context is all about the whole situation relevant to an application and its set of users. In addition, we cannot enumerate which aspects of all situations are important, as this will change from situation to situation (Dey 2001). As an example of context let's think about use context of diving bell. Software in diving bell will be used alone, under the water and pressure and with heavy diving equipments on and mask on the face. Roughly speaking, that is its using context.

However, it is not so simple to define the use context for more abstract things like organizational process model. Is the use context of the process model physical working environment, working atmosphere, company's rules and standards, organizational structure of the company or some kind of a mixture of these? In this thesis these all are referred as organizational factors.

Organizational factors affecting the software development process are studied very little. In this master's thesis literature search considering organizational or contextual factors was done very

profoundly. Searches in different scholarly data bases were done with different combinations of following entries: contextual, organizational, customer, factor, requirement, software development (process) and software design (process). In practice only one relevant study considering organizational factors affecting software development process was found. Also this study (Bern, Nikula, Pasi & Smolander 2007) supports the fact this topic has not been studied. According to Bern et al. (2007: 1) there have been multiple studies investigating failures in the software development projects and the reasons for that as a form of risk identification. However, they complain that these studies provide little help in deciding the practices which would best fit in different organizational and project setting. Instead, they state that these issues have not received much attention so far. Due to lack of studies exploring contextual factors affecting software development process, this thesis is justified to use only one reference in this context.

Bern et al. (2007) presents preliminary research results of the study exploring the organization context of the software development. In the study they had a hypothesis about the contextual factors affecting software process and then they investigated the impact of each factor with theme-based interviews. Initial results suggest that there are nine categories which include multiple contextual factors (figure 5). (Bern et al. 2007: 3-4).

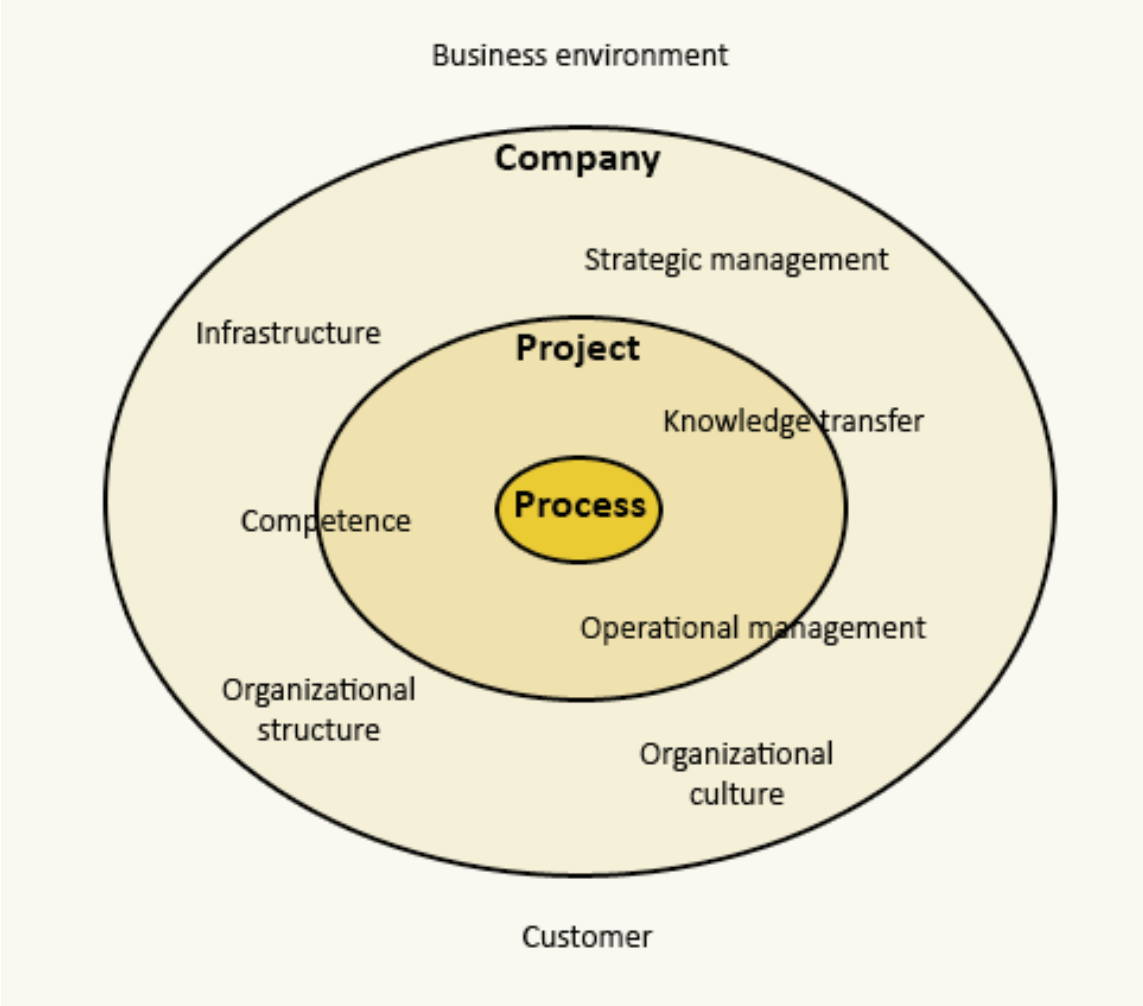


Figure 5 Contextual factor categories (based on Bern et al. 2007: 3-4)

Business environment is related to the operating environment of the company. It determines the amount of resources and costs to be allocated for different projects and how these projects are distributed across the organization and geographical locations. There are six contextual factors which are *level of competition, level of trust, attitude to subcontracting, cost level, language barriers and physical distance*. (Bern et al. 2007: 4).

Customer has a lot of influence on the entire software development process so it is probably the most important component in the business environment. A lot of problems caused by the customer were reported in the study. Contextual factors identified in customer category are *requirements clarity, level of involvement, customer orientation* and *level of customer management formalization.* (Bern et al. 2007: 3).

Strategic management deals with business strategy and the way it influences the software development practices. For example if the target is only to save money and new developers are not hired, the workers get tired and take longer sick leaves etc. There are two contextual factors: *business strategy* and *awareness of the strategy throughout the organization.* (Bern et al. 2007: 4).

Company infrastructure deals with methods, processes and tools used in the company. In general, the importance of following pre-defined processes was quite acknowledged in interviews. There are six factors in this category: *level of method and process formalization, selection of tools, amount of training available, level of change management formalization, availability of reusable artifacts* and *availability of common architecture.* (Bern et al. 2007: 4).

Organizational culture has multiple aspects but quality attitude were recognized to be one of the most important issues. Developers and testers in some companies were so afraid of mistakes that they did not deliver the product further until they were definitively sure that it follows the quality expected in the company. Four factors were observed in organizational culture: *attitude to process improvement, attitude to quality, work time policy* and *personal effect on working practices*. (Bern et al. 2007: 4).

Organizational structure refers strongly to organization of the company. In some companies same person has different roles inside the project and also participates in multiple projects. Also the knowledge of the software engineering practices and their consistency can vary between developers and managers. That complicates improvement actions. In other words there are two contextual factors: *number of roles* and *consistent understanding of software engineering practices*. (Bern et al. 2007: 4).

Competence is influenced by rare skills or lack knowledge of e.g. technology. Also problems in customer's application and business domains were regarded as troublesome issues. Four factors were identified in this category: *level of software engineering skills, working experience, level of business knowledge* and *level of domain knowledge.* (Bern et al. 2007: 4).

Operational management involves managing resources, schedules and projects. Problems were identified especially in the resource management area. This was related to organization structure category since working in multiple roles in multiple projects caused problems in projects. Four factors are identified in this category: *resource management approach, schedule management approach, project management approach* and *company quality assurance.* (Bern et al. 2007: 4).

Knowledge transfer refers to existence of documentation, their availability and use in projects and efficiency in communication. Two factors were identified: *efficiency of communication* and *level of documentation*. (Bern et al. 2007: 4).

These contextual factors related to the success of software development process will construct the use context of the process model. Like in the user-centered design, the use context of the prospective product was explored in this research. Organizational context was studied in order to create a process which fits in the organization. Data collection phase contains methods exploring these contextual factors in the target company. In practice these organizational factors were discussed with process users either directly or indirectly in the interviews.

2.3. Agile User-Centered Design Process

2.3.1. Agile Software Development and User-Centered Design

Agile methodologies and user-centered design have a lot in common. One of the biggest similarities is iterative development. Things are developed in smaller cycles and customer can evaluate and comment deliverables after each iteration. However, one of biggest difference is the customer. In user-centered design customer means end-user of the product – the person who is meant to use the product in their everyday life. Instead, agile software development wants feedback and evaluation from their real customer who is buying the product. In practice this means all the stakeholders related to the project. However, usually real end-users are not presenting themselves. Beyer, Holtzblatt & Baker (2004: 51) defines user as an individual who interacts with the system being designed directly while “customer” is larger term. According to

them customer may be a user or may need the output of the system, prepare input for a system, decide on the need for a system or approve the purchase of a system. In other words understanding the users is the key to getting the design right, while understanding the other customers of the system may be the key to getting it accepted (Beyer et al. 2004: 51). This study refers to user or end-user as the person who will concretely use the product in the future. Customer buys the product (probably sells it further), and may also be involved in the development process but still, they should not be the representative of the end-user.

McNeill (2006: 4) has listed similarities in agile software development and user-centered design. Besides **customer focus** and **iterative development,** also **test-driven development, collaboration** and **visibility** are common features for both methodologies. Testing is in an essential role after each phase or iteration. In user-centered design usability or user interaction tests or evaluations are conducted with end-users while in agile development developers run self-made tests for the code. Collaboration between customers, product managers, business analysts, developers and quality assurance maximizes overall team efficiency in both methodologies. Progress of the project is also visible for all the stakeholders in agile software development and in user-centered design. User-centered design actually uses outputs of the interaction design to communicate what the product will do. (Mc Neill 2006: 4).

Agile and user-centered design differs radically when it comes to philosophy. Strict up-front design is characteristic for user-centered design to guarantee the quality of user interfaces and to avoid troublesome and expensive fixing of usability problems afterwards. Instead, big up-front design is anathema for agile methods (Mc Neill 2006: 4). Experts of agile methods argue that it is useless to design something beforehand since it will anyway change during the project develops. They are said to see up-front design as a waste of time since business changes, business climate changes and customer requirement changes (Beyer et al. 2004: 55). However, user-centered designers see a huge risk on delivering a product that is inconsistent and confusing, if product is implemented by building discrete functional components to be stitched together as they evolve (Mc Neill 2006: 4).

Even though both methods are iterative, there are differences between durations of iteration cycles (Mc Neill 2006: 4; Ferreira, Noble & Biddle 2007). Iteration of code in agile methods lasts

typically weeks while using low-technology prototypes in user interface design takes just hours or maximum days.

Kane (2003) has compared discount usability engineering and agile development based on the principles of agile manifesto. Discount usability engineering is a method which allows sometimes expensive usability design methods to be used in more low-cost way by applying simple, low-cost techniques for design and testing (Nielsen 1993). Examples of such practices include scenarios, simplified thinking aloud, heuristic evaluation and card sorting. However, these methods cannot guarantee the most usable and best system, instead the target is to make the usability of the system good enough (Kane 2003: 1). According to Kane (2003: 3) there is one big difference in visions which reflects totally different philosophies: working software means different things for agile and discount usability engineering. In agile software development working software means that computers are able to verify the performance and capability of the software while in usability engineering working software is able to understand the interface between the system and the users. However, testing how software is working is much easier in usability engineering since users are flexible and able to test paper mockups. (Kane 2003: 3).

2.3.2. Advantages of Agile User-Centered Process

Like similarities suggest, the overall agile process can be very compatible with user-centered design process. However, combining requires a lot of effort and acceptance of different philosophies. As a result of common understanding agile developers produce products which will have fewer problems in use and that way, satisfy end-users better. For instance identifying potential problems early and reducing the need for rework at the end of the coding iteration can be done with the help of prototypes from user-centered design. Agile uses code as a prototype to collect feedback from the customer after iterations. If these processes are combined and concept of agile prototype is expanded to comprehend paper mock-ups and storyboards from user-centered design the cost-effectiveness can be even bigger. With help of paper prototypes and storyboards feedback is collected from the customer before any code is written in the iteration. (McNeill, 2006: 5).

Rapid testing and validation of story concepts before time consuming coding are not the only benefits of considering usability issues in agile development. McNeill (2006: 5) has listed also other advantages of combining user-centered design with agile. They are presented below.

Kane's (2003) study about potential usability gaps in agile development techniques support these findings.

Better understanding of the problem can be achieved when there is an end-user instead of the customer. Rich qualitative data captured from the user studies and participatory methods inform the story writing process better than head office view. (McNeill 2006: 5).

Clear, sociable visual representation of project vision is provided with the help of stories and storyboards. Ambiguous documentation and demanding requirements eliciting can be avoided with visual representation of the end goal. In addition, once end-users are shown storyboards and asked to complete simple processes the time to complete the process is reduced more than 50%. (McNeill 2006: 5).

Engaging the end-user as customer creates processes that meet both the business and user goals. Pure customer is not enough since usually it is not aware enough of end-users needs and capabilities. (McNeill 2006: 5).

Usability is provided by stealth since user and usability is concerned once storyboards are created. This takes the decision of how screens should look away from the developers. Instead of they are provided with a framework to build from. (McNeill 2006: 5).

Estimation basis is improved since visualization of how the application will look like and behave helps identify best development methods with maximum performance but minimum development effort. (McNeill 2006: 5).

Project risk mitigation is achieved with modifiable user story order and pair programming which ensure continual quality assurance. In addition, with storyboards from user centered design it is easier to inform the vision, test the concepts and confirm the usability. (McNeill 2006: 5).

2.3.3. Designing and Developing in Iteration Cycles

Contextual Inquiry developed by Hugh Beyer and Karen Holtzblatt was presented in chapter 2.2.2. It is part of the Contextual Design which is a method for defining customer-centered

systems (see more Beyer & Holtzblatt 1998). After getting a lot of questions concerning Contextual Design in practice, Rapid contextual design was published (Holtzblatt, Wendel & Wood 2005). Originally it was written to help organizations to adapt method, regardless of their particular situation, and to focus on the core Contextual Design (CD) techniques that most easily drive customer data into the corporate design process (Holtzblatt et al. 2005: 22-25).

Rapid Contextual Design (RCD) consists of upfront design and implementation in cycles. Upfront design consists of several planning phases including contextual inquiry activities with end-users and user story building. Tasks to be performed with the system are illustrated in user stories. (Holtzblatt et al. 2005: 288-290).

Implementation is done in cycles. The basic idea is that user interface design is one cycle ahead of the implementation and testing of the implemented code is one cycle behind the implementation (figure 6).

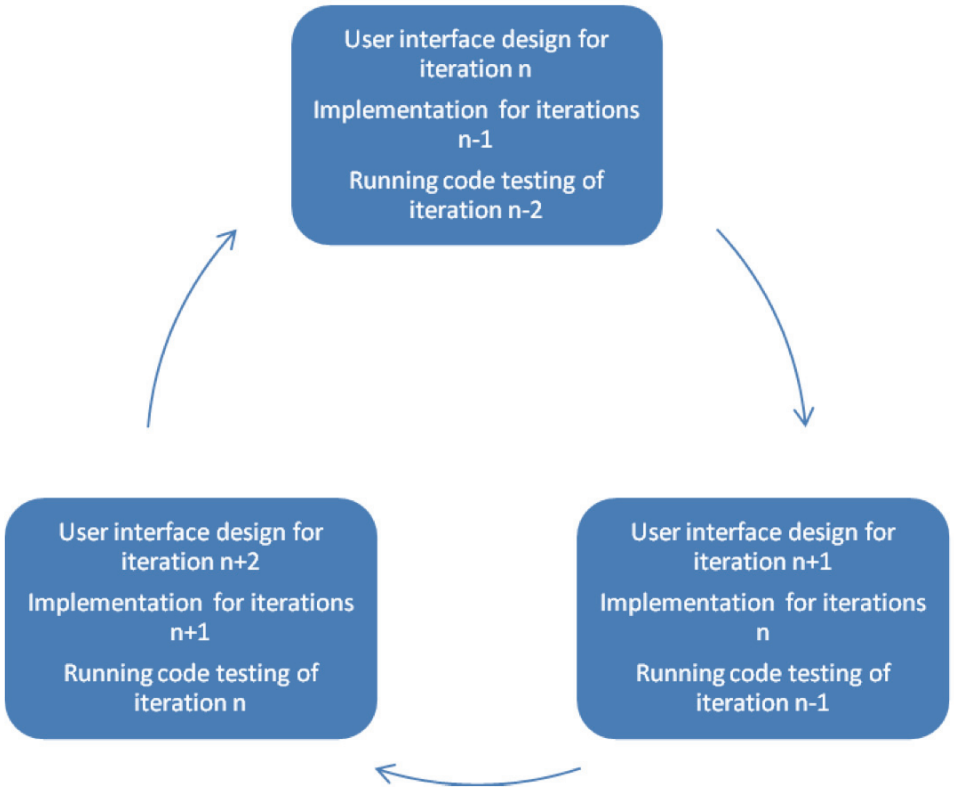


Figure 6 User interface design, implementation and testing in cycles (Beyer & Holtzblatt 2005: 288-290)

UI design is one phase ahead of implementation so there is parallel development of mock-ups for the next iteration. In designing phase UI paper prototypes are tested and discussed with three or four users. Results are used to refine the design and second round of testing is done if time and resources allow. Users are also involved in running code testing. Running code from the previous iteration is tested with users in order to get quick feedback of the actual product. Running code testing can be done after every second iteration or third iteration. (Holtzblatt et al. 2005: 288-290).

Problems or development targets found in user testing may change future stories or existing code. In case problems are found, work estimation need to be updated. (Holtzblatt et al. 2005: 288-290).

Same kind of development idea can be found from other references also. Sy (2007) has written about adjusting and improving user-centered design practices into agile methodologies. This was done when company called Autodesk adopted agile development process for new products. Autodesk wanted to remain traditional usability and user-centered methods in the agile process to guarantee that usability issues are considered in the interface design. They wanted to find a way to conduct usability tests, interviews, and contextual inquiry within agile framework since they wanted to collect data from contextual investigations. That data was supposed to guide rapid iterations of prototypes which are validated by formative usability testing. (Sy 2007: 112).

Earlier Autodesk had proceeded with waterfall process model. To avoid a situation in which some features would be implemented before they were designed they investigated, designed and validated everything well in advance. However, conducting usability investigations almost a full release ahead led to writing many unused or out-of-date feature specifications since they were not able to anticipate all planning issues including business goals. In contrast, agile team only focused on a few new features at a time. They realized that user experience team could also concentrate on part of the release at a time. As a result Autodesk designed just-in-time process model for agile user-centered design. Model is described in figure 7. (Sy 2007: 115-116).

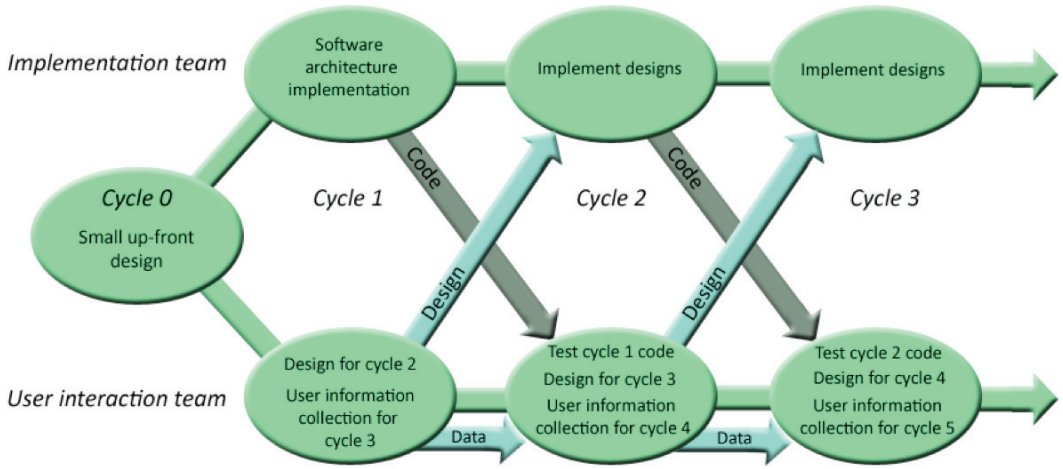


Figure 7 Just-In-Time Process Model (Sy, 2007: 118)

Just-in-time process is very similar to the rapid contextual design for agile development. Process starts with brief up-front design part for gathering requirements and after that user interface design is all the time one cycle ahead of the implementation and testing is one cycle behind. The only difference is user information collection. Unlike in RCD for agile development also user information is collected in cycles. User interface design is based on the collected user information so user information collection is one cycle ahead of the user interface design. In other words, it is two cycles ahead of the implementation. (Sy 2007: 119).

