Aalto University
School of Science
Degree Programme of Computer Science and Engineering

Joakim Gunst

Introducing interaction design in agile software development

Master's Thesis
Espoo, March 30, 2012

Supervisor: Professor Marko Nieminen
Instructor: Tuomas Tolvanen, M.Sc. (Tech.)

Aalto University
School of Science
Degree Programme of Computer Science and Engineering

ABSTRACT OF
MASTER'S THESIS

Author: Joakim Gunst

Title:
Introducing interaction design in agile software development

Date: March 30, 2012 **Pages:** 92

Professorship: Usability and User Interfaces **Code:** T-121

Supervisor: Professor Marko Nieminen

Instructor: Tuomas Tolvanen, M.Sc. (Tech.)

This thesis examines the question of how best to introduce interaction design into the agile software development process. During the past decade, both agile software development and interaction design have become increasingly popular due to the various benefits they bestow. Agile software development increases the visibility and adaptability of projects through the frequent delivery of software in small increments, while interaction design improves the usability of products through a focus on observing and understanding end users. Although the two approaches share some similarities, questions remain about how best to integrate them.

The research was conducted via action research at a Finnish software company. The current state of interaction design and usability was analyzed and new interaction design practices were tested in a project. The tested practices included iterative paper prototyping before development and usability testing of the product. Data was collected through interviews and observation. The results indicate that there were both weaknesses and strengths in the software development process concerning usability and that the new interaction design practices could for the most part be successfully introduced. Based on the results, the thesis presents six recommended practices for introducing interaction design at the company.

Keywords: usability, agile software development, interaction design, paper prototyping, usability testing

Language: English

Aalto-yliopisto
Perustieteiden korkeakoulu

Aalto-yliopisto
Perustieteiden korkeakoulu
Tietotekniikan tutkinto-ohjelma

DIPLOMITYÖN TIIVISTELMÄ

Tekijä:	Joakim Gunst		
Työn nimi:	Interaktiosuunnittelun käyttöönotto ketterässä ohjelmistokehityksessä		
Päiväys:	30. maaliskuuta 2012	**Sivumäärä:**	92
Professuuri:	Käyttöliittymät ja käytettävyys	**Koodi:**	T-121
Valvoja:	Professori Marko Nieminen		
Ohjaaja:	Diplomi-insinööri Tuomas Tolvanen		

Tämä diplomityö tutkii kysymystä, miten interaktiosuunnittelua voidaan parhaiten ottaa käyttöön ketterässä ohjelmistokehitysprosessissa. Viimeisen vuosikymmenen aikana sekä ketterä ohjelmistokehitys että interaktiosuunnittelu ovat kasvattaneet suosiotaan niistä saatavien hyötyjen ansiosta. Ketterä ohjelmistokehitys lisää projektien näkyvyyttä ja sopeutuvuutta tuottamalla uusia ohjelmistoversioita tiheällä tahdilla, kun taas interaktiosuunnittelu parantaa tuotteiden käytettävyyttä keskittymällä loppukäyttäjien havainnointiin ja ymmärtämiseen. Vaikka molemmissa lähestymistavoissa on yhtäläisyyksiä, on niiden onnistuneessa yhdistämisessä vielä avoimia kysymyksiä.

Tutkimus suoritettiin toimintatutkimuksena suomalaisessa ohjelmistoyrityksessä. Interaktiosuunnittelun ja käytettävyyden nykyistä tilaa analysoitiin ja uusia interaktiosuunnittelukäytäntöjä testattiin projektissa. Testattavat käytännöt sisälsivät iteratiivista paperiprototypointia ennen kehitystä sekä tuotteen käytettävyystestausta. Tietoja kerättiin haastatteluilla ja havainnoinnilla. Tulokset osoittavat, että ohjelmistokehitysprosessissa oli sekä heikkouksia että vahvuuksia liittyen käytettävyyteen ja että uudet interaktiosuunnittelukäytännöt olivat suurimmaksi osaksi mahdollisia ottaa käyttöön onnistuneesti. Tulosten perusteella diplomityö esittää kuusi ehdotettua käytäntöä interaktiosuunnittelun käyttöönottamiseksi tutkitussa yrityksessä.

Asiasanat:	käytettävyys, ketterä ohjelmistokehitys, interaktiosuunnittelu, paperiprototypointi, käytettävyystestaus
Kieli:	Englanti

Aalto-universitetet
Högskolan för teknikvetenskaper

Aalto-universitetet
Högskolan för teknikvetenskaper
Examensprogram för datateknik

SAMMANDRAG AV
DIPLOMARBETET

Utfört av:	Joakim Gunst		
Arbetets namn:	Introducering av interaktionsdesign i agil programutveckling		
Datum:	30 mars 2012	**Sidantal:**	92
Professur:	Användargränssnitt och användbar.	**Kod:**	T-121
Övervakare:	Professor Marko Nieminen		
Handledare:	Diplomingenjör Tuomas Tolvanen		

Detta diplomarbete undersöker frågan hur interaktionsdesign bäst kan introduceras i den agila programutvecklingsprocessen. Under det senaste årtiondet har både agil programutveckling och interaktionsdesign ökat i popularitet på grund av de olika fördelar de medför. Agil programutveckling ökar synligheten och anpassningsförmågan hos projekt genom produktion av program i små, frekventa inkrement, medan interaktionsdesign förbättrar användbarheten hos produkter genom fokusering på att observera och förstå slutanvändare. Även om båda tillvägagångssätten delar vissa likheter, kvarstår frågor om hur de bäst kan integreras.

Forskningen bedrevs genom aktionsforskning i ett finskt programvaruföretag. Interaktionsdesignens och användbarhetens nuvarande tillstånd analyserades och nya interaktionsdesignmetoder testades i ett projekt. I de nya metoderna ingick iterativ pappersprototypning före utveckling samt användbarhetstestning av produkten. Data samlades genom intervjuer och observation. Resultaten visar att det fanns både svagheter och styrkor i programutvecklingsprocessen vad beträffar användbarhet och att de nya interaktionsdesignmetoderna till största delen lyckat kunde tas i bruk. På basis av resultaten presenterar diplomarbetet sex rekommenderade praxis för introducering av interaktionsdesign i företaget.

Nyckelord:	användbarhet, agil programutveckling, interaktionsdesign, pappersprototypning, användbarhetstestning
Språk:	Engelska

Acknowledgments

First and foremost, I would like to thank Professor Marko Nieminen for supervising my work and, especially, for giving clear and practical advice on the goals, scope and methods of the thesis as well as the research and writing process. I would also like to thank my instructor Tuomas Tolvanen for the support and good advice he has provided throughout the project, as well as Maija Pero for helping with practical arrangements. I am very grateful to Rapal Oy for giving me the opportunity to do the research as part of my work at the company. Finally, I would like to thank my colleagues at Rapal Oy, especially Jari Turunen, Heli Etuaro and Juha Luotio, for supporting me in the research and for helping with the arrangements that made the end result possible.

Espoo, March 30, 2012

Joakim Gunst

Contents

Chapter 1

Introduction

1.1 Introduction

Software development is an inherently complex activity, and the various needs and motivations of customers, users, executives, developers and other stakeholders contribute to this complexity. For a long time, careful planning and rigorous engineering practices were seen as a way to make this complexity manageable. But during the past decade, companies have increasingly realized that most software requirements are unpredictable, and people seldom know what they need or want before they see it before them. Agile software development, where working software is developed iteratively in small increments, has emerged and gained widespread popularity as a solution to this problem. (e.g. Cohn, 2010; Leffingwell, 2011)

At the same time, another tradition has also gained popularity within software development. Known by many names, it is now often called interaction design, and it puts the goals and needs of the end-users of the software in focus. While all software development strives to create products that are useful, interaction design in addition strives to create products that are usable, i.e. that can be used effectively, efficiently and with satisfaction by the end users. In highly competitive markets, including much consumer software, a focus on usability has already become essential for product success. Now its importance is also growing in other markets, such as enterprise software. (e.g. Cooper, Reimann, and Cronin, 2007)

While agile methods and interaction design share some of the same goals and methods, there are also significant differences in approach. Agile development iterates on working code and gets feedback from customers or their representatives, while interaction design iterates on prototypes and gets feedback from end-users. Interaction design puts an emphasis on

understanding users and designing solutions before development begins, while agile methods strive to avoid up-front design as much as possible. Reconciling these differences successfully has been an important research topic during the past few years. It is also the topic of this thesis.

1.2 Research question and scope

This thesis approaches the topic from the point of view of a small Finnish company producing enterprise software for the management of facilities, real estate and infrastructure. While the company adopted agile software development a few years ago, there has not been any explicit focus on usability or interaction design. Recently, a goal of the company has been to put more effort into improving the user experience of the products it offers. This thesis is part of that effort, and has the goal of answering the following research question:

> *What practices should be introduced into the agile software development process at the company in order to improve the usability of the products it develops?*

The research question can be understood better by discussing it in parts. First, *agile software development* is assumed. This means that the research will not seek answers that are not at least to some extent compatible with agile methods. The research will focus on agile methods in general rather than any specific methodology. However, because Scrum is both the most widely used agile methodology and the one in use at the company, it will serve as the primary reference point when considering agile methods. Agile software development is discussed in Section 2.2.

Second, the goal of introducing new practices is to improve the *usability* of the products developed by the company. The research focuses on improving usability rather than on improving user experience, because usability is a better-defined concept. Usability also makes up a large part of the overall user experience, especially in enterprise software. Usability and interaction design is discussed in Section 2.3.

Third, for the purposes of this research, *practices* can be divided into three types. Team practices, discussed in Section 2.4, consider how teams should be structured and responsibility divided. Process practices, discussed in Section 2.5, consider how the development process should be structured. Activity practices, discussed in Section 2.6, consider which individual activities should be selected and how they should be performed. The research considers all three kinds of practices.

Fourth, the *measures* by which practices are judged are *how well they fit* into the agile development process at the company and *the perceived benefits of adopting them* by different stakeholders. The goal of the research is therefore not to measure how different practices actually improve the usability of products. Rather, the practices that are selected are assumed to improve usability based on the literature.

The research question is important not only for the company where the research is conducted but also for other companies in similar circumstances. Many companies have adopted agile methods during the past decade for various reasons, but improving usability is rarely one of them (see Section 2.2.2). When usability increases in importance, companies with little prior knowledge of usability or interaction design are unsure about how to make this new priority compatible with their development process, especially since the agile literature provides little guidance. This thesis gives one answer to the question, with the hope that it will contribute to a future where more software, including enterprise software, is easy and enjoyable to use.

1.3 Research approach

The research approach used in this thesis is action research. Action research is a qualitative research approach that is related to case study research, except that the purpose of the research is not only to produce new scientific knowledge but also to influence the subject of the study. In this thesis, the purpose was not only to answer the research question but also to improve the usability practices at the case company during the research.

The research is divided into three stages. In the literature review, presented in Chapter 2, the literature was reviewed for possible answers to the research questions. In the current state analysis, the state of usability at the company was analyzed, in order to select the practices best suited for the company. In the action intervention, a set of practices were tested and analyzed at the company. The research methods used for the current state analysis and action intervention are described in Chapter 3 and the results are presented in Chapter 4. Based on the literature review and the results, a set of recommended practices are presented in Chapter 5 as an answer to the research question. Finally, the results and recommendations are discussed in Chapter 6.

1.4 Notes on terminology

The field of interaction design is riddled with terminology, and sometimes it seems that every author wants to come up with a new term. There are many good alternatives to the term *interaction design* that could have been used almost synonymously for the purposes of this thesis. These include *user experience design*, *user-centered design*, *human-centered design*, *usability engineering* and *usability design*. The reason that this report uses interaction design instead of these alternatives is that it appears to be one of the most widely used terms in both industry and the literature at the moment. The term is discussed in more detail in Chapter 2.3.5.

Design is a problematic word with multiple meanings. In software development it is often used to mean technical design or architecture rather than interaction design. In this thesis, the terms design and designer will be used as abbreviations for interaction design and interaction designer, unless otherwise noted.