However, also Just-In-Time model may require more usability activities to be done in cycle 0 which is named as Usability investigation activities. In case of a completely new product, interviews or contextual inquiry is conducted for market validation in order to derive design principles that inform and guide design decisions of the product. For an ongoing release of the old product, earlier contextual inquiry and usability test data is analyzed in order to elucidate release-level design goals to guide further design decisions. User group descriptions and workflows are written from usability investigations if a completely new market is as a target. (Sy 2007: 118).

2.3.4. Using Lightweight Usability methods

Agility and user-centered methods can be combined also in another way than making design and development parallel, design being all the time ahead of the development. In CRUISER

lifecycle model (Memmel, Gundelsweiler & Reiterer 2007) the usability agility is based on the usability methods which are classified as lightweight methods. Naturally coding is done by following agile principles, like in XP. This kind of approach is identified also elsewhere in the literature. For example Constantine (2002: 5) who has been involved in developing usage-centered design (see more Constantine & Lockwood 2002) says that their philosophy has always been simply to use whatever tools and techniques which help them to design more usable systems in less time. As an example they draw diagrams only when they need to or when drawing them is faster than not drawing. That is why it is not surprising that usage-centered design is increasingly mentioned in company with other agile processes (Constantine 2002: 5).

CRUISER which stands for a cross-discipline user interface and software *e*ngineering lifecycle consists of four parts: initial requirements up-front (IRUP), initial conceptual phase (ICP), construction and test phase (CTP) and finally deployment and production phase. Common thing for each phase is that methods used are classified as lightweight or agile. (Memmel et al. 2007).

IRUP is the first phase in CRUISER. The phase must not take longer time than requirements analysis in XP. Timeframe is achievable if the methods used are also agile. Developers of the CRUISER regard for example prototyping, focus groups, role models and task model as agile based on their own research results about similarities between extreme programming and human computer interaction. (Memmel et al. 2007).

Appropriate cross-discipline methods for analyzing user needs in IRUP are role models and task models. They are prioritized and sorted in order to divide them finally in three categories: required, desired and deferred. In the end all the models are translated into scenarios focusing on different aspects of user interaction design (users, tasks, interactions) to guarantee shared understanding of developers and communicate with stakeholders. (Memmel et al. 2007).

The second phase is Initial Conceptual Phase (ICP). The main purpose in ICP is to generate more detailed and interactive prototypes in order to narrow the design space towards a single solution. Timeframe and suitability into agile should be considered when abstract level prototypes are modified into more detailed versions. In agile philosophy prototypes should be easy to work with, quick to iterate and easy to maintain for iteration. (Memmel et al. 2007).

Identifying the most promising design solutions in ICP is guided by UI evaluations. Again here, the process needs to be kept simple by choosing methods which are classified as agile. For example user and expert evaluations, collaborative usability inspection and user experience questionnaires are considered as lightweight evaluation methods. In collaborative usability inspection designers, developers, end-users, graphic designers and usability specialists conduct a structured review of site usability in collaboration. Review can also be iterative. However, this is cheaper and faster method than usability testing. (Memmel et al. 2007).

The third phase of the CRUISER is a construction and test phase (CTP) which is quite similar to XP implementation in which software is released in small parts. In the end of the iteration release is evaluated. Code is reviewed against acceptance tests and UI is evaluated with extreme usability evaluations or automated UID tests (Memmel et al. 2007). Automated UI tests could be for example event recording and mouse movement visualization based on the mouse movement recording (Gellner & Forbrig 2003).

2.3.5. Challenges in Agile User-Centered Design Process

Despite the fact that user-centered design and agile software development can work well together, there are several challenges which should be considered once new process is designed. Paul Hodgetts is an agile process coach who has written about his experiences integrating user experience design practices into agile processes. He has listed issues which can turn out to be real challenges if they are not considered carefully in agile user-centered design. He speaks about user experience designers but in this kind of wide context user experience designer can be interpret as a person who also takes care of the usability and user interfaces.

Hodgetts (2005) has listed following issues as important factors in combining user experience and agile development. **Forming the whole team** is vital important. User experience people need to feel themselves as a part of the project community and their activities need to be supported by rest of the team. Besides forming the whole team it is also extremely important to enable and encourage **close collaboration** between user experience people and other designers. Naturally it is important to understand the importance of different activities in the team in order to support those activities. That is why **promoting understanding of user experience activities**

should be done among the team. Collaboration and interaction should happen daily in order to avoid information gaps which can cause serious problems into developed features. One recommendation from Hodgetts (2005) is to recognize user experience design activities that require much time and to **effectively schedule activities** which need to be one or two cycles ahead of construction cycles. In addition to agile design and coding standards, it is also recommendable to **utilize UED-Wide patterns and standards**. They have enabled greater efficiencies in UED teams and, in contrast, inefficiencies have noticed if they are lacking. (Hodgetts 2005: 7-8).

The list of issues above described things which should be considered in order to success in the user-centered agile process. However, Hodgetts (2005) has classified challenges without any unambiguous solution yet. He sees that **long-running and dependent UED activities** such as user research remain problematic. They take a lot of calendar time and are difficult schedule since they require interaction with groups outside the team. Also **culture and organization** can easily cause problems. They definitely should support physical co-location of the UED people and other team in the project. Otherwise integrating the UED practitioners into community can easily fail. As a last challenge Hodgetts (2005) mentions **understanding and respect**. Even though there have been significant actions to improve the understanding of different disciplines a lot has to be still done. Especially in stressful situations each discipline tends to remain more focused on their individual interests as opposed to the overall project interests. (Hodgetts 2005: 8).

2.3.6. Summary of Agile User-Centered Software Design

Agile user-centered software design process has two key points. The first key point relates to the iterative process in which user interface design with prototype testing and feature implementation are done in parallel cycles. User experience team is responsible for the interface design and prototype testing and that phase is usually one or two phases ahead of the implementation design. In addition, user experience team usually makes usability testing with released features one cycle ahead of the implementation. This technique is based on the iterative agile software development which in general should mitigate the risk of the project failure. However, communication between user experience team and developers is compulsory since their work affects each other's work. Changes can be responded at latest in the end of

each iteration cycle. Also time is saved since no time-consuming user requirements or user-interface specifications are written.

The second key point is the use of light-weight methods. This means that methods used in the up-front design should be light-weight without time-consuming hard-to-adopt rules. It is essential to consider project target and existing organization practices once methods are chosen and adapted to be suitable for the project. There is no need to follow slavishly protocols of different methods. Instead those methods should be utilized as is it is best for project. It is totally accepted that methods are simlified for the project if the target is achieved also with less effort.

Project and organization practices are in essential role when suitable light-weight methods are chosen for projects. In order to create successful agile user-centered process these issues have to be considered also once process is created. Otherwise it is possible that created process model will not fit in the organization culture and practices. That is why organization culture, workers habits and needs have to be researched carefully in this research in addition to the existing software project models.

2.4. Design Science

The term design science is used in different contexts in different ways. Van Aken (2007: 68) defines it as *the body of knowledge of a particular discipline on design methods*. Instead, design is referred as *a representation, a model of an entity to be realized and intended as an instruction for the next step in the creation process* (van Aken 2005: 391). An example of the design is a set of drawings for the machine or building (van Aken, 2007: 68).

Van Aken (2007: 69-70) illuminates the design science with the help of the term "sciences of the artificial". He makes a difference between natural science and the "sciences of artificial", first discussed by Simon (1969) in *The sciences of the Artificial*. According to Simon (1969) the base difference is that natural science produces valid knowledge on natural objects while "sciences of artificial" develops knowledge on man-made objects.

Van Aken (2007: 69-70) calls Simon's "sciences of artificial" as Design Sciences because it designs solutions to field problems when it produces knowledge on man-made objects. Examples of Design Sciences are medicine, engineering and law since they are interested in solving field problems, like how to design effective treatments for cancer patients or how to design more fuel-efficient cars or protect society from crime. In contrast, Simon's natural science, which also can be seen as "explanatory sciences", is engaged in a quest for truth. In explanatory sciences they want to solve pure knowledge problems, like why sun and rain produce rainbow of why there are more suicides in one group of society than other. (Van Aken 2007: 69-70).

To make a difference between upper and lower case of Design Science it is clarified that a discipline can be called a Design Science if the main objective of its research is to develop design science. In other words students in Design Sciences are trained to be able to solve field problems through the application of design science. Since it is design *science,* it concentrates on knowledge-intensive designing rather than natural and intuitive designing. (Van Aken 2007: 70).

2.4.1. Professional Knowledge-Intensive Designing

Knowledge-intensive designing like engineering, medicine and law has certain characteristics of the application of design science. According to van Aken (2007: 70-71) these characteristics include the following.

A focus on establishing the right specifications means that professional designing should be done on the basis of well-defined specifications. The nature and the intended performance of the object are illustrated. (Van Aken 2007: 70).

A strong client orientation refers to understanding the needs and desires of the client as one of the key concerns of knowledge-intensive designing. In addition to professional ethics, it is also important because a failure understand of these issues is one of the most common causes of unsuccessful designs. (Van Aken 2007: 70).

A deliberate use of substantive and procedural design science means that professionals are trained for design science. They acquire the skills of applying this general design science to specific field problems. (Van Aken 2007: 70).

A holistic orientation in design science refers to an interest in wholes and in coherence. A good design must be complete and give all the information needed to realize the designed entity. In addition, design needs to be tested holistically. (Van Aken 2007: 70).

A focus on the desired outcomes means that design is about actions and man-made objects producing desired outcomes. That is why eventual outcomes, how to predict these outcomes and how to evaluate them are central issues. (Van Aken 2007: 71).

2.4.2. Design Science Process in Organizational Development

Design science can be used in organizational change. Van Aken (2007: 72-77) presents a design science process which is suitable for planned change, organized in and through projects. This means that even though organization develops naturally continuously there are every now and then planned improvement processes. The intention of the organization project is to change strategies, roles, routines and possible beliefs governing behavior to improve organizational performance against certain criteria. (Van Aken 2007: 73-74).

The process development activity performed in the target company during this master's thesis can be regarded as such planned organizational development. There is a strong target to change roles and routines in the software and user interface development in order to make software development more efficient and results of it more usable. Even though Van Aken (2007: 74) refers to organizational development as changes in strategies roles and routines in wider perspective (organizational change) this kind development frame is suitable also for smaller and precision development like process improvement. Process improvement in this research is not the same as organizational change but it has the same characteristics: roles and working routines are changed in the organizational level.

Planned improvement process typically starts with recognition of a performance problem or opportunity and ends, hopefully, with a realized performance improvement. Design science proceeds following three steps:

- problem definition and defining specifications;
- first redesign, second redesign and formal change;
- learning to perform. (Van Aken 2007: 74).

Stage 1: Problem definition and defining specifications

In the Stage 1 the problem and the desired performance improvements are defined. Usually "change agents" are in charge of organizational development. Change agents consist of management people and staff members but there can also be external consultants. In Stage 1 change agents are appointed and project plan is drawn up. After these actions the problem definition and the project in general are shared with the rest of the organization. (Van Aken 2007: 74-75).

Stage 2: First redesign, second redesign and formal change

Stage 2 starts with further analyzing of the problem. Subsequently change agents make an outline design of the solution. Then they move on into details and redesign strategies, formal roles and routines as far as deemed necessary. This phase, mostly done by change agents, is called as *first redesign*. Typically first redesign is the result of iterative process in which design is evaluated against specifications and the redesigned. (Van Aken 2007: 75).

Second redesign in Stage 2 is about involving stakeholders, the same people whose individual roles, strategies and routines are to be changed, into development process. The realization of the designed changes in roles, strategies and routines accomplished through communication individually and in groups. During the second redesign stakeholders redesign their own roles, routines and strategies. This is done based on the first redesign outcome but also based on their personal ideas and preferences. Preferably intense communication between stakeholders and change agents during the second redesign will construct new structures. (Van Aken 2007: 75).

The second stage of the organization usually ends on a formal announcement about when the new organization will take effect and how does it look like after change. This is usually done on a set date. (Van Aken 2007: 75).

Stage 3: Learning to perform

The formal changes initiated by the project and realized in the end of Stage 2 do not solve any problems if they are just formally described as strategies, new roles and routines. Members of the organization will still have to learn how to operate in the new setting and how to make a success of it. It should be regarded as a learning process. In case change has been more like frame breaking than convergent, there is naturally more to learn. However, learning should be continuous adaptation of the new formal organization to the lessons learned during the Stage 3. These lessons learned can be monitored and managed by change agents. It is important in the Stage 3 that design is turned into action. (Van Aken 2007: 76-77).

2.5. Process Design With Users

As mentioned in chapter 2.2. user-centered design is part of the research in two aspects: firstly the process model which is constructed is user-centered, secondly the framework of the empirical research is user-centered. In fact, the framework for the research comes from the design science for organizational development. However, in design science there are a lot of issues in common with user-centered design. S*econd redesign* is performed with direct stakeholders in the change. In other words same people whose individual roles, strategies and routines are changed are participating in the design of new organizational system which they will use in the future. This is strongly related in the user-centered design – the only difference is the design target. However, same principles can be utilized regardless if the design target is something concrete and physical like software or something more abstract like process model. Some support for this kind of user-centered process development was also found in the software process improvement literature.

2.5.1. Software Process Improvement With Process Users: End-User SPI

Traditionally software processes are improved with different kinds of “Improvement by Design” models in which the process design is separated in time and space from the use of the process (*separation*) (Aaen 2002: 382). One example of this kind of software process improvement (SPI) is Capability Maturity Model (CCM) which defines five levels of the software process maturity and also provides tools to both estimate and develop maturity level (see for example Paulk, Weber, Curtis & Chrissis 1995). According to Aaen (2002:382) there is also a strong focus on formalization which cause that process design is primarily done by special software process

improvement (SPI) specialists. This means that process knowledge is represented by someone else than process users (*externalization*). This applies also for CCM which defines very precisely different maturity levels and their key processes areas. For example in level two in which the process is called as repeatable, key process areas are defined as requirements management, software project planning, software project tracking and oversight, software subcontract management, software quality assurance and software configuration management (Baulk et al. 1995: 33). This means that level two is not achieved until these key process areas are considered.

Multiple problems caused by *separation* and *externalization* have risen up following questions related to software process improvement (Aaen 2002: 382):

- What is a software process? Is it an artifact or is it a set of actions, like something we do?
- Where do we find the software process – in documents, products, or people's heads?
- Who designs the software process? Process developers or process users?
- When is the software process developed? At distinct time? Continuously?

Aaen (2002: 387) has suggested an alternative for the principle "Improvement by Design". In **End-user SPI** process users individually and collectively design their own software processes with assistance from process experts. Improvisation and changing the design via learning is totally acceptable. Aaen (2002) do not describe the exact method how the process design should be done but he gives few principles or ideas as a basis for using End-user SPI:

- Use approaches aimed at establishing software processes at the group level and modify them through group level interaction.
- Use process specialists as coaches or mentors to solve the problems and to diffuse different process ideas from organization.
- Instead of a rigid blueprint the process should be seen as a recipe since software processes are emergent and open to improvisation.

- Improve first, and then model the outcome to avoid modeling what should be done in the future.
- Define essential process elements rather than all process elements and focus on key objectives and deliverables rather than standard procedures.
- Embedded and continual process evaluations at the project level gives insight to strengths, weaknesses and developments in the process for all the stakeholders.

With these ideas Aaen (2002) tries to integrate process development and use as the responsibility of process users. He also points out that software processes should be recognized as knowledge and competence rather than artifacts. In other words, and as the answer for a question presented earlier, software process is a set of actions people do in reality and best it is described in people's head. Based on the End-user SPI there are answers also for the other questions. When it comes to process design and its designer, process users need to participate. They can even design the whole process with the assistance of process specialists. For the question about "When is the software process developed" we can say that process development is continuous process since circumstances in which the process is used are changing all the time. As a consequence, also the work, that process describes, change continuously. That is why also the process needs to changed. Aaen (2002: 383-388).

Börjesson and Mathiassen (2002) compared two SPI projects which used the phases of the IDEAL process improvement model (table 3) totally differently. According to Aaen (2003: 87) IDEAL model is probably the most well known guide to process developments. It is developed by Software Engineering Institute and it is also proposed to use in CCM improvement (see more Paulk et al. 2001: 81-82).

Table 3 IDEAL process improvement model phases (McFeeley, 1996: 6)

Initiating phase	Target is to learn about process improvement, commit initial resources and build process infrastructure.
Diagnosing phase	Current level of process maturity is established and process descriptions, metrics and action plans are developed.
Establishing phase	The goals and priorities are established and action plan is completed.
Acting phase	It is for research and solution development to process problems.
Leveraging phase	Target is to prepare for the next cycle through the IDEAL model and to refine SPI process based on the lessons learned.

The study of Börjessön and Mathiassen (2002) supports the idea of design the process in collaboration with process users as in End-user SPI. The first project in their study, called The Generic Initiative, is based on the "Improvement by Design" and the change agents never worked in close co-operation with software developers. Most of the working time in the project was spent on the diagnosing and establishing phase. A lot of time was spent to communicate with the steering group to establish issues. In contrast the other study, The Dedicated Initiative, used most of the working time in the acting phase. The acting phase was also performed following the principles of the End-user SPI since change agents worked closely with software developers in the deployment phase to make the SPI implementation successful. Implementation success of the SPI project was high in The Dedicated initiative while it was low in The Generic initiative. (Börjesson and Mathiassen, 2002).

3. Methods

In literature review there were presented design science for organizational development end end-user SPI which involves process users in process improvement. The positive experience of End-user SPI gives support to use design science research framework and user-centered design in this master's thesis. In practice the research will follow user-centered design process inside the design science for organization process. This is presented in figure 8.

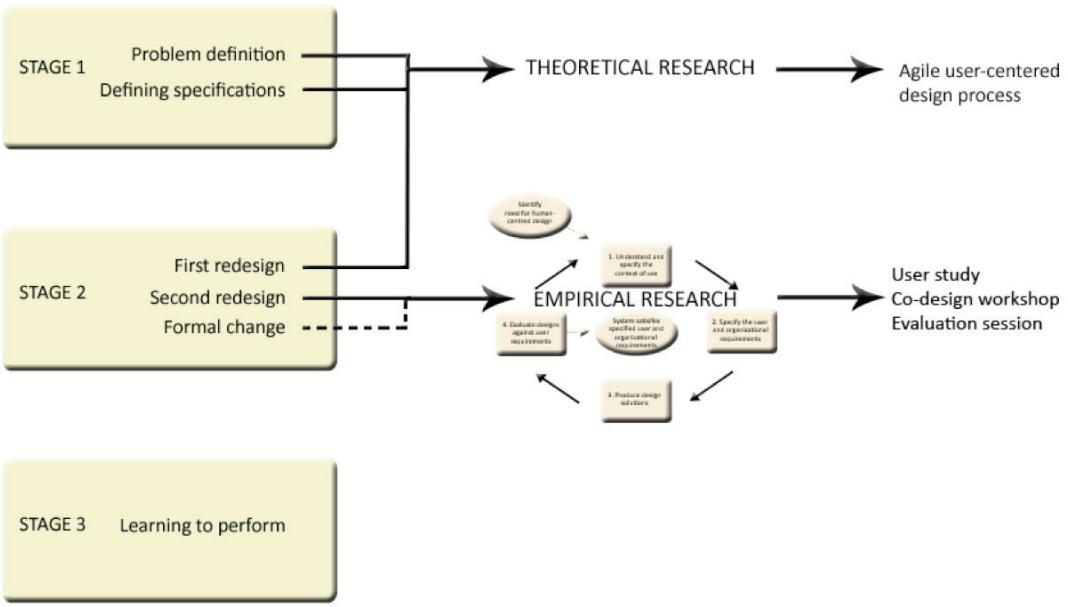


Figure 8 Design Science research framework in the research

Problem definition and defining specifications are discussed in the introduction and literature review. The target in the literature review was to study how user-centered design and agility can be combined and what kind of a process model they could form together. In the end of the literature review, there is an example of the possible agile user-centered design model (Sy, 2007) and how it is used in practice. Model act as starting points for the empirical research, user centered design. It was used as examples and discussion points in the interviews in order to get the interviewee understand what the main point of the research is. That way, it is the *first redesign*.

Second redesign was done during empirical research. People, who are affected by this organizational process development, participated in the designing. They were interviewed in the

user study. In addition to the interviews process users participated in co-design workshop. Process model was constructed based on the interviews and results of the co-design workshop. Uncompleted process model was evaluated with process users and modifications were done based on the discussion and comments.

Once results of this research are completed, in other words once new process model is created and modified into final format; it will be presented for the target company. Then the target company will decide a project or projects in which it is used and tested. This can be regarded as *formal change* since after that at least company employees are aware of the new standard of activity.

Following how new process model succeed in practice is beyond the scope of this research. That is why also *learning to perform* in Stage 3 is out of the research focus. Naturally, it is vital to learn how to operate in the new setting and how to make a success of it. Presumably, during the first projects there will appear some lessons learned so the target company needs to be ready to make changes for the process model. However, this is not handled in this research. It is a natural theme for another research.

Since the research follows user-centered design process, research methods come naturally from the user centered design. Both user study and its methods, and designing with users are presented more in general in chapters 2.2.2-2.2.4. Next chapters only present those methods from the point of view of this research.

3.1. User Study Interviews

User study was conducted as interviews so the main research method was interview. Interviews were part of the *second redesign.* Preliminary process model idea, agile user-centered design process, was the *first redesign* which was redesigned during the *second redesign*. Interviews also served phases 2 and 3 in the user-centered design process. Through the interviews researcher familiarized herself into organizational context and found out user needs and requirements.

Interviewees from the target company were from Finland located in Espoo (16), Jyväskylä (1) and Pori (2). At the beginning target was to make interviews also in Germany to guarantee that

international aspect is considered and wider perspective of visions will be included into process improvement. However, Germany organization did not see participating in this study reasonable due to totally different working habits in software development.

In table 4 there is presented background information of the interviewee participants. Contextual factors column tells whether contextual factors were discussed in the interview or not. However, even though contextual factors were discussed, not all the contextual factors were discussed with everybody. Co-Design Workshop column tells about participation into co-design workshop. In addition, it also reveals which group work group the interviewee participated. Evaluation session column describes if the interviewee also participated in the evaluation session organized after co-design workshop.

Table 4 Background information of the interviewee participants. MinPro stands for Minerals Processing and ORC for Outotec Research Center.

Gender	Age	Job title	Business Division	Contextual factors	Co-Design Workshop		Evaluation session
					Group 1	Group 2	
Male	41	Project Manager	MinPro				
Male	53	Manager	MinPro	x			
Male	31	Manager	MinPro	x			
Male	28	Software Development Engineer	MinPro	x	x		x
Male	59	Project Manager	MinPro	x			
Male	52	Software Engineer	MinPro	x			
Male	28	Software Development Engineer	MinPro	x			
Male	40	Manager	MinPro	x			
Male	62	Project Manager	MinPro	x			
Male	55	Senior Technology Adviser	ORC				x
Male	44	Senior Technology Adviser	ORC			x	
Female	52	Automation Engineer	BaseMetals	x			
Male	49	Automation Engineer	BaseMetals				
Male	32	Manager	BaseMetals	x	x		
Male	33	Automation Development Engineer	BaseMetals	x		x	x
Male	43	Product Engineer	BaseMetals				
Male	29	Automation Development Engineer	BaseMetals	x			
Male	24	Automation Engineer	BaseMetals	x		x	x
Male	31	Software Engineer	BaseMetals	x			

One interviewee group was software engineers and project managers. People who have participated in the software development process in design or project management level discussed comprehensively about their past software development projects. This was done in order to create a deep understanding about how things are done at the moment in the target

company. To find out improvement ideas and targets, they told how software development should be done in their opinion. They were also asked about their knowledge and experiences of the user-centered design, agile methodologies and details about couple of contextual factors.

Basically the target company does not make any bigger-scale implementation. Instead, they use subcontractor for implementation. There was also one software engineer from the subcontractor in the interviews. Questions in that interview varied to some extend since the perspective was different.

Besides software engineers and project managers, also upper level managers were interviewed. Managers from Research and Concept Development, Product Development and Global Technology gave a bigger picture about the software development as part of the company's business. They also described, or at least gave some insights, how they see that software development, or product development in general, should be done in the target company. Also, they were asked about their experiences and opinions of user-centered design and agile software development.

These managers' interviews revealed more about the contextual factors due to position of the interviewee. In addition, impact of contextual factors on the software development process was discussed more deeply than with software engineers.

Interview topics are summarized in the figure 9. Last topic, contextual factors, is marked with pink arrow since it was a voluntary topic – it was discussed if there was time left. However, in manager level interviews some factors were discussed in any case. Interview questions for different interviewees can be found from appendix 2.

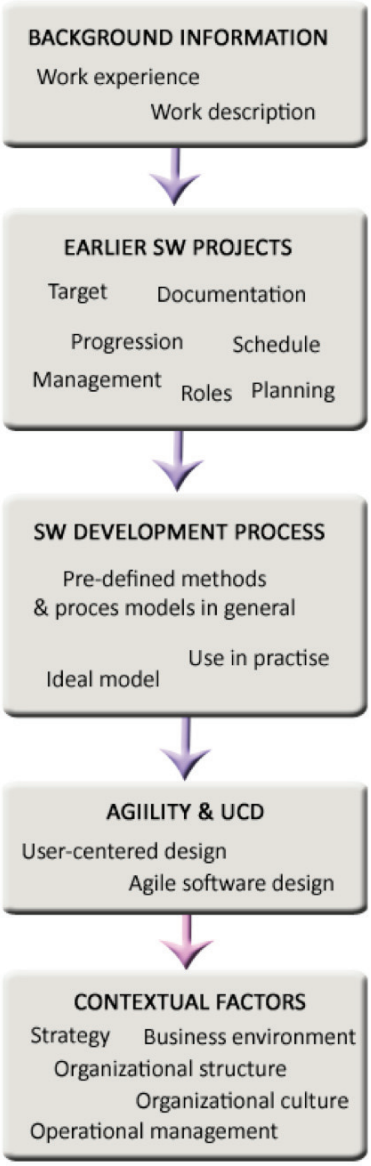


Figure 9 Themes and topics discussed in the interviews

There were a lot of questions listed in the interview question bank but basically the target was to collect qualitative information with the help of pre-defined themes. Not all the questions needed to be asked from every interviewee if some other theme was more interesting and it was reasonable to discuss more about that.

Interviews were recorded with mp3-recorder. However, notes were written during the interview to ease material construction and analyzing afterwards.

3.2. Co-Design Workshop

Prospective process users, in other words employees in the target company, participated in co-design workshop. There were together seven participants. For the co-design task they were divided into two groups, one consisting of three members and one consisting of four members. Each group included participants from Base Metals and Minerals Processing. Five co-design workshop participants participated also in the interviews. They are presented more deeply in the table 3. In addition there was one software engineer (26 years old) and industrial designer, both from Minerals Processing. The industrial designer is one of the subscribers this master's thesis.

In the beginning participants were presented the results of the interviews and user personas constructed based on the interview results. Two different user personas had challenges in working caused by unplanned and not standardized software development practices. After presentation participants analyzed interview observations by constructing an affinity diagram. Affinity diagram construction started with observation grouping (figure 10). Subsequently they started to order groups together and that way build an affinity diagram (figure 11). Interview observations were printed on the paper beforehand so they only needed to clue them on the post-it. When observations were grouped and affinity diagrams were ready they designed solutions which would solve, or at least ease user personas' challenges (figure 12).

Figure 10 Workshop groups grouping observations

Figure 11 Organizing groups

Figure 12 Designing for user personas

The target in the co-design workshop was to create more discussion of the topic in a group. Results, analyzing and user personas were starting points for discussion. New opinions and aspects were hoped to arise from these discussions. These opinions and aspects steered the design process into new direction or, at least gave support to the previous thoughts. If somebody had proposed some kind of a process model, this kind of research data would have been acceptable and definitely noticed in the design phase.

Workshop took one working day with lunch break. Day consisted of following activities:

- Coffee, welcoming, free discussion *15 minutes*
- Presenting the project shortly, workshop day agenda *15 minutes*
- Presentation of the interview study results + comments *60 minutes*
- Break *10 minutes*
- Presenting workshop task, dividing into groups *15 minutes*
- Grouping observations *60 minutes*

- Lunch break *60 minutes*
- Affinity diagram construction, designing based on user personas *90 minutes*
- Presenting affinity diagrams and design solutions + discussion *60 minutes*

Workshop materials for both groups consisted of:

- Beforehand written observations
- Green, blue and purple post-its
- Scissors
- Clues
- Big white papers
- User personas in paper

3.3. Evaluation Session

In user-centered design process prototypes should be evaluated with end-users during the process in order to ensure the right track of the designing. In prototype testing or evaluation session users give feedback of the prototype and feedback is noticed in the further development.

Unfinished process model, prototype, was also evaluated with the target company employees. Feedback and discussion in the session was noticed in the finalization of the model. Evaluation session was the first time when prospective process users were shown something concrete. Researcher was prepared to change the model change even radically after the session.

Evaluation session took 2.5 hours and there were six participants from the target company. Four of the participants are presented in table 3. Two other participants were the same persons who also participated in the co-design workshop (software engineer and industrial designer from Minerals Processing).

Evaluation session started with short repeating of the project. Then participants were presented results from the co-design workshop. Subsequently preliminary step model for increasing the level of agility and user-centered design was presented in detail. Participants commented step model at every step. Discussion was very productive and the attitude towards steps was definitively positive. After that there was a short break. Last hour was spent for presentation and discussion about the model for agile user-centered software design. Even though the idea

of the model was basically agreed with, the model also created differing opinions among the participants. However, they participants presented development targets what was the target in the evaluation session.

4. Results

This part presents first how interview data was analyzed. Mind-maps and user personas were constructed by the researcher. They are based on grouped interview answers. Instead, affinity diagrams constructed in co-design workshop were done by workshop participants. With affinity diagrams they revealed how they would analyze raw data since their post-its consisted of observations collected from the interviews.

4.1. Data Analysis

Analysis done by researcher started with transcription of interview notes. Recordings were listened if something needed to be checked due to unclear notes. Otherwise transcription was done based on the notes. After that most of the answers were grouped into excel-file. One sheet contained answers considering current standards of activities in the target company. They are based on the projects interviewees have participated. Present state in the target company was studied with the help of project descriptions. In the second sheet there were collected answers related to process models in general. The third sheet contained answers and opinions considering different models: agile development, user-centered design and agile user-centered design. Interview answers related to the ideal software design process were grouped into the fourth sheet. Transcript notes and excel file was the base for real analyzing.

4.1.1. Mind-Maps

Interview observations, in other words comments and opinions collected in the interviews, considering process models were organized into mind-map. The focus was on the observations considering ideal software process model since those were the base for user needs and hopes. Mind-map consisted of following main topics with hierarchy level:

- Experiences about models in general
- Existing models
    - Waterfall model
- Ideal software process model
    - Project organization
    - Follow-up

- Documentation
- Implementation
    - Requirements specification
    - Subcontractors
- Testing

Even though mind-map was constructed as affinity diagram, it is not called as affinity diagram for two reasons. It was made with mind-map tool and it does not follow all the instructions of affinity diagram. For example there are too many observations in the groups at lowest hierarchy level. However, in practice the difference between affinity diagram and mind-map is subtle. The definition and way of constructing may differ but the idea is the same: to group observations in order to point out important issues.

After mind-map was created the observations under "Ideal software process model" were considered further. Observations supporting user-centered design were highlighted with red. Similarly, observations supporting agile methods were highlighted with blue. It should be noticed that discussion of the ideal software process model was done in the interviews before agility and user-centered design were presented for the interviewees. Mind-map includes also information of how many times some need or requirement existed in interviews.

Based on the observations under the "Ideal software process model" a new mind-map was created. The purpose was to collect into one file all the issues which should be noticed when the new model is constructed. Besides observations of the ideal software process model, also current state observations were analyzed when restrictions and possibilities were considered. Regardless how much there are user needs and change targets, the current situation in the target company gives the framework for the change. This mind map is presented in appendix 3.

4.1.2. User personas

User personas are used for illustrating research data about users. User personas are also used in designing phase – it is easier to design for some concrete person with concrete features. Two user personas were created based on the interviews and interviewees who are prospective process users. Personas were used in the co-design workshop to motivate and ease design.

Besides background, both personas had challenges that complicate their working. The purpose of these challenges was to describe the current problems in the organization's software development. Target was that concrete persona with concrete and familiar problem would motivate co-design workshop participants to design solutions for the challenges. User personas constructed from the interview results are presented in figure 13 and figure 14.

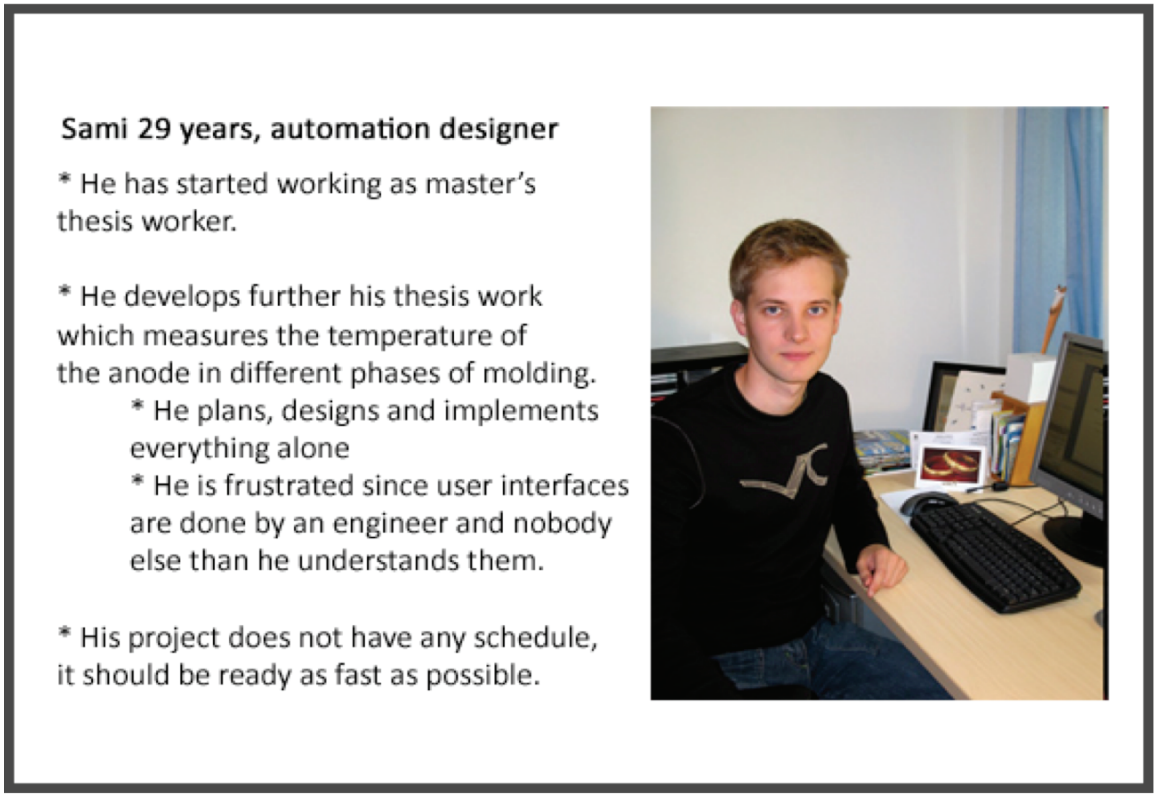


Figure 13 User persona 1

Figure 14 User persona 2

4.1.3. Affinity Diagrams Constructed in Co-Design Workshop

One of the tasks in co-design workshop was to build an affinity diagram (figure 15 and figure 16). Research data collected in interviews was analyzed with the help of affinity diagram. Observation grouping revealed how participants perceived the entity and pointed important issues as topics. Affinity diagram construction was not a proper design task but participants needed to familiarize themselves into topic and observations more deeply before they started to design for user personas.

Figure 15 Affinity diagram constructed by group 1

Figure 16 Affinity diagram constructed by group 2

Afterwards affinity diagrams were digitized. Content maintaining hierarchy and shape were reproduced with Excel.

Participants used affinity diagrams in producing design solutions for user personas. They were instructed to design solutions for the user personas' challenges by thinking how these kinds of problems could be avoided in the future. Without further instructions both groups chose to find solutions from the affinity diagrams. They marked with pencil observations which would ease or

solve the problem. First they gave numbers for problems in user persona paper. Some of the observations in affinity diagrams were suggested to ease multiple problems of user persona. Some were written additional information. In addition to solution searching, they also marked observations which cause those problems, in other words which are risks for successful software development. All of these marked observations are also highlighted in Excel –version of the affinity diagrams. Particularly they were considered in new model creation.

4.2. Current Situation in the Target Company

The target company does not have any software development process model. Minerals Processing division uses in some projects product development process model from quality system. However, if process model is used, it is used without any systematic standard of activity. They do not use any process model in most of the projects. In general people are doing projects as they see fit and there is not any common standard of activity for planning, implementation, testing or documentation. Results regarding current situation in the target company is presented in detail in following two chapters. Based on the research data a process model of the current standard of activity was created. However, current process model is generalization since there are almost as many ways to make software as many developers there are.

4.2.1. Project Stakeholders

Table 5 presents interview data considering project stakeholders. 10 projects out of 20 projects had designers from different disciplines. Typical case was a project group consisting of one software designer, one mechanical design, one or two electrical designers, product manager and/or project manager. However, usually only software designer was the only one to develop the product from the target company. Mechanical and electrical design was usually done by subcontractors. Subcontracting in different disciplines was used totally in nine (9) out of 20 projects.

Not all the projects contained physical device which required mechanical design. There were also pure software development projects. Six (6) projects were middle size software projects which had more than two developers. Most of them used subcontracting for implementation so in practice only one or two persons were working for the project from the target company. Two (2) projects were very small projects with only one or two developers from the target company.

Half of the projects had a project manager. Some of the project managers were supervisors and acted as a support and organizer for the project group. In four (4) projects manager participated into design and development work with other developers. Practically in those cases project manager was just one of the development team.

Usability issues were considered very seldom. In three (3) projects there was used separate user interface designer. Two (2) times industrial designer from the target company designed user interface and in one (1) project they used subcontractor in user interface design. Customers or end-users were asked feedback only in two (2) projects. In addition, three (3) projects utilized feedback from the previous versions. Feedback was introduced by sale, introduction or maintenance personnel.

Table 5 Project stakeholders

Project number	Nature of the project			Project manager	Subcontractor for implementation	Separate user interface designer	Customer / user involvement	Feedback from sales / introduction / maintenance
	Multiple design areas in project	Middle size software project	Small size software project					
1	x				x			x
2	x						x	
3	x							
4			x	x				
5	x			x				
6	x				x			x
7		x		x*	x	x		
8		x		x				
9	x				x		x	
10		x		x*	x			
11			x					
12	x							
13	x			x*				x
14	x			x*	X**	x		
15	x				x			
16				x				
17		x		x	x	x		
18		x		x	x			
19		x			x			
	10	**6**	**2**	**10**	**9**	**3**	**2**	**3**

* Project manager participated in the project as developer, designer or specification constructor.
** Subcontractor used for user interface design.