This report uses lower-case spelling unless a noun is clearly a proper noun. This means that *agile* is preferred to *Agile*, *product owner* is preferred to *Product Owner*, and *scrum master* is preferred to *ScrumMaster* or *Scrum Master*. However, *Scrum* is capitalized as it is the name of a specific methodology. Finally, *sprint* is used instead of *iteration* or *cycle* when referring to the periods into which agile projects are divided, because it is in common usage and is less likely to be confused with the general idea of an iteration.

Chapter 2

Literature review

2.1 Introduction

The purpose of this chapter is to find answers to the research question in the literature. Because this requires an understanding of agile methods as well as usability and interaction design, these foundational topics are first presented separately, in Section 2.2 and Section 2.3. Only enough detail is included to prepare the reader for the following topics, but the overview also includes some recent findings which will be interesting even for readers familiar with the topics.

After presenting the foundation, the main part of the literature review begins. Section 2.4 explores the question of how interaction design work should be structured in agile teams and organizations. Section 2.5 continues with the question of how best to integrate interaction design into the agile software development process. Finally, Section 2.6 discusses how individual usability activities are best performed in an agile environment.

2.2 Agile software development

2.2.1 Introduction

Agile software development, or agile methods, is an approach to software development where the unpredictable nature of software requirements is embraced. Compared to traditional, waterfall software development, much less emphasis is put on specifying requirements and designing the system up front. The agile process is divided into short sprints that deliver complete software increments, and the software requirements are allowed to change between sprints. This allows organizations that use agile methods

to respond both to a rapidly changing environment and to new knowledge discovered during development.

Although many of the ideas that contribute to agile methods were introduced already during the last decades of the 20th century, agile software development was unified under a single set of values and principles in the agile manifesto in 2001 (Beck et al., 2001). This manifesto, signed by multiple software development veterans, encouraged valuing:

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

Also, the manifesto introduced twelve principles for agile software development. These principles encourage, among other things:

- early, continuous and frequent delivery of valuable software
- a welcoming of emerging requirements, architectures and designs
- frequent collaboration between developers and business people
- empowered and motivated teams
- face-to-face communication
- sustainable development and a constant pace
- technical excellence and a pursuit of simplicity
- continuous retrospection and process improvement

The specific agile process and practices varies depending on the methodology used. The most common methodologies include Scrum (Schwaber and Sutherland, 2011) and Extreme Programming, or XP (Beck and Andres, 2004). Out of these two, Scrum focuses mostly on the project management aspects of agile development, while XP also includes a set of technical practices.

2.2.2 Adoption of agile methods in industry

Agile methods have become a very common approach to software development during the last few years. In a 2008 survey among 642 readers of a software development journal (Ambler, 2008), 69% of respondents indicated that their organization was using agile methods in one or more projects. This was the same adoption rate as the survey found the previous year.

Factor	Improved	No change	Worsened
Productivity	82%	13%	5%
Quality	77%	14%	9%
Stakeholder satisfaction	78%	15%	7%
Cost	37%	40%	23%

Table 2.1: The effect of agile methods on productivity, quality, stakeholder satisfaction and cost (Ambler, 2008).

In 2010, the software company VersionOne conducted a large survey on the usage of agile methods in software development organizations (VersionOne, 2010). Among the 4770 participants from 91 countries, most were project managers or other managerial staff involved in software development. The survey found that 90% of respondents worked in organizations that use at least some agile development practices, and 40% worked in organizations that have been practicing agile for more than two years. Out of these 58% employed Scrum while 17% employed a Scrum/XP hybrid, clearly making Scrum the dominant methodology at this time. The top three reasons for adopting agile methods were accelerated time to market (with 37% reporting that it was of the highest importance), enhanced ability to manage changing priorities (36%) and increased productivity (27%).

2.2.3 The benefits and limitations of agile methods

Evidence on the benefits of agile methods comes primarily from surveys, although there are also some empirical studies. Surveys have generally found very positive results. The study by Ambler (2008) presented in Section 2.2.2 asked participants about the effect of agile methods on productivity, quality, stakeholder satisfaction and cost. The results are shown in Table 2.1.

The VersionOne (2010) survey also asked participants about the perceived benefits of adopting agile methods. The top five factors where participants reported either improvement or significant improvement were ability to manage changing priorities (87%), project visibility (78%), productivity (74%), team morale (71%) and time to market (70%). The only factors where only a minority reported improvement were reduced costs (39%) and the ability to manage distributed teams (34%). No factor had more than 8% of respondents reporting worse or much worse results. Many respondents did, however, report concerns about adopting agile. The top three concerns were loss of management control (36%), lack of upfront planning (33%) and management opposition to change (32%).

Although these results are encouraging for agile methods, there is a risk of selection bias in surveys like this. There is less empirical evidence on the

benefits and limitations of agile methods. A systematic review of all agile studies up to and including 2005 found 36 empirical studies (Dybå and Dingsøyr, 2008). The review found benefits e.g. in customer collaboration, handling of defects, team learning, estimation and the ability to think ahead. There was also a tendency towards higher productivity and code quality in agile methods, and both customers and developers generally were more satisfied with agile methods. Limitations reported were that the agile methods did not work out is some cases, that they worked best with experienced teams and that there was too little attention to design and architecture. These findings should be read cautiously, as the authors reported that the strength of the empirical evidence is very low.

2.2.4 Scrum

As indicated in Section 2.2.2, the most used agile methodology today is Scrum. Scrum was developed by Ken Schwaber and Jeff Sutherland in the early 1990s, and has been described e.g. by Schwaber and Beedle (2001) and Schwaber (2004). The most up to date description of Scrum is in the Scrum guide (Schwaber and Sutherland, 2011).

The guide describes Scrum as a framework for developing and sustaining complex products. This is made possible by a focus on three pillars: a *transparent* process, frequent *inspection* of progress, and *adjustment* of the process when it deviates from the goals. Scrum defines roles, artifacts, events and rules, and each of these has the end goal of providing transparency and opportunities for inspection and adaption.

The roles defined in Scrum are the *product owner*, the *development team* and the *scrum master*. Together, these make up the scrum team. The responsibility of the product owner is to make sure that the product is being developed in the best possible direction. Within the team, the product owner therefore has final authority over prioritizing requirements. The development team is responsible for developing the product. Although each developer may focus on what she is best at (e.g. design, architecture or testing), the development team as a whole is cross-functional and self-organizing. This means that there are no predefined roles or titles, and that the team instead organizes itself to collectively produce the best results. Finally, the scrum master is responsible for ensuring that the Scrum is sufficiently understood and correctly implemented.

The three primary artifacts in Scrum are the *product backlog*, the *sprint backlog* and the *definition of done*. The product backlog is a prioritized list of requirements, and is the responsibility of the product owner. It is constantly updated and improved by the product owner, with the help of the team.

The development team is responsible for estimating the effort required to complete backlog items. The sprint backlog is the set of items selected for development in a given sprint. The definition of done specifies the quality requirements, i.e. what is required from a product increment in order for it be considered done.

The Scrum process is structured into sprints that are two to four weeks in length. In each sprint, an increment of working software is developed. Each sprint has four types of events: a *planning meeting*, a *review*, a *retrospective* and *daily scrums*. In the planning meeting, the product owner presents the goal for the sprint and the most important requirements, and the development team chooses what it can commit to. In the review, the scrum team presents the results of the sprint to stakeholders, and the team and stakeholders together discuss what to do next. In the retrospective, the team considers what went well during the sprint, and what could be improved. Every day, in the daily scrum, each team member describes what he has accomplished since the last meeting, what she intends to accomplish next and whether there are any problem hindering her progress. One way of visualizing the Scrum process is shown in Figure 2.1.

2.2.5 Extreme programming

Before the rise in popularity of Scrum during the past few years, extreme programming (XP) was the most widely used agile methodology (e.g. Shine Technologies, 2003), and as indicated in Section 2.2.2 it is still used but most often in combination with Scrum. Extreme programming was created by Kent Beck and is presented by Beck and Andres (2004).

In comparison to Scrum, XP is more focused on describing a set of values, principles and best practices than on describing a process framework. Most of these practices can be incorporated into Scrum, which is probably why hybrids are common. The core values of XP are *communication* within the team, a strive for *simplicity* in solutions, the right amount of *feedback*, a *courage* to speak truths, discard failures and seek answers and *respect* within the team and for the project.

XP describes roughly a dozen principles and two dozen practices. Of the practices, about half are categorized as primary. These include:

- team practices such as cross-functional teams, a sustainable pace of work, sitting together and clearly showing project information in the workspace
- planning practices such as weekly iterations for development, quarterly iterations for longer term planning, writing requirements as user

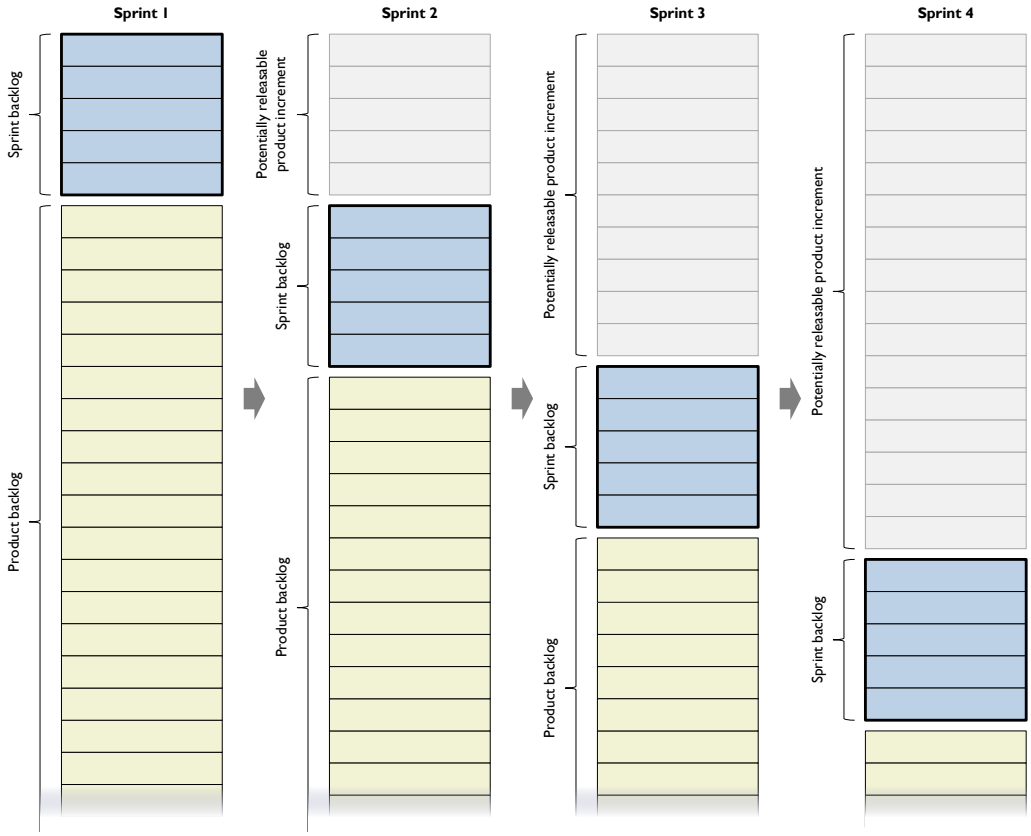

Figure 2.1: In Scrum, items in the product backlog are turned into potentially releasable product increments in two to four week long sprints. Development is done in the order of the backlog items, from top to bottom.

stories and including slack in the schedule
- programming practices, such as programming in pairs, writing tests first, designing the system in small increments and integrating and testing the whole system quickly and frequently

2.2.6 Agile requirements

Gathering and specifying requirements efficiently has long been one of the most important problems in software engineering. At the core of agile methods lies the idea that all requirements cannot be known in advance, and different agile methodologies therefore propose various methods for managing changing requirements.