4.2.2. Project Practices

Project practices varied almost in every project. That is why the classification is done in two categories: projects with only small up-front planning and specification phase before implementation and projects with bigger specification phase before implementation. There were eight (8) projects in former category and five (5) in the latter category. However, projects in both categories were iterative. Iterative development was mentioned quite often since interviewees highlighted that project and development proceeding can not be planned beforehand. They pointed out that something unexpected always appears during the project.

Some kind of prototype construction was done in nine (9) projects. In some projects it was just a testing in laboratory settings but in some cases prototype was tested in real environment. Unfortunately data from the prototype tests is not utilized that well. Usually prototype is considered as ready product if it just works.

Communication in the projects was done with typical tools: informal discussion, e-mail, phone and meetings. Notable fact is that only six (6) projects had pre-planned project meetings. Naturally, if team members are sitting next to each other they can communicate very easily (3 projects). This does not work when there is subcontractor not located in the target company office. Quite many projects (6) with subcontractors did not have pre-planned meetings.

Documentation, again, have multiple practices. In some project documentation can be very detailed and pre-planned while in other project documentation is not done at all. However, integrative approach is satisfaction, or non-satisfaction, to documentation production. In 11 projects documentation production was seen unsuccessful and at least some development targets were proposed. In many cases too little documentation was done or it was done with unsatisfied way. Only one (1) project was seen to be successful considering documentation. In four projects discussion consisted of neutral description about produced documentation.

Table 6 presents projects from project practice point of view.

Table 6 Project practices

Project number	Small up-front planning / specification	Specifications before implementation	Iterative development	Prototype construction	Face to face communication		Documentation		
					Working close to each other	Pre-planned meetings	Not satisfied	Neutral	Satisfied
1	x						x		
2		x	x				x		
3				x			x		
4	x				x		x		
5							x		
6		x							
7					x			x	
8	x		x	x	x	x			
9		x	x	x		x	x		
10		x		x		x	x		
11			X	X		X	X		
12		x							x
13	x		x	x		x	x		
14	x			x				x	
15	x			x		x			
16	x						x		
17	x			x		x*	x		
18								X	
19		X						X	
	8	5	5	9	3	7	11	4	1

* There were pre-planned meetings only in the early phases of the project.

How projects proceeded was also researched. Results are presented in table 7. Eight (8) of the projects used project plan and eight (8) did not use. If the project was master's thesis project the plan was the same as master's thesis plan. The lack of project plan describes quite well the situation in which everybody does as they see fit.

Only three (3) projects mentioned that they did not manage to get the project ready in schedule. One explanation is that when projects do not have plans they do not have any strict dead lines. In addition, some of the software development projects are done when other tasks permit. Five (5) of the projects were mentioned to be this kind of non prioritized projects which are easily postponed.

Regardless of the unsatisfied practices in development phase, interviewees were generally happy with the end results. They thought the project was a success when they managed to produce a working product and usually in a bearable time. Only one (1) project was said to be a horrible mess and subsequently very unsatisfying in the end.

Table 7 Project proceeding

Project number	Project plan used		Project is done when other tasks permit	Time estimations exceeded	Success		
	No	Yes			Yes	Both	No
1	x					x	
2							
3	x	x *	x	x			
4		x					
5	x						
6		x					
7		x	x	x	x		
8	x				x		
9		x					
10		x	x				x
11	x		x				
12		x			x		
13	x						
14		x			x		
15				x		x	
16							
17	x						
18	x **		x		x		
19					x		
	8	8	5	3	6	2	1

* In the beginning there was a plan but after unexpected change the plan was not anymore sufficient and it was not remade.
** There was made a project plan due to Tekes project but it was not updated after changes.

4.2.3. Generalized Process Model

In the tables 5 and 6 issues which occurred in multiple projects were highlighted with grey color. Those were *multiple design area in project*, *project manager* existence, use of *subcontractor*, *small up-front planning, prototype construction* and *unsatisfying documentation*. These issues were repeated with some level almost in every interview. In addition *iterative development* was highlighted even though it was officially mentioned only in five interviews. However, also those

projects in which implemented features were specified beforehand interviewees mentioned that they have much more feedback than in traditional waterfall model. So in practice almost all of the development was iterative, only the level varied.

Based on the interview results considering current software development practices in the target company a generalized process model was created (figure 17).

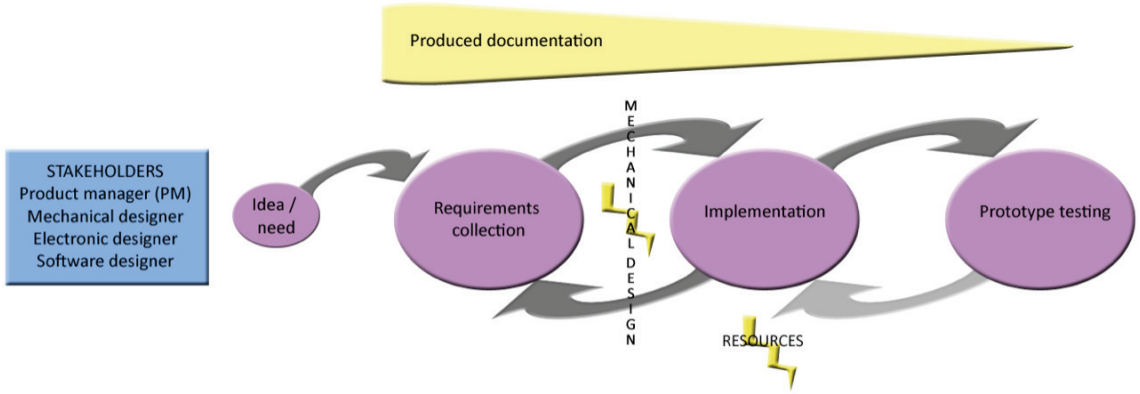


Figure 17 Generalized software development process model

Project stakeholders are at least software developer(s) but usually there are designer(s) from software design, mechanical design and electrical design. In addition usually there is product manager who can act as project manager too or then the project manager is a separate person. They can also be developers. In some cases there is not any designated project manager: either there is somebody supervising or then project is done without a manager.

Project starts from idea or need. Project can be further development of the existing product (10 projects in the research) or new product development (7 projects in the research). However, similarity for both projects is that they start either from idea like “We could use this technology” or from need like “We need to change our line user interfaces into graphical user interfaces”.

After need or idea there is a rather small planning phase which is called as requirements collection. It depends on the project what is planned but in most of the projects some requirements were specified. They can be functional, technical or customer requirements but the main idea is that project group knows what they are going to do. Requirements collection corresponds *small up-front planning*. However, they start implementation quite fast, also since

they see that it is impossible to plan things when you do not exactly know what you are going to do. From implementation there is strong feedback to the requirements collection phase. Software requirements are changed among project proceeding. Features can be changed, modified, added and deleted along the implementation. This is easy when usually there is only so small team developing the software or at least they have clear remits.

After they have something ready as an entity they make prototype testing, either in laboratory or in the real environment. Unfortunately feedback to the implementation phase is quite loose. According to co-workshop discussion, prototype is sent to the world if it just works, no matter how well it works. Usually there is no time to do modifications even though they would be needed. Compulsory repairing is done in the field along the introduction.

The way how documentation is done could not be described in the model since there were not any uniform practices. Instead, the yellow triangle describes the amount of documentation and also satisfaction for produced documentation. It was typical that regardless how much and how well documentation was made in the beginning, the amount and also satisfaction decreased along the project proceeding. This does not mean that satisfaction is straight related to the amount of documentation. Instead it means that satisfaction diminished since they did not produce documentation they would have liked to produce.

There are two lights marked in the process model. One is *resources* which imply the fact that there are too many projects in which software development is done by one or two developers. In worst case they have also other tasks to do and software is developed when other tasks permit. Then the project takes even years. The other light is marked for mechanical design. This goes for projects in which there is a physical device, like measurer, involved. Based on one interview discussion, typical case is that mechanical design has exceeded schedule and when the software development can finally be started, there is no time left. Software needs to be done as fast as possible and it is regarded as bottleneck in total product development. This puts challenges for software development.

4.3. Organizational Factors

Organizational factors, in other words organizational use context of the process model was discussed in the interviews. Organizational factors were not considered particularly in process design but all the solutions were evaluated against recognized factors. In general the quality of business affects software development. The target company is doing business to business. Their customer relationships are long-lasting. That is why there are not any strict schedules and delivery times can be quite long. There is no compulsory need for big resources. In addition, since customers are even partners they at least think that they know the customer and its needs. However, research indicated that the amount of customer collaboration is still quite slight.

Different contextual factors of the target company are summarized below.

Business environment

Mostly strategy affects to developers' work even though some interviewees pointed out that they do not even know the strategy. The target company wants to be the best so employees want to do their best. Individuals have a strong need to show and prove their expertise. Mission of the target company is to create future business with product development so people see their working important. However, in general business is not concentrated on software development in. On the other hand, a lot of products are still related to software development. In Automation there only a small part of the projects related to software but it is big part in turnover. Since there are only few projects there are too small resources. This leads to a situation in which they do not have competence to fulfill the market's needs.

They also want to make good quality. As one interviewee said that it is self-evidence since that is the only way to compete as a Finnish company. However, unfortunately there is seldom time to do good quality. Also knowledge and experience are lacking. One interviewee mentioned that agile development could help for tight schedules since you would have a clear and realistic goal for one sprint.

It was pointed out that quality refers to the product which meets the customer requirements. In addition, for example there can bugs in the code – quality refines how fast those are fixed and how well company is committed in the maintenance.

Their strategy is to add service business. At best this is done with standardized products since then they can provide spare parts and consultant services. In addition they do not need to educate so many maintenance personnel.

Subcontractor and customer collaboration

In general using a subcontractor is a good thing when they do not have resources by themselves. However, there have been also challenges. Typical challenges are problems with communication, software is not completed in schedule or software does not meet the requirements since requirements are not communicated well. Project manager should follow subcontracting proceeding and there should be communication at least once a week. It also affects how much subcontractor has done work for the customer earlier. With experience you can guarantee more predictable and better result. In general, the bigger the effort is from the customer the better results are. Sometimes subcontractors may even have more knowledge about mechanical issues. However, also the expertise of the subcontractor should be questioned if needed.

Core competence which requires own critical knowhow should be done by own designers. Those cannot be done by subcontractors.

Customer requirements have also caused challenges. It has been hard to find common way to communicate customer requirements. They should be more specific. Also when customer makes individual requirements productization is hard.

Operational management

Most of the workers have multiple roles in multiple projects at the same time. There are good and bad point of views. Usually the design work or working flow is interrupted and there can be situations when the product should have been ready already yesterday. On the other hand it is

more meaningful to have more than one task to do. It is also good to have something else to do if main project needs to wait for something.

Strategic management

Strategic management affects a lot. They say that they are building cathedrals, not just piling bricks. The target company's brand is quite popular. However, there are several levels in strategy consciousness: connection to the everyday job can be very low. One weakness is that they are not a software company but they still make software.

Company infrastructure

Working space and atmosphere are good, like possibilities to train if needed. At some point they can affect their projects. There is only man per one task so basically they cannot choose their tasks inside the project.

There is a good support for the work. For instance, If they have asked for some tool they will get it. However, some interviewees considered methods and tools as old-fashioned.

Organizational culture

Everything is under modification in the company. The most important thing is to do useful things organized. Best practices and lessons learned should be considered more. For instance there has been some improvements for working processes but mainly those are based on manager level actions and they have no been useful in practice. Product development department has made some suggestions by themselves to improve their daily habits and they have also succeeded.

Organizational structure

There is a strong matrix organization (earlier it was weak) so they are organized into technology lines. Sales and marketing tells to the lines what kind of technologies should be developed. In customer projects tasks are stricter but in the technology innovation development there is more freedom.

In projects project manager should be fulltime and at least in the most intensive phase designers should also work fulltime in that project. It is hard and risky if only one person is working on the project.

Software engineering practices are volatile but everybody has main knowledge from the education. Otherwise the job is learned in practice. Target is to follow the quality handbook and self-documentation.

Knowledge transfer

Mostly communication is done face-to-face when they see each others. In some projects they organize a meeting if some problems occur. Project managers are in contact with subcontractor about their works. If subcontractor would work at customer's office it would ease communication.

Competence

Recognizability in the market and brand are good. Customers believe that products are of good quality. In reality they have a good knowledge in metallurgy but not so good in the software development.

4.4. User Needs for Software Development Process

Counting how many times some need or requirement was mentioned in interviews is an option to analyze interviews. This would give a strong support for solutions in designing phase. However, since the research method was qualitative also the analyzing method needs to be qualitative. Users were not given options of which they could be able to choose features they would like to have in new model. Those would have been easy to count. Instead ideal software process model was discussed as a free theme, introduced with question: *What do you think: how should software development be done in your organization?* In addition interviews were given following issues which could be handled in the answer: *phases, participants and stakeholders, the size of the working group, work distribution, usability design, agillity, testing, schedule and follow-up.* Interviewees answered for the question from different point of views and they were not conducted into any specific direction after the wordlist – they were allowed to speak about the issues as they wanted. That is why analyzing requires some interpretation

even though there were a lot of similar issues discussed in the interviews. Only the used language differed. For instance somebody said that due to lack of resources, requirements specification should be light and implementation should be started as fast as possible in order to make the product fast. Some other interviewee pointed out that implementation and functional specification should be done in cycles. These both user needs were interpreted to be a need of agile development. They both have the characteristics of agile development. However, more closely thinking, they are not exactly the same need.

It should be remembered that if interviewee did not mention something in the interview, it does not mean that they would have reject it. Interviews could have been asked what do they think about that and that but presumably they would have agreed improvement suggestions anyway. That is why it was more interesting just to listen what comes to their mind.

Project organization

The most discussed issue in interviews was project group. Seven (7) interviewees out of 19 mentioned that there is a need for more than one person in the project. There were multiple reasons. Firstly, it is important to have somebody to discuss with and secondly, it mitigates the risk of lost knowledge if somebody leaves the company. In addition two (2) interviewees pointed out the importance of proper project organization with project manager. Project manager should take the responsibility of the proceeding and quality. Project participants should be aware of the technology (1) and task specification between project participants should be clear (1).

Five (5) interviewees favored using subcontracting in implementation due to lack of resources and knowhow. However, there were additional comments related to subcontracting. Subcontractors should be located at the target company premises (2) and they should be involved already in specification phase (1). One (1) interviewee also pointed out that using subcontractor requires more up-front planning. Only one (1) interviewee mentioned that it is useless to use subcontracting in new product development since then the knowledge will be out of the company.

Software development

Principles of agile development were mentioned in four (4) interviews. Some interviewees wanted specification, implementation and testing to be done in cycles while others just highlighted light upfront planning. Only one (1) interviewee pleaded for profound planning and specification generation before actual implementation.

In general, planning was regarded as a challenge. One (1) interviewee mentioned that if they had more resources it would be useful to do better planning in projects.

Even 10 out of 19 interviewees wanted to have status, checkpoint or review meetings during the project. These meetings support project proceeding and give a communication channel for the project stakeholders. In general frequency depends on the project and its state but interviewees propose that whole project group should meet at least once in a month. In addition there can be smaller status meetings with smaller groups more often.

Prototype construction was seen important in three (3) interviews.

Feedback collection

Seven (7) interviewees pleaded for customer involvement in the software design and development. Some discussed about customers and some about end-users but main message was the same: it is a benefit to collect feedback and opinion from the field. Some of the interviewees approached the issue by mentioning that lessons learned from previous projects should be utilized.

User interface design

Four (4) interviewees wanted user interface design to be done by professional user interface designer. Two (2) of them wanted them to take care of the look and feel also. One (1) interviewee pointed out that readymade basic screen models would ease designer's work in general.

Documentation and testing

Documentation and testing were the most complicated topics. Most of the interviewees did not comment those all, some only mentioned that they need to be improved. Two (2) interviewees

proposed that documentation should be done along the coding. On the other hand one (1) mentioned that model should describe how documentation is done while the other (1) proposed that proceeding should be followed with documentation. Also the language of the documentation should be considered – one (1) interviewee pointed out that it is quite stupid to ask developers to make documentation in English when they do not even want to write in Finnish. However, in general the message was that there should more documentation but still light.

Process model in general

Seven (7) interviewees emphasized that there is a need for common process model but it should be adjustable. In addition to size of the project there can be different priorities. It should also be considered that new product development project or additional development projects are totally different. Some projects also customize the product to be suitable for designated customer.

4.5. First Redesign

According to van Aken (2007) first redesign in organizational development consists of further analyzing of the problem and construction of an outline design of the solution. Usually first redesign is made by change agents. In this research first redesign is made by a change agent (researcher) and due to the problem challenge, it is based on the literature review. Agile user-centered software design model was presented on chapter 2.3. Developing iteratively in cycles is one of the main philosophies of agile methods. Combining user interaction design in this kind of iterative development in cycles seemed interesting. The main idea is that design is done cycle ahead of the implementation and testing is done one cycle behind implementation. Model in figure 18 (Sy, 2007) notices also user information collection which is naturally done before user interface design. It is two cycles ahead of the implementation.

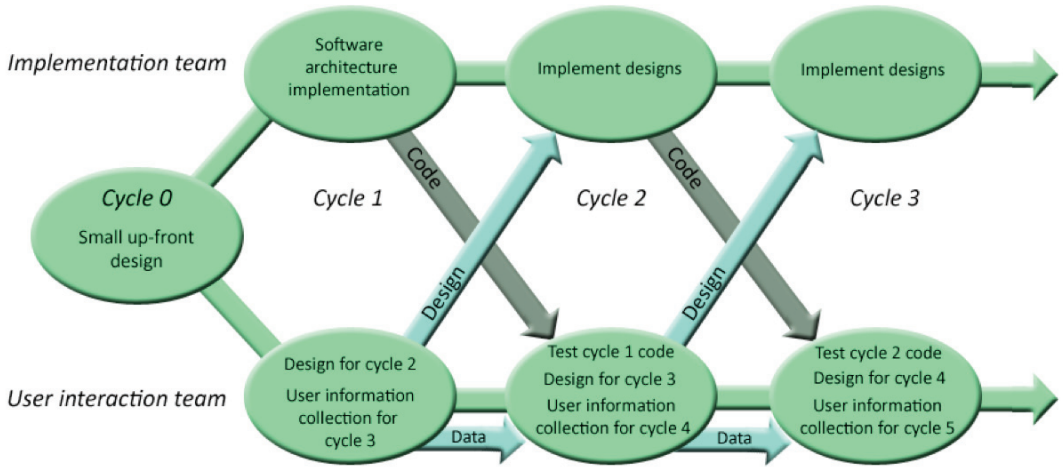


Figure 18 Model for agile and user-centered software development in cycles (Sy, 2007)

Model in figure 18 was chosen to be a starting point for the process improvement. As a starting point it is also the first redesign. This kind of model was a natural choice after making literature review. It combined user-centered design and agile development, starting points of the research, with compact and understandable way. It has clear phases with clear tasks. Of course the model as such was radically different than the current situation in the organization. Initial data revealed that they do not implement agile methods and they do not use end-users in the design and development. In fact, it was quite clear from the beginning that they do not have any process model and people do software development as they see fit. However, radically different model was seen as a chance for second redesign. Usually radical changes (even though they are just suggestions) cause more opinions and discussion. The target was that interviewees really start to consider the situation, possibilities and restriction and give their opinions. This was planned to be achieved with radical, but still simple and compact model.

At this point it is reasonable to illustrate a little bit interviewees' knowledge about agile development and user-centered design. 18 from 19 interviewees were asked about their knowledge and possible experiences. 10 of them did not know anything about agile software development and 12 of them did not know anything about user-centered design. Six (6) interviewees knew some basics about agile development while only two (2) knew more about agile practices and had some experiences about using them. Four (4) interviewees knew some basic ideas behind user-centered design while only two (2) knew how they are utilized in

practical designing. In other words change to the agile and user-centric approach would be a huge step since the level of knowledge in both disciplines is low.

However, first redesign was presented for the interviewees in the interviews. Naturally some of the discussions after presentation were more fruitful than the others but still there were lot good opinions to consider in following phases of the process design.

Following attitudes and opinions are summarized from the discussions in the interviews. Phrases in quotation marks are straight quotations from the interviews. Firstly there are presented opinions considering only agile methods and user-centered design. Interviewees were presented scrum development model as an example of agile method and ISO 13407 as an example of user-centered design process.

Agile development

- Working in pairs or teams is efficient considering problem solution.
- Resources are so small in the target company:
    - Being agile requires a team.
    - People are working in multiple projects simultaneously.
    - In pair programming persona and developing skills are meaningful.
- The model can be even too agile:
    - There needs to be some documentation for the subcontractor.
    - Software is usually long life so you need to know after five years what has been done.
- Documentation should be done in small parts along the implementation.
- How can this kind of project be offered by subcontractor? They can not provide contract by tender so it needs to be done with hour-based invoicing.
- ”It is a big change in the spirit if we change waterfall model into this.”
- ”Scrum or agility in general is quite close to the way we do things here.”

User-centered design

- “We are users by ourselves.”

    - In many cases they make products that are first used for their own purposes. However, usually later they start to sell it for customers.
- ”We have made analyzers for such a long time that we have very strong knowhow of them”.
    - Some of the interviewees did not see end-user's input important since they thought that they already know how to make satisfying and good products.
- Some user feedback is received during the instructions. However, additional discussion with users was seen reasonable.
- At the moment experienced workers in partner companies represent end-users and they are asked for feedback in some projects.
- User interface design and development could be given for subcontractor but calculation algorithms in programs are core competence of the target company.
- Involving customer into the testing is a good thing. It is also beneficial in sales and marketing. However, in this case interviewee discussed about customer, not about end-user.

Agile user-centered design

- The output is definitively better quality in user interfaces but the question is: is there a reason to put so much effort on the user interfaces? Traditionally user interfaces have been quite bare but that has been enough.
    - At the moment there is only industrial designer who has designed user interfaces. That model requires much more resources.
- “In our organization people are just assigned to tasks: now you should do the user interface design”. This model includes more interaction and communication which is reasonable.
- How the model considers possible physical device development?
    - What if mechanical design is done parallel?
    - ”Software designer is not a hardware designer”. Where is hardware designer located in the model?
- User interaction team should at least evaluate user interfaces.

To summarize opinions, the attitude towards agile development was more positive. Especially interviewees liked the communication and interaction in agile methods. In fact, referring to small upfront planning and iterative development they pointed out that they do things agile already now. Instead, involving end-users into design was not seen so beneficial compared to efforts. However, customer involvement was seen useful.

4.6. Second Redesign

Second redesign consisted of model creation based on the interviews and co-design workshop and after that, model modification based on the evaluation session.

4.6.1. First Version of the Process Model

After interviews were analyzed, the actual designing phase started with co-design workshop. Some preliminary thoughts existed before co-design workshop they were not presented in the workshop. Only interview results and user personas were presented for the co-design workshop participants. This was done in order not to affect workshop output. Participants were allowed to comment results and other issues during the presentation. Comments and main points of the discussions were written down and they are summarized in appendix 4.

In co-design workshop participants were divided into two groups and they constructed affinity diagrams from the interview observations. Then they created design solutions for user personas. Both groups chose to highlight observations which solve, ease or affect to the risk user personas' problems. Content of highlighted observations is summarized in following lists. All the proposed solutions with original challenges can be found in appendix 5.

- Agile development
    - Implementation should be done in cycles.
    - There needs to be checkpoints in follow-up.
    - Experiences from previous projects should be utilized.
    - Ready-made basic screen models ease designing.
    - Model should minimize risks.
- User-centered approach
    - Co-operation with customer creates positive working atmosphere.

  - User interfaces could be updated based on the user feedback.
  - End-users should be considered in the software design process.
  - User interface professionals should make user interface design.
- Software development planning
  - There is a need for target positioning, device-, functional and safety specification making.
  - Only small specification should be done beforehand.
  - Business investigation should be done.
  - Work amount estimation is a risk since it hard to estimate work amount beforehand.
- Requirements specification
  - All the stakeholders should be present.
  - There should be a bank for all the requirements. It would available for all the stakeholders all the time.
  - Project manager should finalize specifications.
  - Requirements need to be evaluated against customer requirements before implementation.
- Project organization
  - There should a team to minimize risks.
  - There should a project manager to coordinate working and answer to questions.
  - Subcontractors should be used but there need to be coordinator from the target company.
- Project phasing
  - Need for description: what is required from the developer in different phases.
  - Status meetings are needed during the development.
  - In general project requires target points and review meetings.
- Schedule and resource challenges
  - Lack and unsuccessful distribution of resources is a challenge in the target company.
  - There is also lack of knowledge in the software development in general.
  - People are doing multiple projects at the same time.
  - It is impossible to plan projects beforehand.

- Making good specification of mechanics allows parallel development of the software.

Co-design workshop revealed that documentation is too big issue to handle in this master's thesis work. It is useful to first study documentation practices properly and then to make a suggestion of documentation needs with templates which can be used in documentation. Then employees get practical tools to ease their everyday working. However, profound investigation of documentation targets and needs requires a separate research so it was excluded from research. Also project planning is not widely considered in this research. For instance, process model does not provide any template or model for project planning. This would also require a separate research.

Based on the interview results and co-design workshop output, researcher started to design agile user-centered software design model for the target company.

The target company did not have any model for software design or development, not to mention agile or user-centered model. At latest during the workshop researcher realized that current practices are so far away of agile user-centered design practices that it is useless to propose only a readymade model as a result of this master's thesis. It would have been to big change to do. Instead, it was reasonable to propose development targets which step by step improve standard of activities into more agile and user-centered direction. First version of the step model is presented in figure 19.

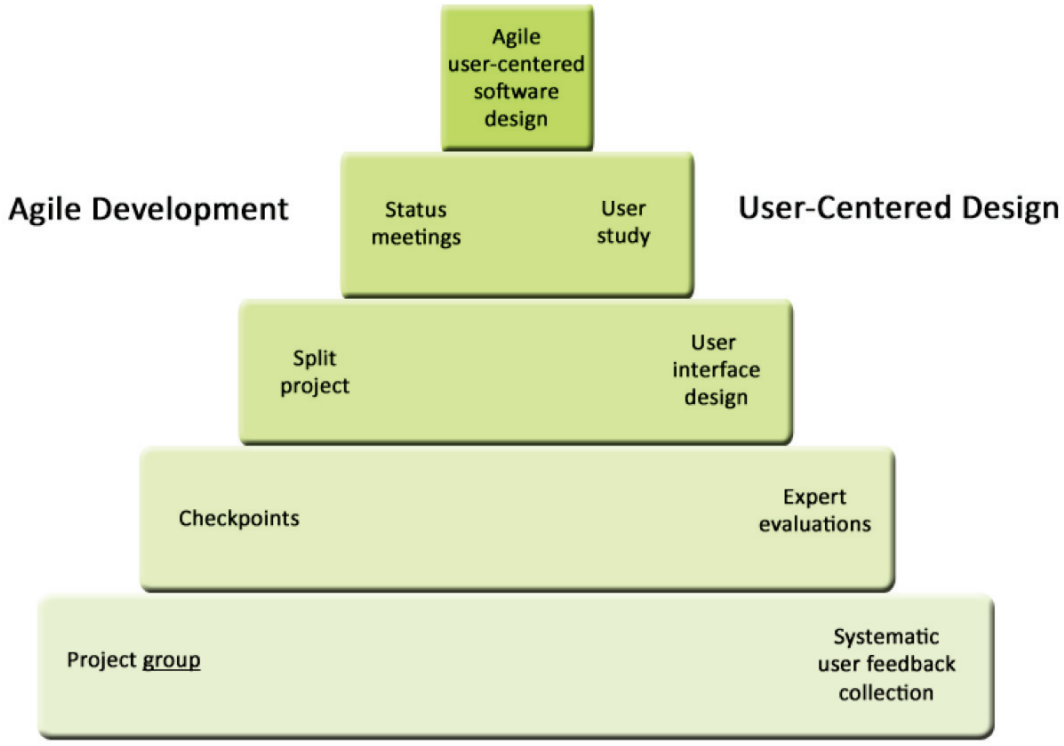


Figure 19 *Step model for increasing agility and user-centered approach*

First two stores are colored with light green while the last three steps are colored with darker green. Slide color is not smooth. This means that only last three steps are really agile and user-centered. Project group, checkpoint meetings, systematic feedback collections and expert evaluations are only preparing activities towards agile and user-centered design. For example making expert evaluations for user interfaces does not mean carrying out user-centered design.

First step in agile development is **Project group**. This means that projects need more than one person. If only one person is doing project it usually means that there is not any follow-up and project is done while other working tasks permit. In other words this step requires project organization to guarantee the proceeding. In most of the cases there should be a project manager to take care of the schedules and quality.

The second step in agile development is **Checkpoints** which stands for checkpoint meetings during the project. Project needs meetings in which different stakeholders (software designers, mechanical designer, electrical designer, project manager, subcontractors, even investigators etc.) communicate and discuss face-to-face about project issues. In practice they go through what every design discipline has been done and what will be done next. If there is a need for changes or even u-turn, all the stakeholders are informed at the same time and obscurities can be discussed. Checkpoint meetings guarantee that project stakeholders know what others are doing and that way conflicts can be avoided. Frequency and stakeholders of the checkpoint meetings depends on the project. However, it is recommended to have a checkpoint meeting at least once in a month.

The third step in agile development is the first step under essential agile development. **Split projects** means that project is divided into cycles and design and implementation is done in cycles. Detailed planning, specification and design are done only for the features decided to be implemented in the next cycle. With split project it is easier to react to changes since everything is not planned beforehand. If something changes developers do not need to update any plans since there does not exist any specified plans for the following phases.

Last step before actual agile user-centered design model is **Status meetings** which are part of the scrum method but definitively not used in the target company earlier. However, status meeting held daily is not reasonable if there are not so many persons from the target company working for the project. Anyhow, status meeting should be held at least once a week. In status meeting stakeholders related to the software development discuss what has been done after last meeting, what will be done next and if there have been problems. Problems are discussed together in the meeting.

The first step towards user-centered design is **Systematic user feedback collection** which is natural extension for the current system in the target company. They get feedback in introductions and maintenance but unfortunately usually feedback does not reach designers. That is why they need somebody to take a response of the feedback collection and processing. Feedback collection should be a planned and systematic process. The responsible should be in

close touch with sales and marketing department since they also get some feedback from the field.

The second step is **Expert evaluations** which mean that usability experts evaluate existing user interfaces and those which are under construction. Then developers make modifications based on the improvement suggestions. Expert evaluations are done by subcontractor until the target company has a usability expert of their own.

The third step which is the first step in essential user-centered design is **User interface design**. This means that professional designers design both user interfaces (wireframe models) and graphical user interfaces. At the moment this needs to be done as subcontracting since there is only one person in the target company with the sufficient background.

The last step before agile user-centered model is **User study**. It is essential in user-centered design but the target company has been in contact with end-users only in few projects. User study step stands for designer familiarizing themselves with user and use context since that reveals real user needs and requirements. User study should be done at least in new product development but also in bigger updates of previous products.

The top of the step model is an agile user-centered design model. Its first version is presented in figure 20. The starting point for the model was the current situation in the target company (figure 17) and some consistency still exists.

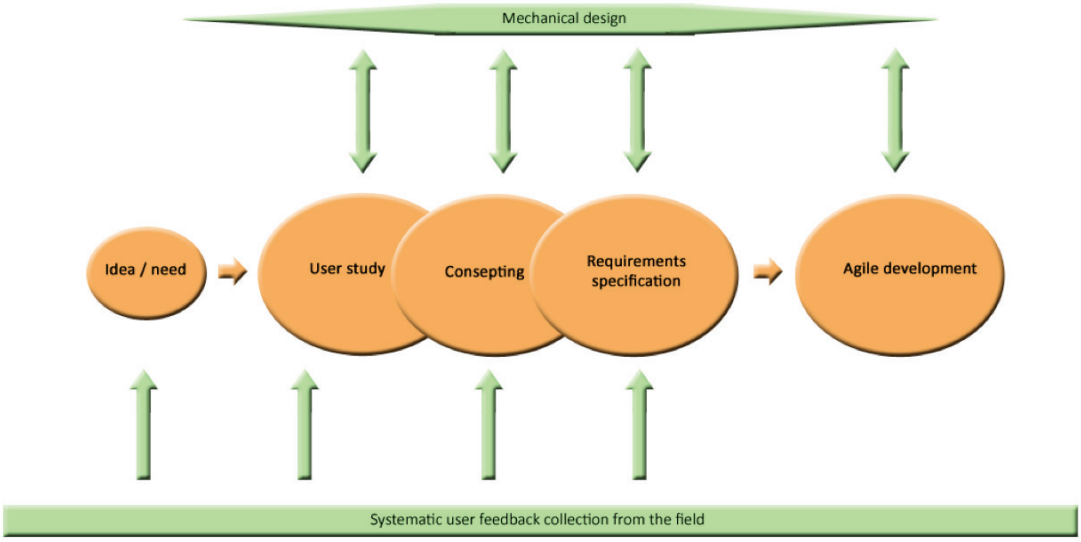


Figure 20 *Software design process model in second redesign*

Systematic user feedback collection is a continuous process. It can give input for idea/need and design phase of the process. Process starts with an idea or a need. Then there is a design and planning phase which consists of user study, concept creation and light requirements specification for the features that are decided to be done based on user study and concept creation phase. These three phases are not separated into different phases for two reasons. Firstly they can be overlapping. For example requirements specification can be user requirements specification which in fact can be part of the user study since it is a result of the user study. Secondly, it would be unreasonable to suggest for the target company that user study should be done in every project from now on. That is because project focus and users vary so much from project to project. In fact, there can be projects in which the user is the developer by himself. That is why the most important thing to understand is that planning and design phase should produce light specification about the features which are known to be implemented at that moment. Usually, concept creation is required to list those features. Concept creation phase in this model answers to following questions:

- What is going to be done with the software?
    - What is the main function?
    - What are additional functions?

- How is it going to be done?
    - What kind of user interface will there be?

In most of the projects answering to previous questions requires an user study. Only end-users know precisely what needs to be done with the software. Instead, the way it is going to be done can be designed by designers. However, users may also have great ideas for that so they should be listened.

The role of the mechanical design is challenging. In this model it is separated from the software design but the main idea is that there is communication connection in each phase. According to research data, usually mechanical design is done before software implementation. If it is not, communication is definitely needed also in actual development phase.

Agile development phase includes agile user interface design and implementation. The model is based on the first redesign combined with process user's needs and opinions. The first version of the model is presented more deeply in figure 21.

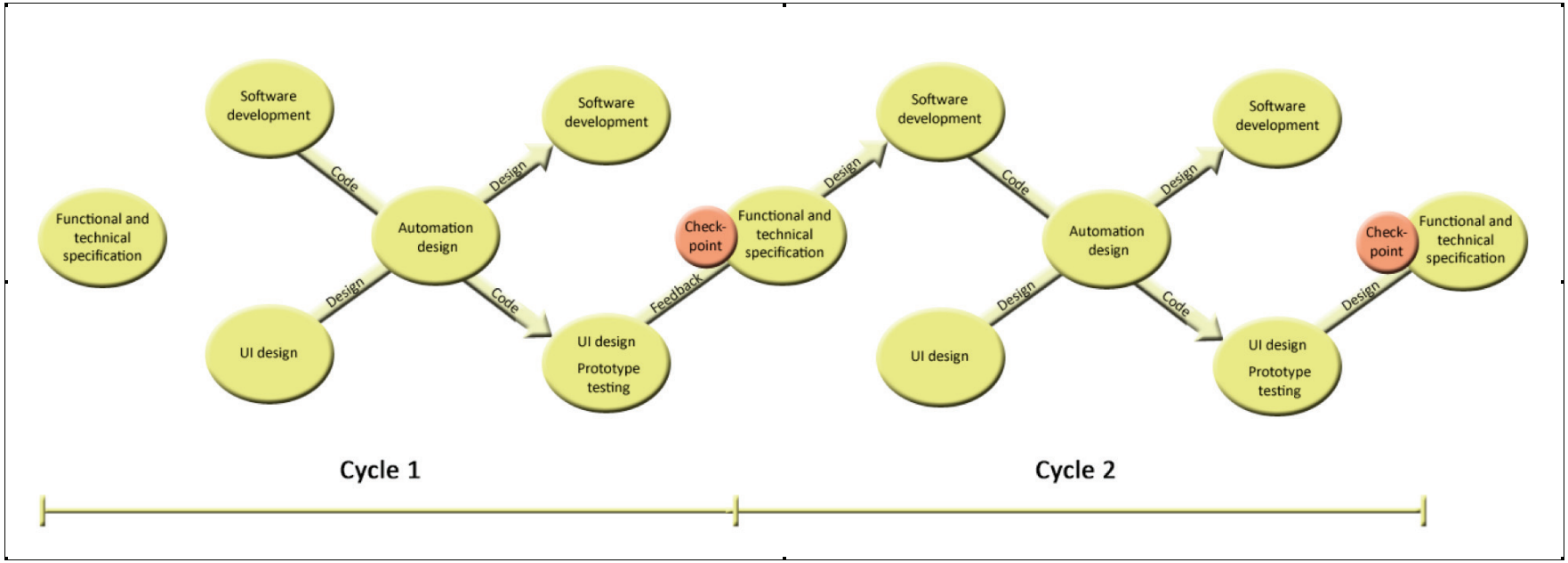


Figure 21 *Agile development in detail*

Development is done in cycles. Each cycle starts with functional and technical specification in which the features chosen to be implemented in that cycle are specified and planned. In cycle one software developers implement features that are not be visible for the user, in other words which does not require any user interface (UI) design. At the same time, UI designers design user interfaces for the chosen features. During one cycle UI designers give designs to software developers and developers give ready piece of product for the usability experts who make prototype testing. Depending on the size of the working software it can be tested only by usability experts or with end-users. End-user tests reveal usually the problems best but expert evaluations are enough if there is not so much software completed yet. The most important thing is that software is tested by someone else than software engineer. However, end-users should test the software at least once during the project.

In this model all the materials go through the "automation design" which stands for the current software developer role in the target company. At the moment the typical situation is that one or two automation designers are responsible for the whole software development, including UI design. In this model automation designer's role is to act as a coordinator between user interaction team (UI design and prototype testing) and software development which is done as subcontracting. Usually automation designer is also aware of the technology process and product history (the target company's core competence) which is needed in software development. That is why all the designs and code goes through them – it is also possible that they participate in design or development at some intense.

In the end of each cycle there is a checkpoint meeting for all the stakeholders. In checkpoint meeting the output of the previous cycle is evaluated and the next cycle is planned. Also the feedback from prototype testing is discussed. If something needs to be changed there is a chance to make changes. Some implemented or only planned features might need to be modified, added or removed based on the checkpoint evaluation. However, this kind of checkpoint after each cycle guarantees that there is a chance to react to changes. After the checkpoint next cycle starts with functional and technical specification for the features chosen to be implemented in the cycle. In cycle two and further cycles software developers can implement only user interfaces since UI designers have made designs already in previous cycles. However, of course there is a chance that those need to be modified based on the checkpoint

meeting changes. Then software developers, again, implement something that is not visible for users. They can also make modifications for the code implemented during the previous cycle if it is needed.

4.6.2. Evaluation Session Feedback

Second redesign was evaluated in evaluation workshop with organization employees, in other words with process model end-users. This kind of evaluation is widely used in user-centered design. Developed ideas are presented for the end-users and they are able to comment and affect ideas already in design phase. Giving feedback later might to be late.

Evaluation session participants (presented more deeply in table 4) were presented step model for increasing agility and user-centered approach in design (figure 19) and agile user-centered software design process model (figure 20 and figure 21). In general, the step model received good acceptance as such. Participants mostly agreed with different steps and they did not have anything to change. The only thing which divided opinions was status meeting: is there a need for a status meeting if there is continuous communication among the project group? From visual point of view it was also discussed that last three stores could be highlighted with more contrast.

Agile development model produced more speculation. For instance the level of planning and specification was discussed a lot, without any clear result. However, the most important thing was the nomination of the coordinator between software development and user interface design. It was called automation design in the model. This can be too limited: there is no use to force that person to be automation designer from the background. However, project manager is also too limited. That role needs to act in the hardware interface. One suggestion was integration design which was less limited since it does not define the background. Automation design was also supported since usually the role is sidekick of the project manager and project manager is usually mechanical designer. In that case, the person between software implementation and user interface design needs to have automation background at least in certain projects.