As described in Section 2.2.4, in Scrum all requirements are listed in the product backlog, which is in essence a prioritized list of work to be done. Pichler (2010) describes the four central qualities of a successful product backlog with the acronym DEEP. First, the backlog is *detailed appropriately*. This means that the items with highest priority are sufficiently detailed to allow development to start, while items further down on the backlog are vaguer. This allows details to be specified at the latest possible moment with the best available information, which minimizes wasted up-front planning.

Second, the backlog items are *estimated*. This allows the product owner to make necessary trade-offs between items. Third, the backlog is *emergent*, which means that it is never set in stone. Rather, items are constantly added, removed, modified and reprioritized. Fourth, the backlog is *prioritized*. The items on top are developed first, and then removed from the backlog

Scrum does not specify any particular format for backlog items. One of the most common formats is *user stories*, which originated in XP, and are described by Cohn (2004). Stories have three important features. First, they are written from the *perspective of the end user*, and only describe features that are valuable to her. This means that purely technical requirements should not be written as stories. Second, they are on purpose lightweight, and *require conversation* between the product owner and the developers to cover all the details. Third, details are written as *acceptance tests*. This makes it easier to determine if a story is done.

Estimating the effort needed to complete requirements has long been a problem in software engineering, and as a result of this many projects have traditionally overrun their schedules, sometimes by wide margins (Moløkken and Jørgensen, 2003). Cohn (2004) describes *story points* as a method of estimating effort for user stories. Story points differ from many traditional methods in that they are *relative*, not absolute. In the beginning

of a project, a story point can be thought of as some absolute unit, e.g. an ideal development day, but further on story points are only compared with the points in previous stories. Story points are estimated collectively by the development team. The amount of story points a team completes per sprint is called the *velocity*, and the velocity of previous sprints can be used to estimate the stories that can be done in future sprints.

2.3 Usability and interaction design

2.3.1 Introduction

People use software to aid them in a wide variety of tasks, and the primary concern of any software development project has to be to provide valuable functionality that allows people to achieve their goals. Just providing the functionality is seldom enough, though. Users and customers will prefer software that is easy to learn and remember, efficient to use and leaves them satisfied at the end of the day. This collection of quality attributes, describing *how well* rather than *whether* users can achieve their goals, is commonly called *usability*.

Authors differ in the specific methods they suggest for creating usable interactive products and services. Most processes share common themes, however, including understanding users through research, creating and evaluating designs with real users, and iteratively refining designs based on the findings of the evaluations. Different authors also use different names to refer to this process. In this thesis the primary term used will be *interaction design* (e.g. Cooper, Reimann, and Cronin, 2007). Other common terms include usability engineering (e.g. Nielsen, 1993), user-centered design (e.g. Norman, 2002), human-centered design (e.g. ISO 9241-210), and user experience design (e.g. Garrett, 2011).

2.3.2 Usability

To understand how to improve usability, we must first understand what usability is. The term is defined in an ISO standard (ISO 9241-11) as:

> "The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use."

While this is precise, and is the definition used in this thesis, a more everyday definition may give a better understanding of the concept. According to Krug (2006),

> "usability really just means making sure that something works well: that a person of average (or even below average) ability and experience can use the thing—whether it's a Web site, a fighter jet, or a revolving door—for its intended purpose without getting hopelessly frustrated."

To better understand usability, it is useful to think of it as made up of different components. The ISO standard includes three components: effectiveness, efficiency and satisfaction. Nielsen (1993) further divides effectiveness into three subcomponents when he, in his popular definition, defines usability as:

- *Learnability*, i.e. how quickly a user can learn to use a product or part of a product the first time she encounters it. This component is most important for novice users.
- *Memorability*, i.e. how well a user can relearn a product after a period of not using it. This component is important for causal users who use the product intermittently.
- *Few and noncatastrophic errors*, i.e. how few errors a user makes while using the product, and how easy these errors are to discover and recover from.
- *Efficiency*, i.e. how quickly a user can use the product once she has reached a sufficient level of expertise. This component is most useful for expert users.
- *Satisfaction*, i.e. how pleasant the use of the product is for a user.

2.3.3 User experience

A related concept to usability is user experience (UX). Although it is sometimes used synonymously with usability, more often it is used to encompass the entire experience a user has with a product, including qualities that go beyond the components of usability. Jordan (2002) presents the user experience as a consumer hierarchy of needs with functionality at the base, usability in the middle and pleasure at the top. This implies that high usability is necessary but not sufficient for a good user experience. Beauregard and Corriveau (2007) describe user experience as the emotions, thoughts and attitudes that arise when a user interacts with a product. Although this thesis focuses on the more well-defined concept of usability rather than on user experience, it is important to keep in mind that the two concepts are related, and that most of the methods that contribute to usability therefore also contribute to user experience.

2.3.4 The benefits of usability

The cost of usability consists of the increased design and development resources needed for interaction design practices. The benefits tend to be harder to quantify, but consist both of increased value to customers because of higher user productivity and satisfaction, of decreased cost to the company due to lesser need for training and support, and in some cases of lower development costs when interaction design helps avoid the development of unnecessary or unwanted functionality.

While the costs are relatively easy to quantify, the benefits are less so, because they depend to a large degree on the type of software developed (e.g. e-commerce site, application or intranet) and the business model employed the company. Nielsen (2003) collected data from 863 website design projects and concluded that a best practice is to devote about 10% of a projects budget to usability. He found that this investment on average increased sales and conversion rates by 100%, traffic and visitor counts by 150%, user performance and productivity by 161% and use of target features by 202%. This averages to an increase in usability of 135%. More recently, Nielsen (2008b) conducted a new survey, in which he found that usability budgets still were around 10% in companies that employed usability practices, and that this investment now led to an average increase in usability of 83%, lower than five years earlier but still high.

2.3.5 Interaction design

The ideal process for creating usable products is by no means settled, and it includes a certain amount of creativity that is hard to specify. Over the years, however, several practices have proven successful. In a classic article, Gould and Lewis (1985) recommend three central principles for achieving usability:

- *Early focus on users and tasks*. The designer should understand who the intended users of the design are, how they behave, and what kind of work they perform.
- *Empirical measurements*. Simulations and prototypes should be used early in the process to observe actual use of the design by the intended users.
- *Iterative design*. Usability problems should be fixed, and this requires an iterative process of design, evaluation and redesign.

Beyer and Holtzblatt (1998) describe a process they call *contextual design*. The process consists of gathering data about users with *contextual inquiry*,

a method where the designers observe the work of users and interview them in their natural work environment. The gathered data is modeled using *work models*, which describe work processes, artifacts, environments and the flow of information. Based on these models, the design is created, prototyped and evaluated, and this process is iterated until the design is done.

Cooper (2004) and Cooper, Reimann, and Cronin (2007) describe an interaction design process called *goal-directed design*. The focus of goal-directed design is to understand and fulfill the *goals* of users, and like in contextual design, this is done by observing and interviewing users in their natural environment. The main difference is that goal-directed design models the users using *personas*, which are personifications of groups of users. Personas are concrete and have fictional names and biographies, but their goals, needs and behaviors are based on the gathered data. The benefit of using personas is that they make the users more real in the minds of designers and developers, which allows them to focus better on actual rather than stereotypical users.

In addition to user research and modeling, Cooper, Reimann, and Cronin (2007) describes how design is done as part of the interaction design process. This process is also elaborated by Goodwin (2009). First, scenarios are written. These are narratives that describe personas interacting with the product in typical ways. Based on the scenarios, the designers sketch an interaction framework, which defines the main functional and data elements and their relationships. Further, a visual design framework is created, the design is refined and evaluated, and the process is iterated until the result is acceptable.

Nielsen (2003) describes a process that is more focused on evaluation than on research and design. The two main methods for evaluation are *expert reviews*, where the usability professional performs typical tasks with a design while looking for problems that break common usability guidelines or heuristics, and *usability testing*, where representative users perform typical tasks with a design while thinking aloud about what they do.

2.3.6 Similarities and differences to agile methods

There are some similarities between interaction design and agile methods, because both movements have recognized some of the same problems in traditional software development. One similarity is the focus on evaluation and iteration. However, agile methods emphasize iterating the working software after evaluating its functionality with customers, while interaction design emphasizes iterating the design after evaluating its usability with

end users. Also, in many interaction design processes (e.g. Goodwin, 2009) the designers complete most or all of the design before development begins, and document it in elaborate design specifications. This is clearly not compatible with agile principles.

Another similarity is the focus on users, which is primary in interaction design but also important in agile methods. Agile practices such as user stories emphasize that requirements should be specified from the viewpoint of the user, and their concrete value to the user should be clarified. However, as Nielsen (2008a) points out, interaction design and usability are still secondary, and are often left to happen as a side product of the coding.

Two case studies illustrate the frictions between interaction design and agile methods. In a controlled XP project using most of the XP practices, Jokela and Abrahamsson (2004) found that the responsibility for usability was transferred to the on-site customer (product owner), and that almost no usability engineering practices were employed by the developers. Federoff and Courage (2009) describe the adoption of agile methods by 30 development teams at Salesforce.com. Initially, six months after the adoption, only 24% of the user experience team members were satisfied with the change. They expressed several concerns, such as difficulties with identifying target users and their needs in a just-in-time process, with achieving a holistic design in incremental development, and with dividing the user experience team effort between the different development teams.

As shown in sections Section 2.2.3 and Section 2.3.4, agility and a focus on usability are both important for most software development projects to be successful. This section has shown that integrating the two approaches is not straightforward, and has discussed some of the main conflicts between interaction design and agile software development. The following sections will set out to find solutions to them.

2.4 Interaction designers on agile teams

2.4.1 Introduction

One of the core practices in agile software development is that there is a single person on the team, the product owner (or internal customer in XP) that either in fact is a customer of the product, or more commonly is someone who understands and represents the actual customer. The product owner has the authority to prioritize requirements, which ensures that the team keeps focused on what is most valuable for the customer, rather than on what seems important to the developers, who seldom have

the possibility to understand customer needs equally well.

While this team structure is not bad from the point of view of achieving usability, Beyer, Holtzblatt, and Baker (2004) describe some of the reasons why it often is insufficient. First, the product owner may have a hard time representing users when the users of a product are not the same as the customers, which is the case in most enterprise software. Second, as the product owner works intensely with the development team and the product, he will become very familiar with how the product works and also to some degree how it is implemented, and this will make it harder for the product owner to approach the product from the point of view of the user, and to give sufficient priority to usability requirements. Third, even in the case when the product owner is an actual user of the product, there is still the fundamental problem that people seldom can express what they do and what they need. Listening to users rather than observing them is not sufficient for high usability.

2.4.2 The interaction designer

Beyer, Holtzblatt, and Baker (2004) suggest that, in order to ensure a usable product, design should be separated from development. This is also one of the fundamental conclusions of Cooper (2004), who states that developers, even when they do their best to keep the usability of the product in mind, are too knowledgeable of the implementation of the product to view it as the user does. Also, there is a conflict of interest between ease of use and other concerns that are important to developers, such as ease of development and maintenance. The solution is to introduce the role of the interaction designer, whose primary responsibility is to ensure the usability of the product.

Interaction design is a specific activity that has to be performed to create high-usability products, and it will be performed implicitly by the developers if not explicitly by a designer. However, this does not necessarily mean that one or more people on the team have to have the role of designer and play no part in development. Fox, Sillito, and Maurer (2008) identify three different approaches to design: the specialist, the generalist and the specialist generalist. In the specialist approach, the designers are concerned exclusively with interaction design. This is the designer role used in traditional software engineering. In the generalist approach, there is no specific designer, but the developers do the interaction design. Finally, in the specialist generalist approach there are specific designers, but these are also involved in development. This is similar to other roles in agile teams, where developers may specialize in a certain area (e.g. architecture

or testing) while still contributing to all aspects of development.

2.4.3 Organization structure and collaboration

Another question is how to manage multiple interaction designers within an organization. According to Nielsen (2009a) there are two main types of structures: the centralized structure and the distributed structure. In the centralized structure the designers form a single functional team within the organization. This allows better specialization within the team, more flexible resource allocation and better support for cross-cutting initiatives. In the distributed structure, designers are assigned to individual projects. This allows them to collaborate more closely with the developers. According to Nielsen, the distributed structure is to be preferred in an agile organization, because of the importance of designers and developers sitting together and communicating actively. However, some organization-wide activities can still be performed occasionally using a matrix structure.

Other studies tend to support this conclusion. In an observational study of a large agile team where the developers and designers were located separately, Ferreira, Sharp, and Robinson (2011) found significant problems in communication, with developers not getting enough feedback from designers, and both groups being defensive about their work. Cohn (2010) also stresses the importance of designers working closely together with developers and considering themselves part of the agile team.

However, there is not always enough design resources to have one or more dedicated designer per team. At Salesforce.com, Federoff and Courage (2009) describe how designers initially worked with as many as four development teams at once. As this proved problematic, the teams were restructured so that designers worked on at most two teams at a time, but this meant some development teams no longer had designers on them. Instead, designers allocated two hours each weeks during which these teams could consult designers. According to Federoff and Courage, this was a successful compromise.

2.4.4 Cross-functionality

A central idea in agile development is that of cross-functional teams, where members may specialize but also perform tasks beyond their specialty (Cohn, 2010), and where all members work for the common good of the team. In Scrum, for example, the only two recognized team roles are the product owner and the scrum master; all other team members are considered developers. Even though the role of interaction designer may be

different enough to warrant a special role, as in the specialist approach described above, steps can still be taken to increase the developers' understanding of design and the designers' understanding of development.

Ungar and White (2008) present the design studio as one method to increase this knowledge transfer. In the design studio, designers, developers and possibly other stakeholders come together for a day to iteratively sketch, present and critique solutions to particular design problems that have been selected in advance. In their case study the authors found multiple benefits in this method, including role sharing and knowledge transfer, better team cohesiveness, a better shared understanding of the design vision, and an opportunity to educate team members about interaction design. Federoff and Courage (2009) also report success with the design studio.

2.5 Interaction design in the agile process

2.5.1 Introduction

One of the most important sources of conflict between agile methods and interaction design is the question of how to integrate the two into one process, and especially the question of how much (if any) up-front design should be done. While many interaction designers traditionally have been of the view that the user should be understood and a solution designed before development begins, agile proponents have strongly avoided any kind of extensive up-front design or analysis work (so called big design up front). This conflict is exemplified in a 2002 debate between Kent Beck and Alan Cooper (Nelson, 2002), two of the main proponents of agile development and interaction design, respectively. In the debate, Cooper holds fast to the idea that much of the iterative development in agile methods is unnecessary if proper interaction design is done up-front, while Kent opposes this view, stating that while he agrees with the techniques, he disagrees with the process.

That debate was ten years ago, and since then solutions to the conflict have emerged in the literature. Most of these revolve around two ideas: parallel tracks, where design is being done one or more sprints ahead of development, and little design up front, where some preliminary envisioning, design and user research is done before development begins. Five years after the debate, Cooper, Reimann, and Cronin (2007) write that they no longer believe that all design work should precede development. Instead, "all aspects of a product should be designed before they are built." In

essence, most of the design work can be moved into the iterative process, as close before the development as possible, to improve agility while retaining design.

2.5.2 Parallel design and development tracks

The most widely recommended method for successfully integrating interaction design with agile methods is the parallel tracks approach. First described by Miller (2005) and Sy (2007), both working at Autodesk, the method prescribes dividing the process into parallel tracks, with developers working in one track and interaction designers in the other. During a sprint, the interaction designers

- Conduct usability tests with the product of the previous sprint
- Work on the design for the next sprint
- Gather customer data for the sprint following the next one

The exception is the first sprint, where initial planning and user research is done, but no development. The interaction designers do not work separately from the developers, however, but participate in the daily scrum meetings and interact with the developers frequently. This allows the developers to better understand and give feedback on how the design develops, and allows the interaction designers to help developers implement the design. The parallel tracks approach is illustrated in Figure 2.2.

Sy (2007) lists several benefits with this method compared to traditional user-centered design. First, usability investigations are conducted early enough so that their results can be taken into account in the same release. Second, the most important parts of the product are designed first, and there is less risk of putting effort into designs that will become obsolete. Third, more of the product is actually designed, and the implementation of the design is more true to the design intent and overall better.

Budwig, Jeong, and Kelkar (2009) describe how they, when introducing an agile usability process at PayPal, first tried doing design and development in the same sprint, but found this to be unsuccessful as it did not give enough time for the design. They adjusted the process so that designers work one or two sprints ahead of development, and found this to work well. Federoff and Courage (2009) also describe how they successfully implemented the parallel tracks approach at Salesforce.com, because they found that there was not enough time for design, implementation and usability testing in one sprint. In addition, in special cases where the product is new and there is much uncertainty in the design, Federoff and Courage

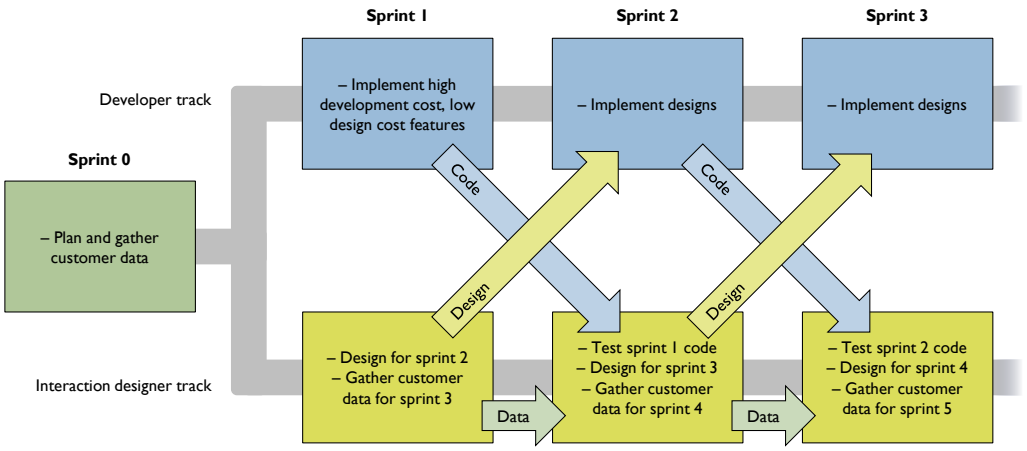

Figure 2.2: Interaction design and development can be done in parallel tracks, with design and customer research being done before the development sprint and usability testing being done after the development sprint (adapted from Sy, 2007).

recommend doing the design an entire release instead of just a single sprint ahead.

The parallel tracks approach is recommended by Cohn (2010), who states that he has seen this approach adopted in dozens of projects. Cohn emphasizes that, even while working in separate tracks, it is essential that interaction designers and developers consider themselves part of the same team. The top priority for interaction designers is therefore, according to him, to make sure that the team delivers what it has committed to for the sprint.

Nielsen (2008a) also recommends using parallel tracks, based on a survey of 105 designers and developers and 12 case studies. He states that quick usability testing is compatible with the approach and can be done even weekly. According to Nielsen, ideally foundational user research should be done before development begins, and may even be done outside the scope of any single project.

2.5.3 Little design up front

The authors cited in the previous section all state that, while much design can be moved to the sprint prior to development, some up-front design before development is required to maintain a consistent vision of the product. In fact, in a systematic review of agile usability studies, Silva et al. (2011)

found that little design up front was the most commonly used practice. While any up-front design may seem problematic from the point of view agile development, it is worthwhile to keep in mind that the design referred to in the agile literature is mostly technical and not interaction design, and that some agile authors (e.g. Pichler, 2010) put emphasis on crafting a good product vision before development begins, in order to keep the project focused and the product cohesive.

Sy (2007) presents the idea of a sprint zero, to be held at the beginning of each release. This sprint is of the same length as the other sprints, i.e. "weeks rather than months," and includes the activities of gathering data to refine the vision of the product and the goals of the release, conducting interviews or contextual inquiries with actual or potential users, and modeling the users and their workflows with models such as lightweight personas or scenarios. These activities vary somewhat depending on whether the release is the start of a new project or an update to an existing product.

Other authors also recommend little design up front. Ferreira, Noble, and Biddle (2007b) found that up-front design is common in agile projects, and concluded that interaction design should be done before development when this reduces risk. Nielsen (2008a) states that the other main recommendation on agile usability, in addition to doing design one sprint ahead, is to create a coherent vision during an initial sprint and maintain it during annual or semi-annual design vision sprints. Budwig, Jeong, and Kelkar (2009) also recommend incorporating quarterly design vision sprints in the normal cycle.

A question raised by these vision sprints is what developers should be doing when the product owner and designers are occupied with creating the vision and doing user research. To an extent it is important that the developers are involved in this work, especially in creating the vision to make sure that it is shared within the entire team (Pichler, 2010). Beyer, Holtzblatt, and Baker (2004) also state that developers may participate in some of the user research activities. Finally, developers can start setting up or improving the development environment (especially in the beginning of a new project), refactoring the code to improve its quality (especially later in a project), or, if possible, implement features that have minimal user interfaces but require much development effort.

2.6 Interaction design activities in agile projects

2.6.1 Introduction

Interaction design can be thought of as involving three main types of activities: user research, design, and usability evaluation. While in any process these activities are to some degree intertwined, e.g. usability evaluations can be used both during design and as a research method, it is still useful to think of these as separate activities. The previous section explored how these activities should be integrated into the agile development process. This section continues by exploring in more detail how these activities should be performed in a manner suited for agile methods.

2.6.2 User research and modeling

To be able to design products that are usable, the interaction designer must first understand the user: her goals, her needs, her skills and her work. While some understanding of users can be gained by observing them in usability tests (see Section 2.6.4), other techniques are more useful when the goal is to design new functionality rather than to improve existing one. The methods for understanding users are known as user research, and the methods for communicating this understanding to the team and other stakeholders are called user modeling.

Two of the most common user research techniques are contextual inquiry and interviews. Contextual inquiry was first introduced by Beyer and Holtzblatt (1998), and involves observing the user acting in her natural environment, i.e. in context, while at the same time interviewing her and asking clarifying questions about what she does. Contextual inquiry is a useful technique when the goal is to understand how users actually behave, rather than how they think they behave. This is because people are often incapable of reporting their work and their behavior, and sometimes also their needs, as these have become unconscious. Ordinary interviews, on the other hand, require less time and are better suited when the goal is to understand the conscious opinions of users rather than their behavior.

Beyer, Holtzblatt, and Baker (2004) describe one approach to user research, adapted to agile development. First, when the project vision is created, the one or two user roles that the product focuses on are identified. Then, contextual inquiries are performed with at least three users representative of each role. The issues found are modeled with affinity diagrams, and the current work processes of users are modeled with sequence models. Multiple team members, including the product owner and developers, can

participate in the contextual inquiries. According to the authors, one week of research can be sufficient for smaller releases, while large projects or those introducing disruptive technologies require more time.

While Cooper, Reimann, and Cronin (2007) agree with primarily using contextual inquiry to understand users, they suggest modeling users with so called personas. A persona is an archetypal user that represents a particular role. In contrast to a description of a role, the persona has a fictional name, photo, and some basic demographic information. The purpose of this is to make the users more concrete in the minds of developers and designers, and easier to empathize with. Although personas contain some fictional details, the actual goals, needs, behavior and work described in personas are based on the user research. Cohn (2004) discusses replacing user roles with personas in user stories. He recommends creating personas only for the one or two most critical roles, and cautions against writing personas unless sufficient user research has been done to base the persona descriptions on data.

2.6.3 Design and prototyping

While the actual methods of coming up with design ideas (e.g. sketching) are not that relevant from the point of view of the agile development process, the artifacts that are produced to communicate the design to the product owner, developers and other stakeholders are. Interaction design in traditional software development produces artifacts such as detailed design specifications. Because documents are deemphasized in agile development, most authors recommend replacing these artifacts with lightweight artifacts, such as low-fidelity paper prototypes, and spending more time in face-to-face discussions. In fact, in their systematic review of agile usability studies, Silva et al. (2011) found that using low-fidelity prototypes was the third most common practice (after little design up front and close collaboration).