The meaning of visualization was discussed a lot. Regardless how good new model is, it is not used if people do not understand its content. The second important feedback from the evaluation session was that process model visualization needs to be reconsidered. The structure and phases are good but the presentation needs to more compact and easier to understand. It was also discussed that model should emphasize more collaboration – this version gives the feeling that everything goes only through automation designer while software developers and user interface designers are not allowed to communicate with each other. This kind of approach is not good since the target is to get of the culture in which everybody is just minding their own business. For the linear model, users hoped that checkpoint would be more like a gate, also in visualization. That would emphasize the important role of the checkpoint meetings – they need to be held in order to continue the project.

Complete summary of the evaluation session comments can be found from appendix 6.

4.7. Final Model

Final models were constructed based on the evaluation workshop data and discussions with master's thesis instructor. Step model for increasing agility and user-centered approach in the target company was considered as good by all evaluators. That is why it has not been changed almost at all (figure 22). Contrast between first two and last three stores is bigger. This is supposed to indicate that only last three stores mean agile and user centered design. Having project group, checkpoint meetings, collecting user feedback systemically and evaluating user interfaces with usability experts are only preliminary steps towards agile and user-centered standard of activity. In addition to contrast, also the visual structure was modified. Instead of stores, there are blocks. More closely considered stores would be misleading. For instance Expert evaluations and Checkpoints or User interface design and Split projects are not related to each other. The target company can increase their agility without increasing their user-centered approach or vice versa. Blocks describe better the individuality of step actions.

Contents in the blocks remained the same as in the second redesign version.

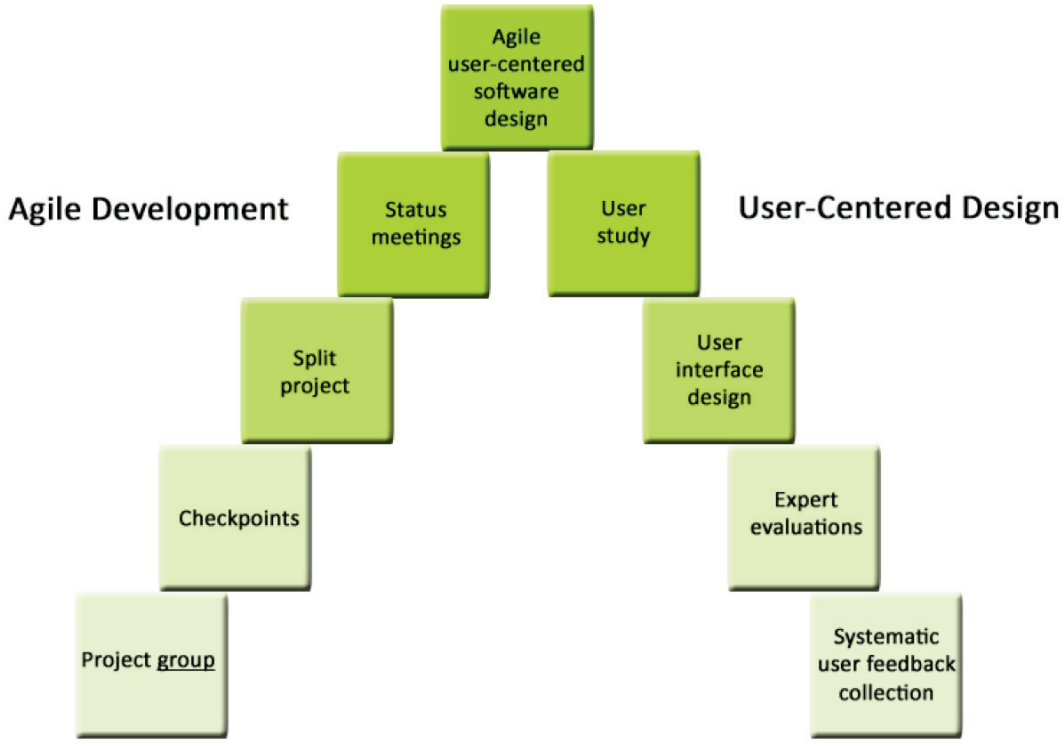


Figure 22 *Final step model for increasing agility and user-centered approach in the target company*

Agile user-centered development process model was designed further based on the evaluation workshop and other discussions. The final model is presented in figure 23.

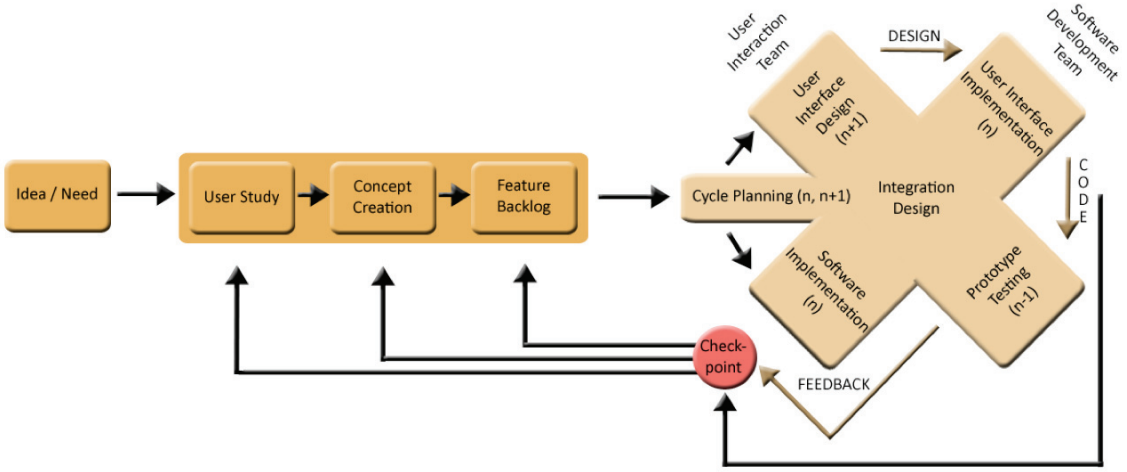


Figure 23 *Final Agile User-Centered Software Design Process Model*

First of all, separate blocks for different roles were forgotten since the target is to get rid of the limited and roles and working tasks. Now there is *User Interaction team* and *Software development team* inside the same pattern. In the middle of the pattern, in the intersection of the teams, there is *Integration design* which refers to the target company's core competence. Integration design (person or a team) has at least the knowledge of the technology process and possible also of product history. In addition, integration design role decides about subcontracting need in different design areas and have contacts to automation, hardware, mechanical and electrical design. In practice, integration design can represent some design area but they do not need to be project or product manager. Based on the user needs, this role should have more hands-on information about software design and development than project managers usually have.

Process starts with an idea or a need. For instance it can be based on the systematic user feedback collection. Before actual software design and implementation, project group needs to decide what they are going to do. Feature backlog contains all the features which are known to be implemented for the software. It is corresponding to Product backlog in the Scrum method (see more chapter 2.1.2.1). However, in most of the cases feature creation should be started with the user study. End-users and product's use context are studied in order to find out end-users' needs and hopes and restrictions and possibilities of the use context. Due to lack of user information collection practices in the target company user study is done in traditional way without dividing it to cycles. User information collection in cycles would be too big step for the target company. It would be also difficult since users are often singular and located even abroad. After user study there is concept creation phase in which following questions are answered:

- What is going to be done with the software?
    - What is the main function?
    - What are additional functions?
- How is it going to be done?
    - What kind of user interface will there be?

Answer for the "What is going to be done with the software?" refers strongly to user study results. Software should be built based on user needs. Once concept creation is done it is easy to construct feature backlog which contains all the features that are known to be implemented, at that moment.

User Study, Concept Creation and Feature Backlog are inside one big block in the process model. This means that all the phases are not compulsory. Phases can also be combined. For instance, if the software is a monitoring software for own purposes and the user is developer by himself, there is no need for user study. Instead, maybe development team creates a feature backlog in the meeting in which they discuss what kind of software they are going to build and what are the purposes software is used for. In that case there is a small concept creation phase along feature backlog creation.

Implementation is done in cycles of which each lasts approximately one month, depending on the project. Each cycle starts with Cycle Planning in which the development and design team should be present. Features to be implemented in the cycle are chosen from the Feature Backlog and they should be chosen in Cycle Planning. Alternatively, they can be chosen in the last cycle's Checkpoint meeting but Checkpoint meeting is meant for all the stakeholders so it might not be reasonable to plan following cycle while there are present people who are not involved in design and development. In Cycle Planning phase the features, their design and implementation are discussed and planned. For instance following questions are answered:

- Which features are going to be implemented?
- What is the order for implementation?
- What kind of specification is needed to be done before implementation?
- Who is going to do and what?
- What kind of user interface design do we need before implementing user interfaces?
    - What was done during last cycle?
    - How do they look like?
    - Do they need modifications?
- How the features implemented during the last cycle are going to be tested?
    - With end-users?

    - With usability experts?
- Which features will be implemented in the next cycle after this?
    - Preliminary planning for the following cycle.

User interaction team and software development team work parallel. In general user interface design is one cycle ahead of the implementation and testing one cycle behind the implementation. However, if there has come out some modification needs in the checkpoint meeting user interaction team starts by finalizing designs for software development team. In case there has emerged radical changes it is possible that they need to be totally redesigned. At the same time software development team is doing modifications for previous cycle features if needed or they implement features which does not need user interface design. If they have nothing else to do than user interface (UI) implementation, but UI designs are not completed, software developers can use place holders in user interface implementation.

Anyhow, the main tasks in the cycle are user interface design for the following cycle and software implementation which uses designs made in the previous cycle. These designs might have been modified or finalized in the beginning of the ongoing cycle. In the end of the cycle user interaction team will also test the features which are implemented during the previous cycle. End-users should be involved in testing at least once during the project to collect feedback. However, end-user testing might not be reasonable if the amount of completed features is not big enough. Then expert evaluation done by user interaction team is sufficient.

Cycle ends with Checkpoint meeting in which all the project stakeholders are present. Before that software developers deliver ready features as code for user interaction team to be ready for the prototype testing in the next cycle. In Checkpoint meeting the completed work in past cycle is discussed:

- What was done?
- What was not done (which was still planned to be done)?
- What kind of problems there were?
- Do we need to change something?
- What was the feedback of the prototype test?

- Do we need to change or update something?

Based on the answers there might be need to change feature backlog. However, it is allowed to change, modify, remove, return or add features in the feature backlog. New cycle starts with cycle planning in which exact features for the upcoming cycle are chosen.

However, if checkpoint meeting reveals radical changes it might be that there is a reason to return to the concept creation phase. Probably the whole software or product requires additional consideration. In addition, changes might reveal the lack of user information. Maybe some changes need to be confirmed by end-users or changes in general require user information which does not exist. Then there is also a need for additional user study.

User Interaction Team and Software Development team should have status meetings inside the cycle if there is more than one person in the team. However, these are not marked in the process model since their frequency, duration and assembly depend on the project. If both User Interaction Team and Software Development team are subcontractors they may not have a meeting together but if they work inside the target company, both teams could very well have a short and common status meeting even more than once a week.

This chapter presented the final versions of step model for increasing agility and user-centered approach and agile user-centered design process model. These models, as such, were also presented for the target company. They will be tested in the future projects. In design research approach that is the learning phase. Unfortunately testing the model is beyond the scope of this master's thesis. However, the project with the customer will continue so the feedback of the model will be collected in the future. The model is not regarded as completed in any cases. Based on the process users' feedback, the model will be modified in the future, like in user-centered design principles and learning phase in design science for organizational development.

5. Conclusions

Agile user-centered software design process was the starting point for the research. Naturally the first thing to explore in the research was the agile and user-centered software design process. Exploration was done as literature review. In order to find out what is an agile user-centered software design process the researcher first explored agile software design process and user-centered design process.

Agile user-centered software design process combines principles from agile methods and user-centered design. In agile methods development is done in cycles. There is a development team who communicates continuously with each other. There is a small up-front planning in which project group lists features that they know that will be implemented for the software. Quite fast after that starts implementation in cycles. In each cycle developers implement piece of the working software which can be reviewed with the customer. That way they do not need to make documentation about proceeding. Also documentation before implementation is left to minimum since all the stakeholders communicate with each other continuously. It is totally acceptable that after one cycle they need to change, remove or add something to the feature list or even to modify implemented code. In fact, it is desirable that changes are responded right after they appear.

In a nutshell user-centered design means that users are involved to the process from the very beginning of the software design and development. Users, their needs and use context are studied in a user study in order to understand what is really important in the software. Prototypes and unfinished versions of the product are tested with users in order to confirm solutions. In addition, professional user interface design is important in order to create usable user interfaces. These activities are also done in cycles in agile user-centered design. User interface design is done one cycle ahead of the development while (prototype) testing is one cycle behind the development. User study or user information collection can also be done in cycles – then it needs to be one cycle ahead of the user interface design since user needs are the base for the user interfaces. However, depending on the situation and project, user information can also be collected before implementation starts. For instance this is practically inevitable if users are located abroad.

Doing everything in cycles is just one version of agile user-centered software design model. For instance usability activities can also be done before agile software implementation. Then it is important that chosen methods are agile or lightweight. For example user and expert evaluations, collaborative usability inspection and user experience questionnaires are considered as lightweight evaluation methods. They are cheaper and faster than usability testing.

Literature review acted as a basis for the essential research. The goal of the research was to develop an agile user-centered software design process for the target company. However, main research question considering the model is answered later since process model development required some background data. Background data was collected by answering to sub questions.

First sub question of the main research question was:

> **How software and user interface development is done in the target company at the moment?**

It was answered based on the interviews. Interviewees described projects they had participated lately. In quite many projects there were similar principles recognizable even though the target company did not have any common software development process model in use. They did not have any common practices for planning, communication, implementing software, follow-up or testing. Every project manager (if there existed) steered the project as they saw fit by experience.

Software projects starts when there is some idea or need for new software. There can be a customer project in which customer have ordered something. In contrast, in research project developers test some new technology or idea. Development is actually quite agile. Not much is planned or documented since almost every time they do something novel. It is hard to estimate how much resources it requires and often there is no time for documenting. Usually in small projects one person just starts to code something and project proceeds among other working tasks. Of course there are also wider projects which have project organization and even

subcontractors. Those are better planned and followed-up but either they do not have any common proceeding model.

Usually they build a prototype in projects. Prototype is tested and in some projects it is tested also in real use context. Unfortunately there are projects in which prototype is regarded as completed project just when it works.

The second sub question of the main research question considered user needs for the process model:

> **What kind of development targets and needs the target company employees have considering the software and user interface development?**

This question was answered mainly based on interviews. The biggest need was to get rid of the projects with only one employee. Employees want to have at least two persons participating in the project but rather real project organization with project manager. Another person is seen to ease working since then you have somebody to discuss with. It also mitigates the risk that nobody knows what has been done if some employee leaves the company.

Agile development practices were mentioned in multiple interviews. In general interviewees wanted planning and requirements specification to be light and implementation to start fast. Some interviewees also mentioned that implementation should be done in cycles. The meaning of feedback was highlighted almost in each interview.

Need for more official follow-up was obvious. Employees want to have checkpoint and status meetings during the project. There has to more target points than just start and end. In checkpoint meetings it is important to get feedback of the made solutions and afterwards make changes if needed.

Involving users in the design process created two kinds of opinions. Some thought that they definitively should collect feedback from end-users for example in the prototype testing. The others thought that user studies are useless effort since the products they are doing do not

need to be excellent. They just need to work. However, multiple interviewees thought that user interface design professionals should design user interfaces instead of their own developers.

Both documentation and testing practices need to be improved. There were not any suggestions how testing should be improved. Documentation was mainly discussed from the code commenting point of view. In general interviewees wanted to have more documentation but it should be light.

Process model was created based on agile user-centered software design process from literature point of view, current situation on the target company and prospective process user's needs. The research question was:

> **What kind of agile and user-centered software design process model is suitable for the target company?**

Research question is answered with figures. Agility and user-centered design approach should be increased in the organization gradually since otherwise the change is quite big compared to the current situation. Step model is presented in figure 24.

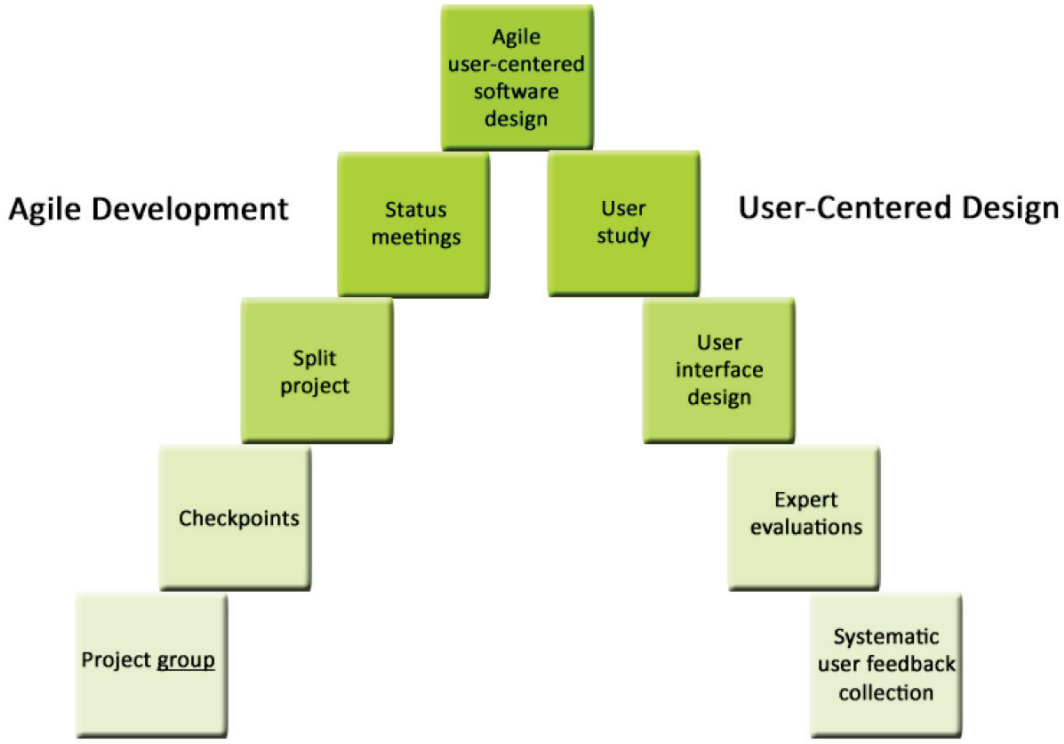


Figure 24 *Step model*

The first step is *Project group* which stands for having more than one person doing the software project. Implementation is faster and developer has somebody to talk with. Also follow-up is easier when at least two persons know about the project. It also reduces the risk of losing project information totally outside the organization. Second step is *Checkpoints* which means that project should have regular checkpoint meeting with all the stakeholders. In that meeting they go through what has been done lately and what will be done next. If there are some challenges or need for change all the stakeholders can discuss about issues face-to-face. The first step in real agile development (green blocks) is *Split projects* which mean that projects are split in cycles and each cycle is implemented at a time. This eases implementation since then there is a focus and also goals for shorter time period. The fourth and the last step before the actual process model is *Status meeting.* This means that developers should have short meetings very often in which they go to through checkpoint issues in smaller scale. Meeting can be even

daily but it should be done at least once a week. Meeting guarantee that people are communicating with each other and they have a chance to discuss about problems.

The first step in user-centered design is improving their current *feedback collection* system. User feedback should be collected in instructions and maintenance but it should be also utilized in software development. Somebody should take care of the feedback collection planning, execution and forwarding. The second step is *Expert evaluations* which stand for evaluating existing user interfaces and user interfaces being under development. Usability experts evaluate interfaces and give improvement suggestions to improve the usability. The first actual step in user-centered design is *User interface design.* This block states that user interfaces should be designed by usability and graphical professionals. The last step before the actual model is *User study* which means that user information is collected before software design in order to find out user needs and requirements.