While there are many lightweight artifacts for communicating design, and different artifacts are suited for different purposes, prototypes have the benefit of both communicating the design and allowing exploratory usability testing (see Section 2.6.4). Snyder (2003) describes four dimensions of prototype fidelity: breadth, i.e. how much functionality is covered, depth, i.e. how much the functionality has been fleshed out, look, i.e. how much the prototype looks like a finished product, and interaction, i.e. the extent to which the prototypes allows realistic input and output methods. Paper prototypes are prototypes that are hand-drawn or drawn on the computer and printed out. They are generally narrow but deep, i.e. they allow only specific functionality but they cover the entire path of that functionality.

Both looks and interaction are low-fidelity. Compared to this, working prototypes (often called high-fidelity prototypes) run on the computer, and therefore provide higher-fidelity interaction and looks.

Several studies indicate that low-fidelity prototypes are sufficient for finding usability problems early in the design. Hall (2001) summarizes earlier studies and concludes that "it is possible to achieve good design information from user testing of low-fidelity prototypes early in the design process". Walker, Takayama, and Landay (2002) found that low- and high fidelity prototypes are equally good at uncovering usability issues and Sefelin, Tscheligi, and Giller (2003) also found that both types produce almost the same quantity and quality of findings.

In addition to allowing usability testing before coding starts, prototypes can also be used to communicate the design to developers and other stakeholders. Sy (2007) describes how, when moving to agile, they replaced design specification with prototypes (both low- and high-fidelity). These prototypes are presented in person to developers, and typical workflows are demonstrated. Federoff and Courage (2009) also explain that they replaced the need for specifications with working prototypes created by the designers. Chamberlain, Sharp, and Maiden (2006) concluded, based on a field study of three projects, that one of the five most important principles in agile usability is the willingness of designers to provide developers with prototypes.

2.6.4 Usability evaluation

One of the central ideas of interaction design is that actual or representative end users should be involved in the development process to evaluate the usability of designs or of the working product. The most important method for this is the usability test, in which a user performs typical tasks with a prototype or a product while thinking aloud. Rubin and Chisnell (2008) identify three main types of usability tests suitable to fast-paced development: exploratory tests, assessment tests and validation tests. Exploratory (or formative) tests examine how well initial design concepts work, and are usually done with low-fidelity prototypes. Assessment (or summative) tests examine how well the design has been implemented in working code. Finally, validation (or verification) tests measure the usability to examine whether a product fulfills certain usability standards.

Sy (2007) describes how, in the parallel tracks approach, exploratory tests can be done rapidly with prototypes during the sprint before implementation, to improve the design before coding starts. Assessment tests can be performed in the sprint after coding, to evaluate whether the de-

sign has been implemented successfully. Because this requires many test participants, Sy explains how they use internal users who share some characteristics with the end users for some tests, and reserve external users for those tests where they are absolutely necessary. In conclusion, Sy found that, while exploratory tests can be done iteratively without an agile process, assessment tests benefit greatly from agile development, as they can be done earlier, and the results can be taken into account in the same release.

Beyer, Holtzblatt, and Baker (2004) also suggests both exploratory testing of the design in the sprint before development, and assessment testing of the working product after it has been coded. The authors stress that exploratory testing with paper prototypes is more important than assessment testing, because it aids in making sure the design is sound before development begins, which minimizes the risk for expensive rework.

To be able to conduct exploratory usability tests rapidly within a sprint, simplified usability techniques are necessary. Nielsen (2009b) describes the essentials of discount usability engineering, which he first introduced in 1989. In discount usability engineering, low-fidelity prototypes with narrow functionality, usually paper prototypes, are created and tested with three to five users in qualitative tests, where the focus is on finding problems rather than gathering quantitative data. In addition, inspection methods that do not involve end-users, such as heuristic evaluation, may be performed. Nielsen (2011) recommends two iterations of testing, but states that one iteration is better than none. Using discount techniques, he states that tests can be conducted as often as weekly, so as long as there are enough test participants, it is possible to conduct multiple rounds of exploratory testing within a four-week sprint.

Medlock et al. (2002) describe an even faster method of exploratory testing, called rapid iterative testing and evaluation (RITE). In RITE, tests are scheduled with couple hour breaks between them, and usability problems are fixed as soon as they are discovered and a quick solution is found, during the time before the next test begins. The prototype is therefore iterated between the tests, so finding the same usability problem twice is not as common. Also, the entire team participates in the test sessions, and in identifying problems and coming up with solutions. Federoff and Courage (2009) discuss how RITE has become almost the only testing method used as Salesforce.com, because it is quick and involves the entire team in engaging problem solving and not just passive observation.

Chapter 3

Methods

3.1 Introduction

This chapter describes the research methods used in the thesis. The goal of the research was to find out what practices to introduce into the agile software development process at the company in order to improve the usability of its products, and the research approach selected for this end was action research. This chapter therefore starts by describing action research in Section 3.2. Next, in Section 3.3, the case is presented, including descriptions of the case company, the case project and the project team. After this, in Section 3.4, an outline of the research process follows. In Section 3.5 the procedures for data collection and analysis are presented, and in Section 3.6 ethical considerations and confidentiality are discussed. Finally, in Section 3.7, the usability practices that were tested during the research are presented.

3.2 Action research

Action research is described by Avison et al. (1999) as "an iterative process involving researchers and practitioners acting together on a particular cycle of activities, including problem diagnosis, action intervention, and reflective learning." It is closely related to case study research (e.g. Yin, 2009), with the difference that the researcher actively participates in the studied activities. As a method, it is therefore well suited for a research project where a major goal is to improve the practices at a company, in addition to creating scientific knowledge.

Runeson and Höst (2009) describe four research methodologies applicable to software engineering research: survey, case study, experiment, and

action research. Like the case study, and in contrast to the survey and the experiment, action research is primarily concerned with qualitative (and not quantitative) data, and uses a flexible (and not fixed) research process which can be changed during the course of the study as new findings emerge. This is because both methods are interested in understanding phenomena in context. They sacrifice the ability to explain causal relationships or allow generalization from a sample. Instead, they provide a deeper understanding of the phenomena under study. In software engineering, where the context often is complex and varies widely between cases, these methods are well suited.

Much of the research design for the thesis is based on the guidelines presented by Runeson and Höst (2009) on case study research in software engineering, because no equally useful set of guidelines exists for action research. Except for the action intervention, case study research is very similar to action research. Runeson and Höst also state that "[for] the research part of action research, these guidelines apply as well."

3.3 Case description

3.3.1 The case company

Founded in 1991, Rapal Oy is a privately owned Finnish company specializing in producing information to aid the decision making of owners and users of premises as well as constructors of infrastructure. This is achieved by providing software as a service, complemented with consulting. Rapal Oy has a staff of roughly 50 employees, and in 2010, the net sales amounted to approximately €4.6 million. (Rapal Oy, 2012)

The three products provided by Rapal Oy are Optimaze.net, Fore and Forecast. Of these, Optimaze.net is the largest by sales. It includes web applications for users of premises and owners of real estate. For building users the product gives information about how their spaces are utilized, how much they cost, and what their environmental impact is. For owners the product allows the management of property holdings, including contracts and invoicing.

3.3.2 The case project

While the current state analysis concerned the entire case company, the action intervention was performed within a single software development project at the company. The goal of the project was to create a new product,

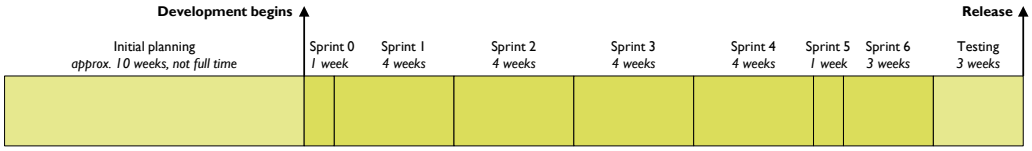

Figure 3.1: The case project schedule.

Optimaze.net Scenario, which would be related to but separate from the main Optimaze.net product. Optimaze.net Scenario would be a product for strategic facility planning that would allow customer organizations to get an overview of the buildings they use, and to plan changes to their building portfolios in order to improve utilization and reduce costs. In contrast to the main product, Scenario was meant to work as a lightweight tool that does not require an extensive deployment process, and therefore can be taken into use by new customers easily. A secondary goal of the project was to serve as the context for this research and thereby allow the team to try out new usability practices that could later be employed in other projects.

Initial visioning and planning of the product started in the spring of 2011, and initial design started in the summer. Development began in September, and the product was released in March 2012. The development of the product followed the Scrum methodology and was divided into nine sprints. These sprints included an initial one-week warm-up, four four-week development sprints, one- and three-week development sprints (the fifth four-week sprint was split into two due to special circumstances), and a three week testing sprint. The project schedule is shown in Figure 3.1.

3.3.3 The project team

The project team consisted of a product owner, a scrum master, two developers and an interaction designer. The product owner had been responsible for the entire Optimaze.net product for some time. There was one other project under way at the same time, and the product owner therefore worked only about half-time on the case project. The scrum master was a former software developer at the company, but worked exclusively as scrum master and did not do any development during the case project. While he had some other responsibilities, he worked mostly full-time on the case project.

The two developers were both quite new at the company. While one of them had worked on agile projects before, neither had extensive experience with agile software development prior to starting at the company. Neither

also had much experience with interaction design prior to the case project. Both developers worked full-time at the project. Finally, the interaction designer was the author of this thesis. He had previously worked as a front-end developer at the company. While the amount of time dedicated to research versus work (including the action intervention) varied somewhat during the project, on average the interaction designer worked about half-time on the project.

3.4 Research process

The research consists of three stages: the literature review, the current state analysis and the action intervention. During the first stage, the goal was to review the literature for information on how best to integrate interaction design with agile software development. During the second stage, the goal was to analyze the current state of usability at the case company and to examine its particular problems and needs. During the third stage, the goal was to test usability practices in the case project and to analyze how well they worked. The usability practices tested in the action intervention are described in Section 3.7.

The three stages did not occur serially; rather they overlapped each other somewhat. For example, the literature review was not fully completed when the current state analysis began, and parts of the action intervention were started before the current state analysis was completed. The reasons for this were mostly practical. Because some of the usability practices required several months to test in practice, and many required the involvement of other team members, company employees and end users, it was not possible to completely isolate them to a specific part of the project. This arrangement also had the benefit that the research process could be adjusted slightly as new information was gained.

3.5 Data collection and analysis

3.5.1 Interviews

The primary method of collecting data was semi-structured interviews with company employees. During the current state analysis stage, nine interviews were conducted with employees representing all the business units in the company. During the action intervention stage, five interviews were conducted with the project team members as well as the head of

development. Before each interview, questions were planned and written down, but the order of the questions was in some cases varied, and some questions were improvised during the interviews in order to clarify important subjects. The interviews varied from about 25 minutes to about 60 minutes in length. All interviews were recoded and transcribed. The interview questions for the two interviews are available in Appendix A and Appendix B.

The current state analysis and action intervention interviews were analyzed in roughly the same way. After transcription, the interviews were color coded by participant and divided into statements, where each statement included a single message. The statements were then grouped by topic. The topics mostly followed the predefined topics planned before the interviews, but especially in the current state analysis interview the topics were adjusted based on the interview data. Next, each topic was analyzed for important messages. Messages were extracted, and each message was annotated with the interview participants that had stated it in the interview. When applicable, the messages were also grouped into negative and positive messages.

3.5.2 Observation

In addition to the interviews, the researcher performed observation during the action intervention, according to an observation framework, and wrote down notes when interesting phenomena related to the usability practices were observed. Also, the sprint retrospectives arranged at the end of each sprint, where team members discussed what worked well and what did not during the sprint, as well as suggest improvements, were well suited for observing and discussing how the team members had reacted to the usability practices. At the end of the action intervention stage, the researcher wrote down additional observations notes, based on the observation framework. The observation framework is available in Appendix C.

The observation notes were analyzed by first combining all notes from different sources, and combining similar notes. Next, the notes were grouped according to the topics in the observation framework. The notes were further grouped into positive and negative observations, where applicable. Finally, the most important observations in each group were extracted based on the subjective view of the researcher.

3.5.3 Triangulation

Triangulation was used to improve the precision of the research. Data triangulation was used in the interviews, as many people with different roles were interviewed. The interview participants were consciously selected to increase the diversity of viewpoints. In addition, methodological triangulation was used to some extent, as both interviews and observation were employed to collect data. Observer triangulation was not used, as only one researcher collected the data and analyzed the results.

3.6 Ethical considerations and confidentiality

3.6.1 Informed consent

All participants in the interviews were asked verbally for their informed consent. Participants were informed about the purpose and possible benefits of the study, their roles in it, that participation is voluntary, that their participation is anonymous, that there is a risk that they can be identified based on their work descriptions or the answers they give to interview questions, and that they can contact the researcher with any questions they have during or after their participation. All participants agreed to participate on these terms.

3.6.2 Confidentiality

The researcher had signed a confidentiality agreement when employed at the company, restricting the publication of some of the research results. However, there was a separate verbal agreement with the head of development that none of the topics covered by the research were to be considered confidential, as long as no customer or other sensitive information was included.

3.7 Tested usability practices

In addition to the new interaction designer role (presented in Section 3.3.3), two new usability practices were tested during the case project: paper prototyping one sprint ahead, and usability testing of previous sprint results. The practice of high-fidelity prototyping, which had been used in earlier projects at the company, was also employed. All usability practices were arranged by the interaction designer, although other team members

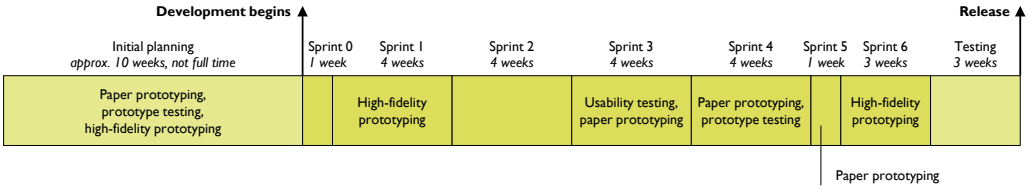

Figure 3.2: Usability practices tested during the case project.

participated in some of them. The usability practices tested during the project are shown in Figure 3.2.

3.7.1 Paper prototyping

Paper prototyping was used in the initial planning stage, as well as in sprints 3 and 4. During the planning stage, a paper prototype was created covering roughly the functionality planned for sprints 1 and 2. This prototype was tested in two rounds, first in June with four users and then in July with three users. The tests were standard usability tests in which the participants were asked to perform representative tasks using the prototype while vocalizing their thoughts. Each test lasted for about one hour, and ended with general discussion about the prototype. The second round of testing was recorded on video. Both rounds of testing were analyzed afterwards, and the prototype was improved based on the findings.

No paper prototyping was done during sprints 1 and 2. During sprint 3, a paper prototype covering the functionality planned for sprint 4 was created, but the prototype was not tested with users. During sprint 4, a paper prototype covering the functionality of sprints 5 and 6 was created and tested with three users, using the same test procedure as the second round of testing in July. Finally, in sprint 5, additional paper prototyping was done for sprint 6 functionality.

The paper prototypes were primarily created by the interaction designer, although the scrum master and product owner participated extensively in sprints 3 to 5. While the scrum master and product owner participated in the first round of testing, the second and third rounds were done only by the interaction designer, who summarized the results for the rest of the team. In addition to testing with end users, the prototypes were occasionally used as the basis for discussions with internal domain experts.

The paper prototypes were used during backlog grooming sessions as well during sprint planning, to help the developers understand and estimate the user stories. In addition, they were available during the

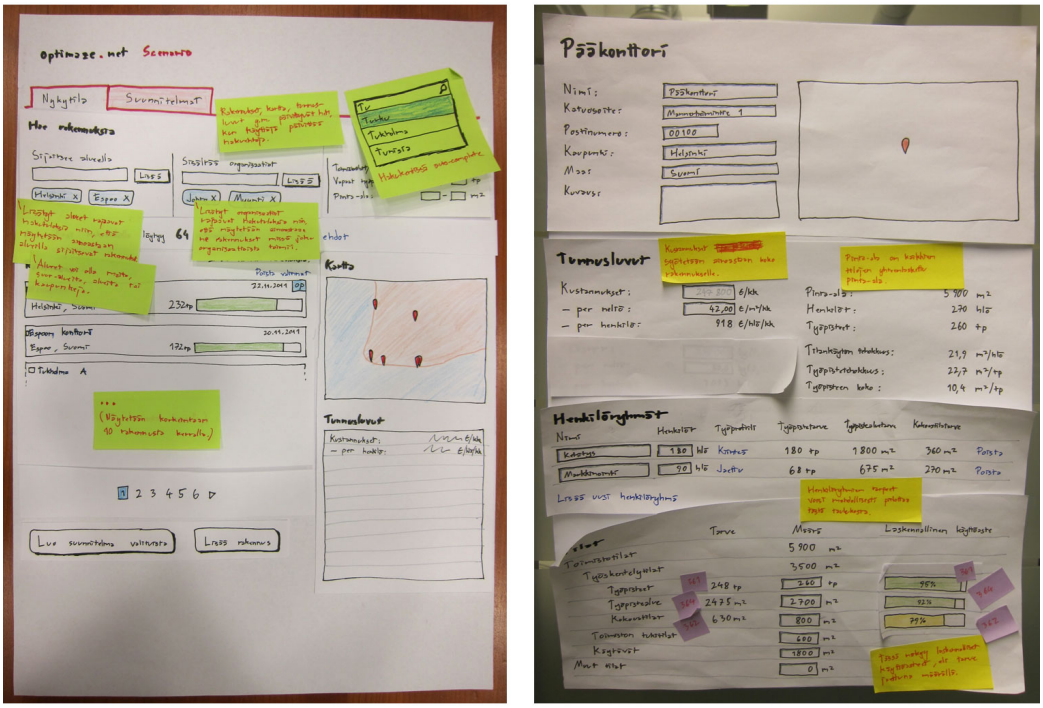

Figure 3.3: Two examples of the paper prototype. The prototype on the left is from sprint 3 and the prototype on the right is from sprint 5.

sprint, first in a folder and then, from sprint 4 onwards, on a wall in the working area. When used for development (but not during the tests), the prototypes were annotated with notes explaining the interaction of the prototype. During the last sprint, they were also annotated with small notes explaining which user story a particular part of the prototype belonged to. Two examples of the prototype from different parts of the project are shown in Figure 3.3.

3.7.2 Usability testing

Usability testing of a working product increment was done once during the project, in sprint 3. The tests covered the functionality developed during the first two sprints. Two users participated in the tests, which mostly followed the same procedure as the paper prototype tests, with the users given representative tasks and asked to perform them while thinking aloud. The users' actions were captured using screen recording software, and notes were also taken. After the tests, the results were analyzed by the interaction designer. Only the designer was present during the tests.

One of the two tests was also analyzed in common by the entire team. The team first watched the recording of the test, and each member then individually listed the most important usability problems. The problems where combined and prioritized by the team, and were then either directly taken into account in the design for the next sprint, or added as items to the product backlog for the product owner to prioritize.

3.7.3 High-fidelity prototyping

High-fidelity prototyping using static HTML pages was done before development started as well as in sprints 1 and 6. During the planning and sprint 1, the high-fidelity prototype was used primarily to create the initial visual design for the product. During the last sprint, it was used to finalize the visual design for the product. The high-fidelity prototypes were only used as aid for the developers. They were neither tested with end users nor discussed outside the team.

Chapter 4

Results

4.1 Introduction

This chapter presents the results of the research. It is dived into two parts. The first part, in Section 4.2, presents an analysis of the current state of usability in the case company, based on interviews with company employees representing the three business units in the company. The analysis covers the usability of current products, the strengths and weaknesses of the software development process concerning usability, and views on the future development of usability practices in the company.

The second part, in Section 4.3, presents an analysis of the action intervention performed during the research, i.e. the usability practices employed in the case project, based on interviews with all team members and the head of development as well as observation during the research. The analysis is dived into seven themes: usability activities in general, the interaction designer role, the product vision and user research, the use of paper prototypes, designing one sprint ahead, usability testing of sprint results, and communication within the team.

4.2 Current state analysis

4.2.1 Interview participants

Nine interviews were conducted with Rapal Oy employees from all three business units. The development and maintenance unit was most strongly represented, with participants including three developers, a tester, a product owner and the head of the unit. Two of the developers also worked as scrum masters. A consultant and the head of training participated from the

services unit and a sales specialist participated from the customers unit. Of the participants, three were women and six were men.

The participants' reported knowledge about the development process at the company varied significantly, with participants from the development and maintenance unit mostly reporting a good or excellent understanding of the process, while other participants reported a general or somewhat lacking understanding. Most of the participants reported at least an average understanding of what usability means, with some mentioning that they understood the concept very well. Usability methods were less familiar, with only about half reporting an understanding of at least some of the methods.

The participants were unanimously of the opinion that usability is very important. Four participants said that it is "very important", one that it is "incredibly important", two that it is "among the most important" attributes of software, and one even that it is "the most important part of software". Several participants stressed that usability is important as a way of differentiating a product from the competition and of gaining new and retaining old customers. One mentioned that the user experience in general is important.

4.2.2 Usability of the current products

Rapal Oy has two main software products, Optimaze.net for managing facilities and Fore for managing infrastructure projects. Optimaze.net is further divided into sub-products, aimed at different user roles. Optimaze.net Moment is the main product and is used by organizations to manage and optimize the use and costs of facilities, as well as to view floor plans. The first version was launched in 2003. Among the later product, Optimaze.net Channel is a product for reporting aimed primarily at management, and Optimaze.net PM allows real estate owners to manage their property, rental agreements and invoicing. Optimaze.net Scenario, as the case project for the research, is not counted among the current products. As Moment is the most widely used product, most answers related to the current products primarily focused on it.

Overall usability of the current products

The interview participants' evaluations of the usability of the current products of Rapal Oy were mixed, and ranged from "usable" to "poor" or "very poor", especially in the case of the oldest product, Moment. Seven participants immediately mentioned that there is improvement to be done, and

Statement	2008	2009	2010
Optimaze.net Moment			
Amount of respondents	79	83	67
I can easily find the information I need	5.1	5.2	5.3
The product is easy to use and user-friendly	4.8	5.0	5.1
The user interface is user-friendly and the layout is pleasing	5.0	5.2	5.2
Optimaze.net Channel			
Amount of respondents	62	56	67
I can easily find the information I need	4.9	4.8	4.9
The product is easy to use and user-friendly	4.7	4.7	4.9
The user interface is user-friendly and the layout is pleasing	4.9	5.1	5.0

Table 4.1: Customer satisfaction data on the usability of Optimaze.net Moment and Optimaze.net Channel. The survey uses a 1–7 Likert scale with 1 indicating disagreement and 7 agreement.

four that Moment has a high learning curve or requires training. However, four participants mentioned that Moment is usable compared to the competition, even "light-years ahead", and that in the competing products functionality is often found "behind a million buttons".

The customer satisfaction data gathered by Rapal Oy give a view of the usability of Optimaze.net Moment and Optimaze.net Channel. Out of the seven questions asked about each product, three concern usability (see Table 4.1). The survey is sent out to all users of the product, and uses a 1–7 Likert scale with 1 indicating disagreement, 4 indifference and 7 agreement. The average score on the three usability-related questions has in the case of Moment increased from 5.0 in 2008 to 5.2 in 2010, and in the case of Channel increased from 4.8 to 4.9. While these results indicate a somewhat positive view of the usability of the products, some interview participants expressed concerns that the answers may not be representative of the average user, as the ratio of respondents is quite low. It is also possible that new or infrequent users are less likely to take the time to fill in the survey, which may skew the results upward.