The last step in the step mode is Agile User-Centered Software Design Process model which is presented in figure 25.

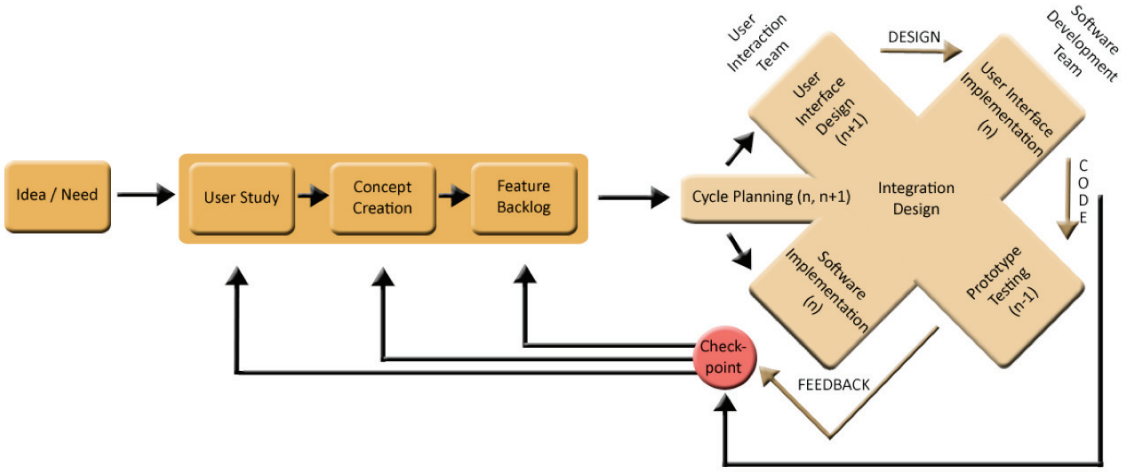


Figure 25 *Agile User-Centered Software Design Process model*

Process model starts with an *idea* or a *need* which leads to a software project. Then there is a *User study* in which users, their needs and use context of the software is researched. User study results are a starting point for *Concept creation* in which team decides what is going to be done

with the software and how. Based on the concept creation it is easy to collect list of all the features that will be implemented (*Feature backlog*).

Implementation is done in cycles. There are two teams: *Software development team* and *User interaction team* and integration design which represent core competence of Outotec. They all participate in cycle planning in which they discuss what features are going to be implemented during the cycle and how they are going to be implemented. They will also do preliminary planning for the following cycle. Then user interface designers of the user interaction team finalize design for this cycle if there has come out some modification needs in checkpoint meeting. At the same time software developers implement something which does not require designs. Right after designs are completed they can start to implement those. User interaction team makes designs for the following cycle. User interaction team also carries out prototype testing in which ready piece of working software from last cycle is tested. Testing can be done with end-users. Once all the tasks are done, cycle is finished with checkpoint meeting. All the project stakeholders discuss about the work done in the last cycle and feedback from the prototype test. Changes are made to feature backlog if needed. If there is a need for bigger changes and executing those needs requires concept creation or even additional user information collection it is possible to return to those phases. Otherwise new cycle starts with cycle planning in which new features are chosen to be implemented in the oncoming cycle and the next cycle.

6. Discussion

6.1. Validity

This research is all about design, even the result is generated via design. One might consider that it is not academic way to do science. There is not any quantitative evidence that the model will be a success. In fact, there is no evidence that it will even work. However, like van Aken (2007: 68) pointed out, with Design Sciences researchers try find out answers for field problems. According to him field problem is a problem regarding the realization of a "better" reality. This kind of a better reality can be only achieved by searching, trying and experimenting. In this case the target company wanted to achieve the "better" reality in the discipline of the software design and development. Researcher has a strong background in usability so it was natural choice to combine user-centered design in the field problem solving. Suddenly researcher found out that this kind of approach had been used also earlier in organizational development. Second redesign of organizational change means that organization employees participate in designing of their new roles, strategies and routines (van Aken 2007: 75).

Even though there is not any quantitative data supporting the success or feasibility of the process model the role of the prospective process users should not be forgotten. From the beginning, process end-users have been involved in the process design. Researcher familiarized herself with the target company current situation with end-users. End-users' hopes, needs and ideas for the software design and development were collected in the interviews. Subsequently end-users of the process designed solutions for software development problems recognized in the interviews. Last input to the design process was evaluation session in which process end-users commented and gave improvement suggestions for the preliminary process model. This kind of design does not guarantee that the model really works but at least it is agreeable for its users and it is definitively made by customizing it for the target company's need.

Even though there is no scientific evidence about the success of the process model one thing is sure. The first redesign of the model which was constructed based on the literature review would have failed in the target company. Roles were too isolated from each other and doing all the things in cycles, in practice without any planning, would have been too big step for the

organization. We should remember that most of the interviewees had not heard about agility and user-centered design before this project. In the end, the final model was quite different than the first redesign and even the second redesign. All the changes were made based on the process end-users' feedback. In these circumstances the benefit of involving users in the design process is unquestionable. In fact, it proves that model is usable for its users and sufficient for the use context.

6.2. Reliability

Master's thesis is quite extensive research but still it is done alone. There is always a risk of the research to become subjective. In a way, it is even emphasized with designing. Design is always somehow subjective: it is a handprint of somebody. Someone may regard the result of designing as good and someone bad. However, involving end-users in the process rescued the researcher from designing everything all by alone. All the tiny details in the model need to be justified with user needs. It does not matter who has invented the idea or detail, the researcher or the user, if end-users agree its relevance and meaning.

Even though designing is not done alone, being subjective can be discussed from another point of view. The target company is huge. It employs over 2600 person in 21 different countries. There were together 21 persons participating in the research, all from Finland. It is self-evident that process model will not be suitable for each software development process in different organizations. However, this process model is supposed to be used in Research and Design departments of Finland, Germany and Australia. It is a pity that interviews with Germany people were cancelled since those would have provided researcher with valuable international aspect. Most of the participants were from the target company's Espoo office so there is a possibility that model is only feasible for Espoo developers. However, that is the situation in user-centered design in general. You are not able to study all the users from all over the world. Anyhow, 19 users from two different business divisions give quite wide perspective about current situation, user needs and hopes for the process model. Usually it is recommended to have together 10-20 people participating in contextual interviews if overall work process is studied (Beyer & Holtzblatt 1998: 76). In these circumstances collected user information is sufficient for generalizing research results.

Starting point for the research was that the model should be agile and user-centered. It is possible that this has steered the design process. When the target is that outcome is agile and user-centered it is easy to concentrate on those issues. In these circumstances it is possible that some other user needs, probably very important, was left with less attention if they were not easily compatible with agile and user-centered model. For the same reason some opinions might have been interpreted to be more supportive against agility and user-centered design than they really were. However, since the starting point was specified by the target company it would have been useless to highlight those rare opinions against agile development. Instead it was important to consider the model as such that it does not contest against their stances.

Designing software design and development process with user-centered design is not common in the field. Even the term design science is not widely spread. Usually software processes are improved by first evaluating the present state and then making changes which, based on the literature, improve the quality of produced software. Agile and user-centered design combined together based on the literature review would have speeded up software development projects and improved the quality. However, model would not have fit in the organization. This kind of organization centered software process design is not researched so much; at least researcher did not find any comparative studies. If these are done, they could have revealed more about the feasibility and success of the process models designed with users. Of course some support of the success was presented in studies presented earlier in this work (van Aken 2007; Börjesson & Mathiassen 2002) but wider research would have been needed to give reliable guarantee of the feasible process model.

6.3. Model Evaluation

Agile user-centered software design process is quite far away from the current standard of activity in the target. Especially user-centered design is almost totally new thing for the R&D department. It might be that the chance is too big. For instance it is possible that people regard user study and concept generation phases as useless and do not want put effort on those. However, to make change easier there is presented also step model for increasing agility and user-centered approach in the organization gradually. If the model is seen troublesome to be applied in some project probably there is a possibility to apply for instance expert evaluations for the user interfaces.

Model created in this research is quite complicated. People need to examine it to understand it. There are multiple roles and tasks and proceeding goes both linearly and in cycles. This can be a disadvantage. People may reject to use model since it seems to be too complicated. In general process models should ease working, not to make it more confusing.

How model will work in practice is tested later. However, some speculations can be discussed already now. Second redesign models and preliminary final models were also presented for two software design professionals from Design Agency Fusion Ltd. In that discussion success of the process in which designing is one cycle ahead of the development and testing is one cycle behind the development was questioned. Fusion has done user interface design and client side implementation in software development projects for several customers. If designing has been ahead of the implementation there have been cases in which design has started to steer to project in general: features to implement are collected from user interface designs (J. Malviniemi 10.7.2009, personal disclosure). In addition, if design is ahead of the implementation it is more difficult to react to accurate changes which appear only along the implementation (J. Malviniemi 10.7.2009, personal disclosure). This might be a problem also in the target company, especially when they may use subcontractors for both software development and user interface design. One option is to reduce design work which is done one cycle ahead of the implementation. Only preliminary designs are done one cycle ahead but practically final design work is done in the ongoing cycle. Then designs are designed and implemented in the same cycle and in the beginning implementation team has to something which does not require designs.

Process improvement is also complicated issue as such. In this research process model is developed first and tested afterwards. In other words process model is just a blueprint on the paper. This kind of approach is dangerous. Like it was written earlier in the research software process is a set of actions people do in reality and best it is described in people's head. Blueprint on the paper is far away from that. That is why it should be reasonable to improve first and model after improvement. However, this model is not completed yet. It will be modified based on the experiences from practice during the learning phase.

6.4. Future research

The most important further research theme is naturally to test how feasible and successful the process model is in practice in the target company. Luckily this will be studied in the near future. Project is continuing even though this master's thesis is done. From the beginning its role was just to create the model with very profound organizational research. However, this issue should be also studied more in general. It would be interesting to know how process models designed with users succeed in general. What kinds of benefits are achieved compared to other process improvement methods? Is all the work worth of results?

Documentation and project planning were beyond the scope of this research. In order to improve software development practices in the target company comprehensively those issues need to be considered. There should be common practices for documentation production: what is documented, when and how. Naturally documentation model should fit together with the agile model created in this research. Also project planning needs a model. Consistent project plan template for agile and user-centered development would ease project managers, and also other team members, in the future. This thesis also excluded technical software implementation practices including testing. To guarantee a good quality of code a research about software implementation practices is needed.

References

Aaen, I. (2002). *Challenging Software Process Improvement by Design*. In Proceedings of Xth European Conference on Information Systems, Gdansk, Poland.

Aaen, I. (2003). *Software Process Improvement: Blueprints versus Recipes*. IEEE Software, vol. 20, no. 5, pp. 86-93.

Abras, C., Maloney-Krichmar, D., Preece, J. *User-centered design*. (2004). In Bainbridge, W. *Berkshire Encyclopedia of Human-Computer Interaction*. Great Barrington, MA: Berkshire Publishing Group, pp. 463–468.

Agile Manifesto. (2001). *Manifesto for agile software development*. Retrieved July 31st 2009, from http://www.agilemanifesto.org.

Bern, A., Pasi, SJA., Nikula, U., Smolander, K. (2007). *Contextual Factors Affecting the Software Development Process - An Initial View*. The Second AIS SIGSAND European Symposium on Systems Analysis and Design. Gdansk, Poland. Jun 2007.

Beyer, H., Holtzblatt, K. (1998). *Contextual Design: Defining Customer-Centered Systems*. Morgan Kaufmann Publishers Inc., San Francisco.

Beyer, H., Holtzblatt, K., Baker, L. (2004). *An Agile Customer-Centered Method: Rapid Contextual Design*. In XP/Agile Universe '04: Extreme Programming and Agile Methods, Calgary, Alberta, Canada, August 15–18 (2004), C. Zannier, H. Erdogmus, and L. Lindstrom, Eds., vol. 3134 of Lecture Notes in Computer Science, Springer, pp. 50–59.

Brandt, E. (2006). *Designing Exploratory Design Games: A Framework for Participation in Participatory Design?* In Proceedings of Participatory Design Conference, ACM Press, Italy (2006).

Boehm, B., Turner, R. (2005). *Management Challenges to Implementing Agile Processes in Traditional Development Organizations*. IEEE Software, vol. 22, no. 5, September 2005, pp. 30-39.

Borjesson, A., Mathiassen, L. (2003). *Making SPI happen: the IDEAL distribution of effort. System Sciences, 2003.* In proceedings of the 36th Annual Hawaii International Conference on: 328-337.

Constantine, L. L. (2002). *Process agility and software usability: Toward lightweight usage-centered design*. Information Age, vol. 8, no. 8. August 2002. Also in L. Constantine (Ed.), *Beyond Chaos: The Expert Edge in Managing Software Development.* Addison-Wesley, Boston, 2001.

Constantine, L..L.., Lockwood, L..A. D. (2002). *Usage-Centered Engineering for Web Applications*. IEEE Software, vol. 19, no. 2., March/April 2002. pp. 42-50.

Cooper, A. (1999). *The inmates are running the asylum.* HW Sams, USA.

Cooper, A., Reimann, R. (2003). *About Face 2.0: The Essentials of Interaction Design*. Wiley Publishing, Inc, Indianapolis, Indiana, USA.

De Micheli, G., Gupta, R. (1997). *Hardware/sofware co-design*. In Proceedings of the IEEE, vol. 85, no. 3, March 1997, pp. 349–365.

Dey, A.K. (2001). *Understanding and Using Context*. J. Personal and Ubiquitous Computing, vol. 5, no. 1, February 2001, pp. 4–7.

Dybå, T., Dingsøyr, T. (2008). *Empirical studies of agile software development: A systematic review*. Information and Software Technology, vol. 50, no. 9, pp. 833–859.

Dybå, T. (2005). *An Empirical Investigation of the Key Factors for Success in Software Process Improvement*. IEEE Transactions on Software Engineering, vol. 31, no. 5, 2005, pp. 410–424.

Erickson, J., Lyytinen, K., Siau. K. (2005). *Agile Modeling, Agile Software Development, and Extreme Programming: The State of Research.* Journal of Database Management, vol. 16, no. 4, pp. 88–100.

Ferreira, J., Noble, J., Biddle, R. (2007). *Agile Development Iterations and UI Design*. In Agile '07: Proceedings of the AGILE 2007 Conference. IEEE Computer Society. Washington, DC, USA. pp. 50–58,

Gellner, M., Forbig, P. (2003). *Extreme Evaluations – Lightweight Evaluations for Software Developers.* In Proceedings of Interact 2003 Workshop - Closing the Gaps: Software Engineering and Human-Computer Interaction. Zürich, Switzerland, September 2003. pp. 75-88.

Guenther, K. (2006). *Developing Personas to Understand User Needs.* (2006). ONLINE, vol. 30, no. 5, pp.49-51, Sep/Oct 2006.

Hackos, J. T., Redish, J. C. (1998). *User and Task Analysis for Interactive Design*. John Wiley & Sons, New York, USA.

Holtzblatt, K., Wendell, J., Wood, S. (2005). *Rapid Contextual Design: A How-to Guide to Key Techniques for User-Centered Design*. Morgan Kaufmann.

ISO 13407. (1999). *Human-centered design processes for interactive systems*. International Organization for Standardization, Geneve.

Kane, D. (2003). *Finding a Place for Discount Usability Engineering in Agile Development*: *Throwing Down the Gauntlet*. Agile Development Conference 2003, pp. 40-46.

Kvan, T. (2000). *Collaborative Design: What Is It?* Automation in Construction, vol. 9, no. 4, pp. 409-415. Elsevier, New York, USA.

Kreitzberg, C. (1998). *The LUCID Design Framework (Logical User-Centered Interaction Design)*. Cognetics Corporation, Princeton, NJ, USA.

Lindvall, M., Basili, V.R., Boehm, B., Costa, P., Dangle K., Shull F., Tesoriero R.,Williams L., Zelkowitz M.V. (2002). *Empirical findings in agile methods.* In Proceedings of Extreme Programmingand Agile Methods—XP/Agile Universe 2002. pp. 197–207.

Mao, J.Y., Vredenburg, K., Smith, P.W and Carey. T. (2005). *The State of User-Centered Design Practice.* Communications of the ACM, vol. 43, no. 3, March 2005, pp. 106-109.

Mattelmaki, T. (2006). *Design probes*. Doctoral dissertation. University of Art and Design Helsinki. Helsinki, Finland.

Martin, R. (2002). *Agile Software Development ,Principles, Patterns and Practices*. Prentice-Hall.

McDonald, S. (2005). *Studying Actions in Context: A Qualitative Shadowing Method for Organizational Research*. Qualitative research, vol. 5, no. 4, pp. 455-473.

McFeeley, B. (1996) *IDEAL: A User's Guide for Software Process Improvement*. Pittsburgh: SEI. Handbook, CMU/SEI-96-HB-001.

McNeill, M. (2006). *User-centered Design in Agile Application Development*.

Memmel, T., Gundelsweiler, F., Reiterer, H. (2007). *Agile Human-Centered Software Engineering*. In Proceedings of the 21st BCS HCI Group conference. British Computer Society, Lancester, UK.

Nielsen, J. (1993). *Usability Engineering*. Academic Press, Boston, USA.

Outotec. 2009. *Outotec in brief.* [Online] Available at: http://www.outotec.com/pages/Page____7536.aspx?epslanguage=EN [Accessed 20 August 2009].

Paulk, M.C., Weber, C.V., Curtis, B., Chrissis, M.B. (1995). *The Capability Maturity Model: Guidelines for Improving the Software Process*. Addison-Wesley, Reading, MA.

Pruitt, J., Adlin, T. (2006). *The Persona Lifecycle: Keeping People in Mind Throughout the Product Design.* Elsevier Inc. (Morgan Kauffman), San Francisco, USA.

Turk D., France R., Rumpe B. (2002). *Limitations of Agile Software Processes*. In Proceedings of 3rd International Conference on eXtreme Programming and Agile Processes in Software Engineering (XP'2002). Alghero, Sardinia, Italy. May 2002. pp. 43-46.

Schilit, B., Adams, N., Want, R. *Context-Aware Computing Applications*. (1994). In Proceedings of the Workshop on Mobile Computing Systems and Applications. Santa Cruz, CA, December 1994. pp. 85–90

Schuler, D., Namioka, A. (1993). *Participatory Design: Principles and Practices.* Lawrence Erlbaum, Hillsdale, New Jersey, USA.

Sanders, E.N. (2006). *Scaffolds for building everyday creativity. In Design for Effective Communications*. In *Design for Effective Communications: Creating Contexts for Clarity and Meaning*. Jorge Frascara (Ed.) Allworth Press, New York.

Simon , H. A. (1969). *The sciences of the artificial*. Cambridge, MA: MIT Press.

Schwaber, K., Beedle, M. (2001). *Agile Software Development with Scrum*. Prentice Hall, New Jersey, US.

Sy, D. (2007). *Adapting Usability Investigations for Agile User-centered Design*. Journal of Usability Studies, vol. 2, no. 3, May 2007, pp. 112–132.

Vaajakallio, K., Mattelmäki, T. (2007). *Collaborative design exploration: Envisioning future practices with make tools*. In Proceedings of the 2007 Conference on Designing Pleasurable Products and Interfaces (pp. 223-238). ACM Press, New York.

Van Aken, J.E. (2005). *Valid knowledge for the professional design of large and complex design processes.* Design Studies, vol. 26, no. 4, July 2005, pp. 379-404.

Van Aken, J. E. (2007). *Design science and organization development interventions: Aligning business and humanistic values*. The Journal of Applied Behavioral Science, vol. 43, no. 1, March 2007, pp. 67-88.

Appendix

Appendix 1: Principles behind the agile manifesto.

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

Appendix 2: Interview questions for different employee roles. Bolded questions had the highest priority.

Interview / Software engineer

Background information
Age
Work experience and education shortly
Work description at the moment

Earlier software development projects

What kind of software development processes have you participated in?