Usability strengths of the current products

Concerning the usability strengths of the current products, three participants mentioned the graphical floor plan viewer. Two participants mentioned that the products are relatively usable and efficient for advanced users. And while not directly related to usability, two participants mentioned that the products are important and useful for their customers. Other stated strengths were the search functionality in Moment, speed, and a stability that users grow accustomed to. Concerning Fore, participants

stated that it works better for infrastructure calculations than Microsoft Excel, and that the visual design is quite pleasant.

Usability weaknesses of the current products

Almost all participants mentioned several usability problems with the current products, with most answers focusing on Moment. Two participants mentioned that it is important to differentiate between external customers, who are often novice or intermediate users, and internal users, i.e. company employees mostly in the services unit, who tend to be expert users. From the point of view of external users, the most commonly mentioned issue was that the product is illogically structured, with some basic tasks requiring the user to move between several views while keeping important information in memory. Six participants mentioned problems related to the structure. A related concern was learnability. Three participants mentioned that Moment is difficult to learn for new users, and commented that it does not guide users and that it requires extensive training.

Two participants stressed the point of view of the internal users. One approximated that as much as "half" of the work of the internal users and consultants is spent on work that may be considered "unnecessary" or "routine". The participant stated that much of the internal work was due to misunderstandings by external users, but was unsure how much of this can be solved by improving usability. However, the unnecessary work was expanded by inefficient tools and low usability for internal users. The two participants were both concerned that usability improvements that would greatly enhance the internal efficiency constantly get sidelined by requests from customers. Two other participants mentioned that much of the internal work is due to customers not being able to manage some aspects of the software by themselves, e.g. usage roles or permissions.

Some other usability weaknesses were less commonly mentioned. One was that Optimaze.net is a monolithic product. While Channel has a separate web interface, PM shares the user interface with Moment. As there are many user roles with different goals and needs, this means that no single user role is served perfectly by the user interface. Other mentioned problems included that the information provided by the products is not very refined, well summarized or visualized, that the user is given little indication of what is important at the moment, and that the visual design makes the product appear complex.

Knowledge about the usability of the current products

The participants differed somewhat in their view on how good knowledge the company has about the usability of the current products. Three participants mentioned that the internal users and those interacting with customers provide much feedback and improvement suggestions. Two other participants mentioned that many employees have a hunch of the problems. However, six participants were of the opinion that data gathering needs to be improved, and four of these specifically mentioned that the methods should be made more systematic, concrete or research-based. Two participants mentioned the yearly customer satisfaction survey, and both were of the opinion that it is insufficient and does not accurately reflect the average user.

4.2.3 Usability in the development process

Overview of the development process

The development process at Rapal Oy consists of two tracks. The fundamental track includes development and maintenance of products currently in use by customers, including Optimaze.net Moment, Optimaze.net Channel, Optimaze.net PM and Fore, as well as admin tools for internal users. In addition to following a roadmap, a significant part of fundamental development is made up of feature requests by individual large customers. The future track includes development of new products. In it, the focus is more on the long term, and there is more collaboration with outsiders such as research departments.

The development methodology in use at Rapal Oy is Scrum, although it is adhered to closely only in the future development track. Each of the two main products, Optimaze.net and Fore, has a product owner that is responsible for prioritizing requirements. The overall vision is provided by the management team, which consists of the CEO as well as the heads of the three business units.

Most participants described the future development track as following Scrum quite closely. At the time of the research, development was done in four week sprints by a team of two developers, one part-time designer (the author), a scrum master and a product owner. Each sprint produces a fully working and tested product increment, and after a sufficient amount of sprints, the product is released to customers.

The fundamental track was described by four participants as more chaotic, even “ad-hoc”, and only nominally Scrum. One participant de-

scribed how the Scrum methodology is still relatively new in the company, as there has been a gradual change from traditional development to agile development, and that much has been achieved in a short time. Two participants mentioned that the issues with following Scrum may be due to the constant pressure caused by customer projects, i.e. requests for new functionality that are considered so important that they may bypass the process. Another participant mentioned that the fundamental track team, which is larger than the future track team, is less coherent.

Strengths in the development process concerning usability

The following sections concern the fundamental track and future track projects prior to Optimaze.net Scenario. It excludes the Scenario case project, which is analyzed as part of the action intervention in Section 4.3.

While interaction design practices such as user research, iterative prototyping and usability evaluations were not used in the development process, Scrum and agile practices were generally seen as beneficial to usability. Three participants described the agile focus on working in teams and communicating frequently as aiding usability, because it creates a shared understanding of the product and the problems it attempts to solve. Writing requirements as user stories and testing them with acceptance tests was also mentioned as a strength by four participants, because user stories put focus on the needs of users rather than on the technology. A developer also mentioned that development is easier if the goals of the user are communicated clearly. Two participants also mentioned having a single product owner overseeing and prioritizing requirements as beneficial to usability.

Another strength, mentioned by three participants, was the fact that the company employs experts that work as internal customers, and that are able to provide feedback, including feedback on usability, as well as to aid in development. One participant described the in-house interaction of developers and domain experts as one of the underlying ideas of the company. In addition, one participant suggested that it may be useful that the company has many young employees with knowledge about usability.

Weaknesses in the development process concerning usability

Reported weaknesses in the process concerning usability were much more numerous than strengths. The most widely reported problem was too much focus on quickly satisfying customer requests and on short-term gains. Other problems included difficulties creating a unified product vision, lack of sufficient design before development, lack of skill, problems

working in teams, and no user testing.

Five participants stated that the focus in the company on quickly satisfying customer requests by adding functionality to the products made the usability of the products suffer. Even though one of the strengths of the company was the possibility to gain rapid feedback from internal users, two participants stated that in practice this feedback is seldom taken into account, as new functionality requested by customers push it to the bottom of the product backlog. One participant especially viewed the Scrum practice of always developing what is on the top of the product backlog as causing minor but important improvements, especially qualitative improvements, to fall between the cracks, while highly visible new functionality is promoted. Another participant mentioned that this focus may be the result of a period of rapid growth, but that it is not sustainable in the long run. Three participants mentioned that not being able to focus on a certain set of functionality for a full sprint, due to customer interruptions, makes putting effort into usability more difficult. Also, two participants stated that many projects were only one or two sprints in length, which is too short to give time to any kind of usability work, and that many products have been left in an uncompleted state as a result.

A second problem, mentioned by three participants, was a difficulty in creating a unified vision for the products and communicating it effectively to the developers. Two of the participants stated that this is a common problem in agile development, where the focus on doing one user story at a time detracts from pausing and thinking about the big picture. While there are epics, i.e. high-level user stories, their sufficiency to communicate the vision was questioned by one of the participants.

Three participants mentioned as a problem a lack of investment in usability when developing the main products. When adding functionality to existing products, usability is given little, if any, consideration. Two participants also mentioned that there is too little usability knowhow in the company. A lack of teamwork and communication between developers was also seen by two participants as sometimes decreasing the consistency of the products. When developers focus too much on their individual parts of the product, they may create solutions that are individually usable but difficult to use when put together.

Two participants mentioned a problem with finding time for design before development. This was seen as a weakness in moving directly from user stories to coding, as what is coded might not be the best or even a good solution to the user need. One participant mentioned that, when user stories are clarified as late as the sprint planning or during the sprint, there is not sufficient time for design. When there is no specific design

phase before development, as in traditional development, it is too easy to leave out necessary design work when time runs short. A final problem, mentioned by one participant, was a lack of user testing, including not only usability testing but also alpha- and beta-testing before products are released.

4.2.4 Usability in the future

All nine participants were of the opinion that more should be invested in improving usability in the future. The most commonly mentioned reason for investing in usability was that it will increase the amount of customers in the long term, both because selling usable products is easier, and because current customers that are satisfied with the usability of the products will tend to recommend them to their colleagues. Six participants mentioned this as an argument. Another benefit, mentioned by three participants, was that increased usability will make the internal users more efficient in providing services. One participant also mentioned that if internal usability is improved, it is easier to collaborate with other companies, allowing them to provide services while Rapal Oy provides the products. Finally, one participant mentioned that improved usability will decrease the load on customer support.

Concerning drawbacks with investing in usability, five participants stated that it requires greater resources and thereby increases the cost of development. Also, there is often a trade-off between usability and new functionality. However, two of the participants that mentioned increased costs stated that these costs are only short-term, and that in the long term, investment in usability will pay itself back with increased sales and decrease need for support. Other possible problems mentioned were that the company has to learn to work with longer projects and release cycles, and that there is a risk that investment in usability will not be targeted at the products or product areas where it would be most beneficial.

Four of the participants stated that they were optimistic about the company's investment in usability in the future, as user experience is now part of the strategy. However, five participants stated that, even though there is now a clear message from management that usability is important, it is still uncertain how much resources management is willing to invest in usability in practice. There is the risk that, if resources are insufficient, it may be too difficult to decline requests for new functionality, and usability may draw the short straw in the trade-off between quick gains and long-term investment in quality.

4.2.5 Summary

Nine company employees were interviewed on the state of usability in the company. The results are summarized in Figure 4.1. All participants considered usability to be an important or very important concern. There was some disagreement on how usable the current products are, but overall more weaknesses were mentioned than strengths. The most commonly mentioned strengths were the graphical floor plan viewer and efficiency for advanced users, while the most commonly mentioned weaknesses were illogically structured products from the point of view of the user, low learnability, and inefficient user interfaces for internal users.

Both strengths and weaknesses concerning usability were identified in the current software development process. The most commonly mentioned strengths were teamwork and communication, requirements that are written from the point of view of the user, internal users that can provide feedback and product owners that oversee and prioritizes requirements. The most commonly mentioned weaknesses were too much focus on quick fixes and short-term gains, a difficulty to create and communicate a unified product vision, a lack of effort in improving usability, too little usability knowhow, and insufficient design before development.

4.3 Action intervention

4.3.1 Usability activities in general

Interviews

All case project team members as well as the head of development were interviewed after the action intervention. All interview participants agreed that there had been significantly more effort put into usability activities than had been in previous projects. Paper prototyping and usability testing where mentioned in particular. The product owner mentioned that this was her first project where there was feedback from end users before the release. Both the product owner and scrum master were very satisfied with investing more in usability. The head of development mentioned that the project had worked as a model project for testing new usability practices.

Some general problems with usability resourcing were mentioned. The product owner was of the opinion that more could have been done in interviewing users during the early stages of the project. She also thought it difficult to communicate to management that fixing usability problems requires development resources, and thought there was too much pressure

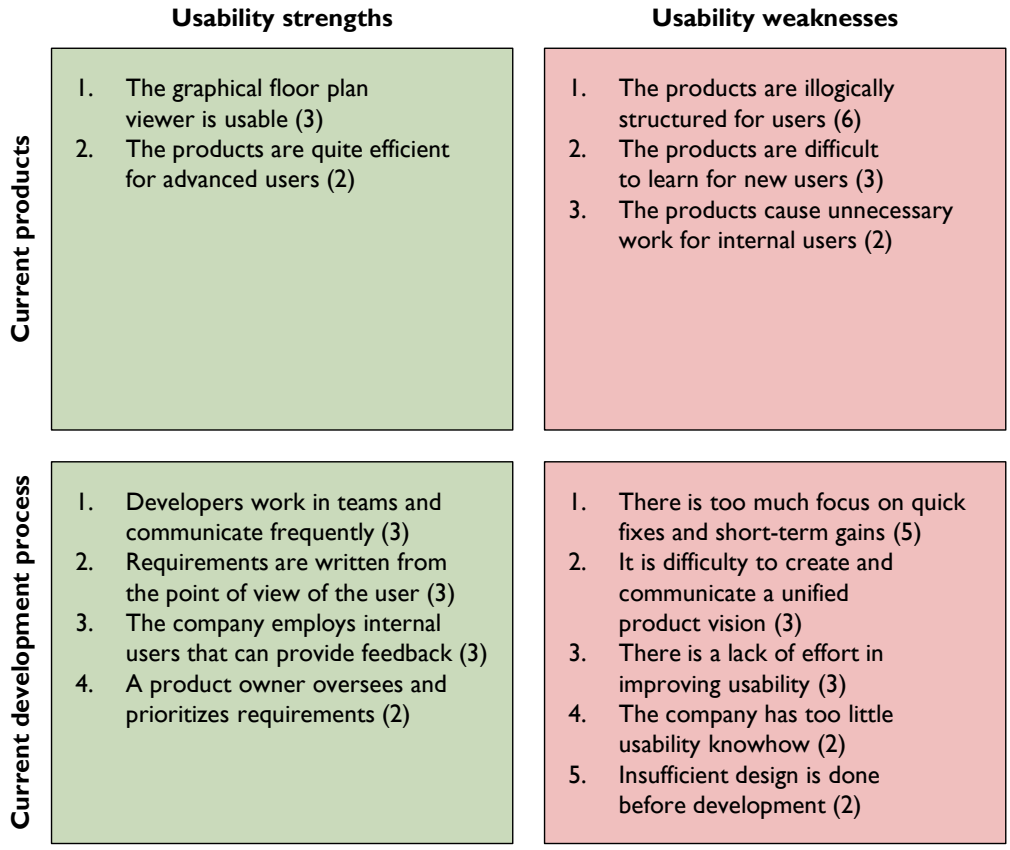

Figure 4.1: The most commonly mentioned usability strengths and weaknesses in the current products and development process. The numbers in parentheses are the amount of participants that mentioned the item.

on adding new functionality. The scrum master thought that spending more time on educating the rest of the team about usability would have been important. He also thought that it would be better to have a full-time interaction designer on the team.