- What was the target in the project?
- Why was the project done? How was the need recognized?
- **How was the project done?**
  - Who did participate? **List all stakeholders and their roles.**
  - What did they do in the project?
  - **How did the project proceed in different phases?**
    - Up-front design
    - Implementation
    - Testing
- How was the project planned?
  - Who was the planner?
  - What kind of a project model was used?
  - How aware stakeholders were about the plan?
- What kind of a schedule was there? How much working time was needed?
- How was the project managed?
  - How the proceeding was followed?
- How was the communication done during the project?
  - Distance of working locations
  - Face-to-face communication, meetings, e-mail
- What kind of documentation was done and what was the format?
  - How useful was it in different phases?
- **How did the project succeed in your opinion?**
  - How was it evaluated?
  - **What good things were there?**
  - How well was the target achieved?
  - **What should have done in different way?**
- How did the project affect to other ongoing or coming projects?
  - Process improvement?

Software development processes

What kind of experiences do you have about pre-refined methods or models for working?

- Do they ease or complicate working? Why?

What kind of software development process models or formalized methods there are in Outotec?
How do people know them?
- How are they documented?

How they are used in software projects?
- Followed step by step or adjusted?
- Different practices in different projects/departments?

What do you think: how should be software development done in Outotec?
- Phases?
- Participants and stakeholders (users)?
- The size of the working group?
- Work distribution?
- Usability design?
- Agile implementation?
- Testing?
- Schedule?
- Follow-up?

- Formalized process?
- Adjustable process?
- No process at all?

Discussion

What do you know about agile software development methods?
- **How do you feel about them?**
- **Have Outotec used them?**
  - How and where?
- **How they could/should be used in Outotec?**

What do you know about user centered design?
- **How do you feel about including user into development process?**
- **Have Outotec used them?**
  - How and where?
- **How they could/should be used in Outotec?**

What kind of process improvement actions have been done lately?
- What kind of a priority these process improvement projects have?
- Need for software process improvements?

Present the project.

Organizational context

How does your company's strategy affect your everyday working?
What do you think about company's quality attitude?

- How does it affect your everyday working?
- Is there something you would like to change?

Customer (also inner) / Subcontractor

What kind of experiences do you have about following issues?

- Customer requirements?
  - Delivery?
  - Quality?
- Customer involvement in the process?
- Customer project management?
- Subcontractor involvement in the process?
- Subcontractor project management?

Operational management

How are projects, resources and schedules managed in your company?

- Any improvement ideas?

How are projects, resources and schedules managed in your customer?

- Any improvement ideas?

Have you had multiple roles in multiple projects at the same time?

- How do you feel about that?

Interview / Manager

Background information

Age
Work experience and education shortly
Work description at the moment

Earlier software development projects

What kind of software development processes have you participated in?

- Target
- Stakeholders and their roles
- Proceeding
- Success / targets for improvement

Software development processes

What kind of experiences do you have about pre-refined methods or models for working?

- Do they ease or complicate working? Why?
- What kind of formalized methods or processes there are in Outotec in general?

What kind of software development process models there are in Outotec?
How do people know them?

- How are they documented?

How they are used in software projects?

- Followed step by step or adjusted?
- Different practices in different projects/departments?

What do you think: how should be software development done in Outotec?

- Phases?
- Participants and stakeholders (users)?
- The size of the working group?
- Work distribution?
- Usability design?
- Agile implementation?
- Testing?
- Schedule?
- Follow-up?

- Formalized process?
- Adjustable process?
- No process at all?

Discussion

What do you know about agile software development methods?

- **How do you feel about them?**
- **Have Outotec used them?**
  - How and where?
- **How they could/should be used in Outotec?**

What do you know about user centered design?

- **How do you feel about including user into development process?**
- **Have Outotec used them?**
  - How and where?
- **How they could/should be used in Outotec?**

What kind of process improvement actions have been done lately?

- What kind of a priority these process improvement projects have?
- Need for software process improvements?

Present the project.

Organizational context

Business environment

Describe the business environment from the software design point of view:

- Level of competition
- Level of trust
- Attitude to subcontracting
- Cost level
- Language barriers

- Physical distance

How does business environment affect software development?

Subcontractor (Delta Enterprise) / Customer

What kind of experiences do you have about following issues?

- Subcontractor involvement in the process?
- Subcontractor project management?
- Possible customer requirements?
  - Delivery?
  - Quality?

Strategic management

- How does the core strategy of the company affect software development projects?
- Are there any changes coming in the near future?
- How do you think they will affect software development?
- How do you inform workers about the company strategy?
- **How aware are software developers of the company strategy?**
- **How does it affect to their everyday working?**

Company infrastructure

What kind of working tools there are available?

- What if something else is needed?

What kind of working training there is available?

- What if something else is needed?

Organizational culture

What kind of a quality attitude there is?

- How high is quality appreciated in organization?
- What is the attitude towards iterative quality generation?

What kind of process improvement actions have been done lately?

- Is something in planning level?
- What kind of a priority these process improvement projects have?

How people can affect their own work?

- Working tasks: what is done and how
- Changing working practices

Organizational structure

How is the work and role distribution done in different projects?

- Who makes it? Can designers affect?
- Can people participate in multiple projects and/roles at the same time?
- How does this affect project management?

How aware people are of software engineering practices in the company?
How the knowledge about practices is spread through the organization?

Competence

Describe your strengths and weaknesses (competence) related to software/product development:

- Level of software design / engineering skills
- Level of working experience
  - Senior/junior designers
  - Work force changes
- Level of business and domain knowledge

Describe strengths and weaknesses of your subcontractors (competence) or customers.

Operational management

How are projects, resources and schedules managed in your company?

- Any improvement ideas?

How are projects, resources and schedules managed by your stakeholders?

- Any improvement ideas?

Knowledge transfer

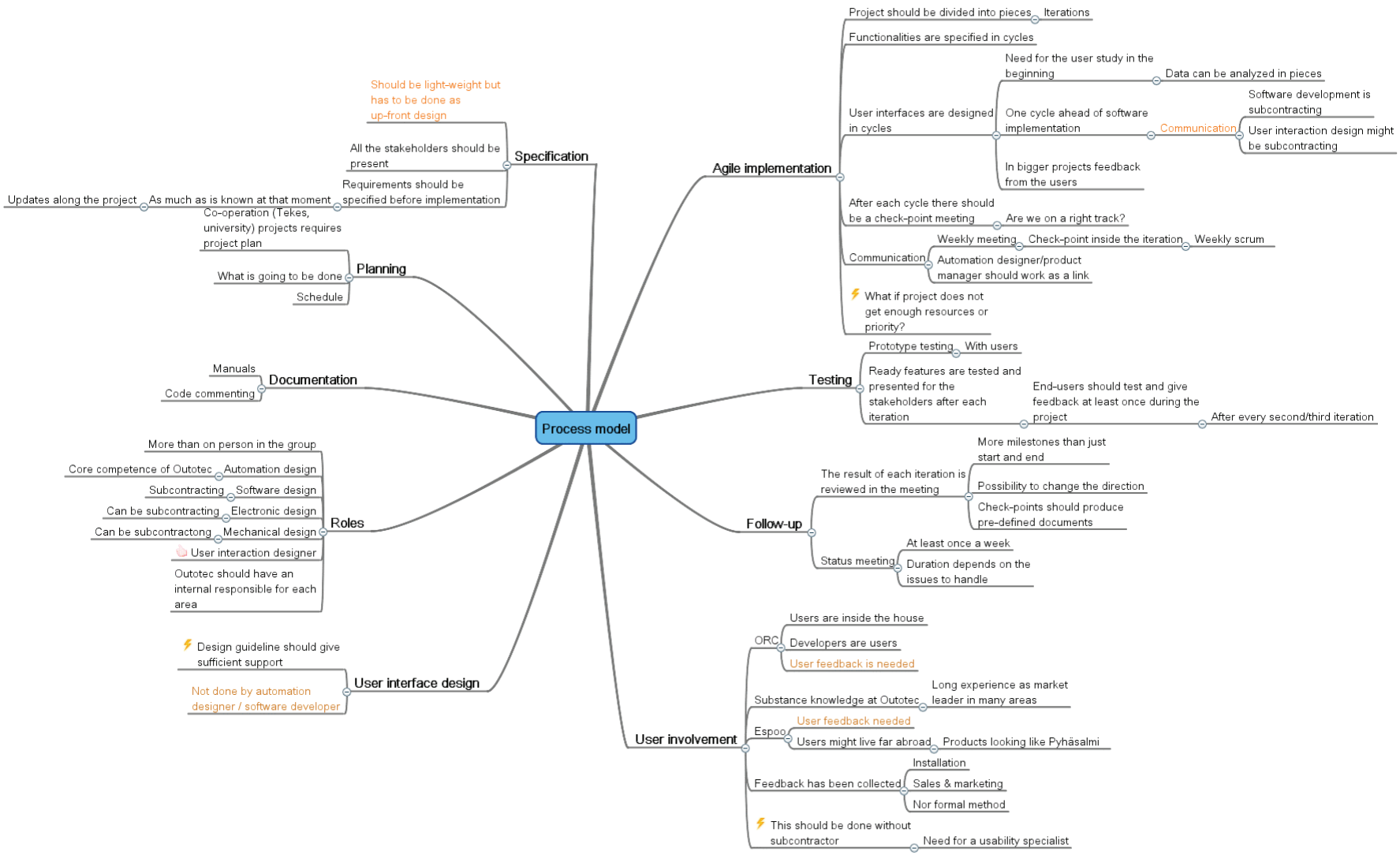
How do you communicate in Outotec?
How sufficient and efficient is that communication?

- How could it be improved?

How do you communicate with subcontractors and other stakeholders?
How sufficient and efficient is that communication?

- How could it be improved?

Appendix 3: Mind map of all notable issues in process creation.



Appendix 4: Comments considering interview results presented in co-design workshop.

Software design and development process

- If there is only one person in the project the knowledge is only in his head. Usually the knowledge is not documented.
- "I had a project in Minerals Processing. When I changed to Base Metals the project was forgotten."
- "We need to have process in order to know how to proceed."
- There needs to be some kind of project control if developer is project manager by themselves.
- "Employees are designed to steer the project even though they have no experience. That is dangerous."

Physical device related to the software

- "Mechanical designers have to make product specification before software development can proceed."
- In device oriented development you can not proceed separately.
- There is no possible for prototype testing → there will not exist physical hardware even though mechanical design would proceed.
- Simultaneous planning is important. Naturally mechanical is little bit ahead.
- "If product has been decided to produce mechanical design has already proceeded → end-users should be involved right away."

Subcontractors

- Subcontractors want to remain waterfall model.
- The target company should be able to be to in straight contact with subcontractor developer.
- If there is toll-free billing there has to be profound specification. This does not fit in agile development. Of course it is possible that subcontractor makes agile implementation.
- Should the review be different when subcontractors are participating compared to internal reviews?

Product development practices in general

- "People is US was wondering that we get money from doors and windows even though we do not have anything to sell in shelves."
- Usually customer project means that a prototype is built during the project. In other words they do not have any products when they sell it.
- "We should consider such products in which the series size."
- The problem is that everything is done in customer projects – there should be more internal research or development projects done before the product is sold for the customer.
- In some cases product development project in customer project is like swear word.
- Most if the internal projects are in further development projects.
- In general there is only a little strategic product development without customer project. However, some good examples about product development projects without customer still exists.

- The problem is in sales and marketing: product can not be developed without purchaser but nothing can not be sold without a concept.
- Software is not considered as own product → it is done with minimal effort. However, the interest in doing better software is increasing all the time.
- "Our problem is technology oriented development. It should be user-centered".
-

Organizational development in general

- "How do we inform our management about these important results?"
- Too big changes probably are hard to implement.
- There are a lot of improvement targets in general – how they are prioritized?

Appendix 5: Solution suggestions for challenges of user personas. Suggestions are collected from affinity diagrams and they are grouped under user personas' challenges. Observations with grey background are presented in the research.

Developer plans, designs and implements everything alone	**User interfaces are done by engineer**	**Project does not have schedule, except to become ready as fast as possible**	**Project manager is responsible for two projects**	**Project does not proceed since developers have other customer projects.**	**Software development should proceed despite of mechanical design problems**
All the stakeholders should be present once requirements are discussed.					
There should be at least two persons in each project: firstly, it is not a catastrophe if somebody leaves and secondly, you have somebody to discuss with.			RISK: The amount of work is difficult to estimate in software development. Estimation is hardly never correct.		
It is impossible to find all resources inside the company -> we need subcontracting.			PROBLEM: These projects include always research, studying and learning which are not considered in schedules.		
The model should emphasize what is needed from the software developer in the beginning, during and in the end of the process.	We should utilize feedback and experience from the projects in which the product's previous version has been used.	RISK: Functional specification was not finalized -> this caused problems when employee left the company.	The biggest problem in planning is budgeting of schedules and resources.		
Planning started with target positioning: making device, functional and safety specification.	It would nice to have basic model for screens -> then design do not need to be started from the scratch.	Documentation was done by collecting issues into Power Point file.	New process model should minimize risks.		
Beforehand there was only light planning. Functionalities and technology were specified also light just before implementation.	Co-operation with customer create more positive product development environment.	When we made a good planning and documentation, automation and software development did not need to wait mechanical design to be ready.	SHOULD BE: Project proceeding should be evaluated against project plan.		
All the requirements should be written in one file and they should be generated together with whole team. Otherwise they appear one by one.	Sometimes customer is asked feedback about the product.	End-users need to be considered in the project but it requires effort.	The frequency of follow-up meetings depends on the project state.		
Model should instruct what kind of documentation is done one how.	In one project user interfaces were redesigned based on the user feedback.	Communication between stakeholders is slow.	There does not exist any official model for project planning.		
There is no project plan for further development project.	How software is really used can be only seen from the end-users.	Project proceeding and success should be evaluated against the project plan.	Project plan is based on experience.		

Functional and technical specifications need to be evaluated against customer requirements before implementation is started.	Testing routines need to be improved.	The target was to have weekly meetings in the project. In the end it was just discussion in the corridors.	RISK: It is not possible to use huge resources on testing since sales numbers are so small and different customers order different version.		
Final documentation should be done along coding and code commenting.	Industrial designer has designed user interfaces.	The meaning of the communication is highlighted, if documentation is light.	Project stakeholders are usually software, mechanical and electrical designer, project/product manager and in some case, also sales representative.		
There should more documentation but it should be light.	Professional user interface designer should take care of the look and feel also.	Communicating with subcontractor is done mainly by e-mail (Pori).	The role of the project manager is to coordinate co-operation of different stakeholders.		
Feedback about the previous version of the product can be asked from the maintenance and instruction people.	Professional user interface designer should design user interfaces.	There should more target points than just start and end.	We need project organization with project manager to take care of the quality.		
The role of the project manager is to coordinate co-operation of different stakeholders.		There is no project plan for further development project.	Project manager should participate at least in planning and he needs to have knowledge about technology and technology process.		
More than on person is needed for the project in case the worker leaves.		Implementation should be done in cycles.	Project manager's role is act as a link between software developer and end-user.		
If there was more than person developing the product, more than on person also knows about it afterwards.		Project plan was master's thesis plan.	Mechanical, electrical and software design needs a representative from the company -> they decide if subcontracting is needed.	Implementation should be done in cycles.	There should be standard interface for software architecture -> same components could be used everywhere.
We need project organization with project manager to take care of the quality.		Implementation phase should be followed-up for instance with reviews.		Follow-up needs checkpoints: this is working , this is not working, now we need an u-turn.	Projects are changing all the time so planning is difficult.
One big challenge in the company is the lack of knowledge for software development.		Projects require status meetings in which current situation and possible problems are discussed.		It should be considered in specification phase if subcontracting can be used (also in services).	Project manager finalizes specification.
			Proper business investigation helps to find issues which support sales and marketing.	CURRENT SITUATION: Month after a month project is postponed. Once a year date is changed to plan since nothing is done even though people are asked to do the work.	There should more documentation but it should be light.

				Project has proceeded slowly since there have simultaneous customer projects which have taken all the resources.	The meaning of the communication is highlighted, if documentation is light.
				Project needs meeting at least once a week at certain time in certain place. Duration depends on the issues to handle.	
				It is quite normal in the company that schedules are slipping. The main reason is lack of resources but also a low level of knowhow affects.	Professional user interface designer should take care of the look and feel also.
				There should have been subcontractor from the beginning since own employees did not seem to have time.	Implementation phase should be followed-up for instance with reviews.
			Software development uses subcontracting.		
			Using subcontracting requires a lot up-front planning and specifications unless subcontractors are not working at company premises.		

Appendix 6: Evaluation workshop comments considering step model and agile user-centered design model. Texts in quotation marks are citations.

Project group

- If they would have user interface designer and graphical designer in projects they would already have a project group.
- Also hardware designer from subcontractor raise the group size.
- "If there was be two persons developing software it would ease the project in any case – even though partner did not have any solutions or ideas for the problems. Usually if you just have a chance to discuss with somebody you will invent the solution by yourself".
- More than one person in project group inside the target company raises motivation to make good quality but does it affect to subcontractor quality?
- Project manager is needed to take care of the schedules and quality.
- Project manager should know more than just technology process. He should know about design and development. In some projects project manager is just one of the designers or developers. However, there was one project in which project manager and he did not know anything about software development.

Meetings

- In Pori they do not have meetings with all the stakeholders. They only communicate among the own group or working partner. However, this approach was criticized. "The problem in our organization is that people do their own things own way in their own desk. That way they do not want anybody to interfere with their job".
- It would also be reasonable to organize such meeting in Pori but once a month is too complicated since research partners might be even abroad.
- "If there is continuous communicating inside the project, is it really reasonable to organize status meetings?"

Split projects

- Splitting sounds reasonable. "There have been so many examples when things are done multiple times".
- Only main directions should be planned beforehand and then those are split into smaller entities which are done in parts.

- Requirements versioning has been tested in one project.

Systematic feedback collection

- There is a quality manager who is responsible for feedback collection and forwarding. However, nobody has heard anything about him.
- At the moment there is a possible to send feedback via webpage. However, feedback sending via web page was not found fitting in this context.
- They suggested that there could be an extranet where customers could report about their problems.
- However, it has highlighted that user and customer is not the same person.

User interface design

- User interface designer and graphical designer should be open resource in the organization. Only then they would be used in projects. Otherwise money is not budgeted for user interface and graphical design.
- Sometimes user interface is done last. If it was done first the software would be easier specified.
- In addition to graphics also user interface vocabulary should be designed. It is important to know what the users' vocabulary is.
- Also engineer should be involved in UI design since they have the knowledge of the technology process and product history.

Agility in general

- Sometimes things are done even too agile since there is not done any documentation.
- Also there have been situations that features are listed on by one. If developer had known new features earlier he would have other features different from the start.
- "We need to have some agreed level for planning and specification making. The question is what the level is?"
- One option is that planning is always made two weeks ahead.
- "In worst case requirements are read from product brochure". This means that sales and marketing has been too agile in their discipline.

Concept generation

- In one project they wanted to make training software. Usually projects are very technologically oriented: how we are going to make education software? However,

industrial designer wanted that they would concept education software more: Where else it could be used? What kind of additional value could be brought for the user? This kind not technologically oriented concept generation is needed in their projects.

Roles in agile development

- "It is wonderful that automation designer is located in the middle of the process".
- However, the nomination may not be correct. It is a bit misleading.
- Nomination was discussed a lot and no exact solution was found.
- Integration design sounded most reasonable since the role is to act in hardware interface.
- On the other hand automation designer is sidekick of the project manager and project manager is many times mechanical designer – from that point of view automation designer for the coordinator is quite reasonable.

Working tools

- They did not know if Fanuk automation can be developed simultaneously by multiple persons.
- Also some other tools did not allow multiple workers development at the same time.
- Somebody worried if it is possible to make nice user interfaces with WinCC tools. However, industrial designer has tested that Photoshop files can be repeated with WinCC.

Others

- Even though there will be process model with roles it should be remembered that people like to do decisions about their own work. Until now they have been able to decide and do everything by themselves.
- At the moment the culture is that there are closed functions with closed job descriptions.
- However, employees need to allow that other workers participate also in their job. This is strongly related to the attitude towards checkpoint meetings. Results can be evaluated and commented public.
- In most of the projects mechanical design is ready before software is started. Unfortunately it is quite technology driven approach.

- Visual outlook of the agile development model needs to be considered. Content is quite ok but at least visualization needs concept generation.
- Checkpoints should be more like gates, also in visualization.