Observation

In general, the interaction designer was given almost free hands in introducing and testing new usability practices. The designer also got support with implementing the practices, including recruiting test participants, arranging and piloting usability tests, rewarding test participants, and acquiring the necessary materials for prototyping. While the rest of the team members did not have that much time for usability practices, the scrum master was able to help significantly with prototyping. The product owner also supported the usability activities to the extent that her schedule allowed.

4.3.2 The interaction designer role

Interviews

All participants where positive about having a person with the designated role of interaction designer on the team, although there were some concerns. Among the most important responsibilities of the designer, the participants mentioned design and prototyping, arranging usability tests, determining user needs, making sure the user interface is consistent, and aiding the product owner in understanding the product. All participants thought the end product would have been of lower quality and less consistent had there been no interaction designer.

There was disagreement among the respondents about whether the responsibility for usability and the user interface was too concentrated in the interaction designer. The developers were neutral or thought that this specialization was useful, and the product owner also thought specialization produced better results. However, the scrum master was concerned that this arrangement made the designer somewhat separated from the team, and made the developers take less responsibility for the user interface, passing on even minor decisions to the designer. The scrum master was also concerned that, though he and the product owner had participated in prototyping, the developers had not participated at all.

The head of development, product owner and scrum master all mentioned that there would have been a need for more usability knowledge

on the team. They mentioned that there were some slowdowns in the development when the designer was not present, and that either a full-time designer or more usability knowledge on the team may have solved this. Also, the head of development mentioned that in the future, there would be a need for more interaction design coordination between individual projects and the entire product portfolio.

Observation

Having a dedicated person on the team with the responsibility for the usability of the product was important, as it guaranteed that sufficient time and effort was put into the necessary usability activities. While activities such as paper prototyping have quite a low threshold and could conceivably have been done well by developers alongside development, any activity that involved interaction with external users required dedicated time for planning and execution. More immediate concerns often trump this longer-term thinking. This was noticed several times during the project when the designer was asked to aid in tasks such as front-end development, bug fixing and technical testing. The product owner also would not have had time to plan or arrange these activities.

4.3.3 The product vision and user research

Interviews

Most interview participants stated they had a good understanding of the product vision throughout the project, although one developer mentioned that this was not the case, especially in the beginning of the product. The same developer also mentioned that the team's understanding of the target users' needs was lacking, and that this caused unnecessary changes during the project. Likewise, the head of development also stated that it was unclear who the product users really are. He also stated that some of the big questions concerning user needs had not been resolved during the project. Concerning the usability goals of the project, all participants had a moderate understanding of usability goals. However, the head of development mentioned that the goals were not concrete, and both he and the scrum master mentioned the difficulty of creating concrete usability metrics.

When asked about initial planning, the head of development stated that a clear vision and understanding of users is very useful when making business decisions about whether to invest in projects, and that it is easier to

make good decisions if prototypes are created before development begins. All participants thought a better understanding of the target users would have been useful before development began, but both the product owner and scrum master were skeptical about how much can be learned before development has started and there is something concrete to test or discuss.

Observation

Some effort was put into understanding the users before development began. The different user roles of the product were clarified in the beginning and two rounds of paper prototyping was done during the initial planning stage. The prototype tests included some discussions with the users about their needs. The product vision was also quite clearly defined, and high usability was stated as one of the most important goals in the vision.

There were also several problems concerning user research. Early contact with users was achieved only through prototype testing, and not through interviews or observation of how the users currently solve the problems the product focused on. From this followed that the focus was more on validating initial design ideas than on understanding the users and their needs. Also, some of the users participating in the tests were probably not that representative of the target users of the product. Late in the project, an interview was conducted with a user that was able to explain his needs much better than any of the users during the beginning of the project. Gaining this understanding earlier would likely have reduced the amount of change and improved the product.

In addition, during the beginning of the project, the understanding of the users was mostly limited to the product owner, interaction designer and scrum master. The developers did not participate in the initial planning or prototype testing, and user modeling methods such as personas were not used. The usability testing in sprint 3 was the first time the entire team saw a representative user interact with the product or a prototype thereof.

4.3.4 The use of paper prototypes

Interviews

All participants agreed that paper prototyping was a good way of designing the product. The head of development and product owner mentioned that paper prototypes are very useful because they visualize the direction of development better than user stories, and make it easier to improve the design before code is written. Both the head of development and the scrum

master were of the opinion that paper prototyping is a very cost-efficient way of designing. Both developers and the scrum master mentioned that paper prototypes provide sufficient information for development, and make it easier to estimate user stories.

Some problems with paper prototyping were also mentioned. The most commonly mentioned issue was that paper prototypes could not depict interactivity (such as drag-and-drop) well. One of the developers mentioned that the relationship between user stories and prototypes sometimes was difficult to understand, as one prototype included multiple stories. Annotating each part of the prototype with the user story it belonged to helped with this problem. Paper prototypes were also difficult when doing remote work, as they had to be digitized to be accessible remotely. Finally, the scrum master was concerned that the developers sometimes perceived the prototypes too strictly as a specification, and therefore thought less themselves about the usability of the product.

Observation

Paper prototyping was a very useful method for designing the product. Creating paper prototypes was quick, efficient and fun. Compared to higher-fidelity prototypes, drawing on paper was so flexible that it did not unnecessarily restrict the possible solutions to those that are easily implemented. Working on the prototype together with the product owner and scrum master also made it possible to quickly comment on and improve each other's ideas. Many of the best solutions were achieved this way. It was also easier to resolve disputes about the design, because when ideas were made concrete in the prototype, it was often easier to see which idea would likely work best.

Testing the prototype with end users was also useful. Preparing the prototype for the tests was time-consuming, but forced the designer to think through how the prototype would actually be used, which already found some problems. Testing the prototypes also found multiple problems. Some users, however, did not completely understand the idea of using the prototype, and preferred discussing it in general terms, which was less useful. While the tests produced much data, it was difficult to communicate some of the findings to the team. Testing would likely have been more useful if at least the product owner had been able to participate in the test sessions.

A paper prototype also worked well as a way of communicating the design to developers. There were some misunderstandings about the prototype, but these were generally resolved by communication during the

sprint. Some important misunderstandings concerned more complicated workflows in the product, as these could not be easily captured in the prototype. Paper prototypes also worked reasonably well as a way of communicating to stakeholders outside the team, although this depended a lot on the person. While some stakeholders understood the idea of paper prototypes well, others were not that interested, and paid much greater attention when they saw a working product.

The benefits of paper prototyping were seen most clearly in sprint 3, whose user stories had not been prototyped before the sprint began. This caused several misunderstandings, and even when the user stories were correctly understood, some of the solutions were not ideal from the user's point of view. Some of the stories therefore had to be redone in sprint 4, this time with paper prototyping done first.

4.3.5 Designing one sprint ahead

Interviews

All participants agreed that it was best to design one sprint ahead of development. The head of development and the product owner both mentioned that in previous projects the sprint demo often was the first time they saw some of the functionality, and at that point changes could be too time-consuming to implement. Designing one sprint ahead made it possible to improve the design when it was still quick and inexpensive. Also, the product owner mentioned that although in theory she could guide the development through the entire sprint, in practice because of her schedule this was seldom possible. Designing one sprint ahead gave her time to validate the design before it was implemented.

According to all participants, a paper prototype was an appropriate level of design before the development sprint begins. One of the developers and the scrum master mentioned that the simplest stories may not require a prototype at all. The product owner stated that a higher-fidelity prototype seldom is required before the sprint begins, as the design can be refined during the sprint. The product owner and scrum master both mentioned that it was easier to refine the user stories, update acceptance tests and notice missing stories based on a prototype than without one. The scrum master also noted that well written user stories, i.e. user stories stating the need of the user without specifying anything about the implementation, seldom needed to be updated as the prototype progressed.

Observation

Designing one sprint ahead worked much better than designing in the same sprint. Most importantly, it provided sufficient time to refine and test the prototype. This would not have been possible during the same sprint, as it would have created a bottleneck with the developers waiting for the first stories to be designed and tested. Also, it was often useful to design on the sprint level rather than the story level. Even if the user stories were independent, their designs often affected each other. If the design would have been done one story at a time during the sprint, it would have been more difficult to keep the user interface consistent. Finally, approximately one sprint of functionality was the smallest increment that was reasonable to test in one round of prototype testing. Testing the designs for individual stories during the development sprint would not have been practical.

4.3.6 Usability testing of sprint results

Interviews

The product owner, scrum master and both developers thought that viewing the recording of a usability test and analyzing it together with the team was very useful, and that viewing a recording instead of the live test provided a sufficiently good understanding of the test results. The product owner thought it was enlightening for the entire team, especially the developers. One of the developers stated that viewing the test provided a good understanding of the usability problems in the product. The scrum master mentioned that it was possibly the single most important usability activity because it helped the entire team understand user behavior.

As a problem with usability testing, the product owner mentioned the difficulty of finding good participants, and the head of development also expressed a concern that not having actually representative users as participants would provide misleading results. The product owner also thought that too few of the usability test findings could actually be fixed in the product, due to a lack of time. Finally, one of the developers mentioned that viewing only one test was not enough to give a good view of the range of users using the product.

Observation

Testing the working product in addition to testing the prototype was useful, because it found problems that could not have been found using a paper prototype. Some of these problems were related to user input or visual

design; others were due to unexpected user behavior that would not have been possible in the paper prototype. One of the users that participated in the working product tests also later participated in a prototype test. The former was more useful, because in the latter the user did not interact as much with product.

Watching and analyzing one of the tests with the team was also very useful. The developers were interested in the results, because they knew the product well. The list of prioritized usability problems that the analysis produced was also more useful than the list the designer produced independently before that, both because it included new important problems, and because it was based on the shared understanding of the team. Many of these problems were solved in later sprints, but some important problems were never fixed due to lack of time.

4.3.7 Communication within the team

Interviews

The participants mentioned several problems with usability communications within the team. The developers mentioned that there were sometimes significant delays when they needed information from the designer, especially during the beginning of the project. The scrum master emphasized the importance of participating in daily scrum meetings, stating that at times when the designer did not participate the usability activities became somewhat invisible to the team, and this lessened team cohesion. This also led to problems developing a shared understanding of the expected level of usability, and there were misunderstandings when something was implied by the designer but not clearly included in the prototype.

Also, the project wiki did not work that well as a way of communicating usability test findings to the team. The scrum master mentioned that it in some cases became a place to put information without anyone ever looking it up. The paper prototype wall was much better received. One of the developers mentioned using it continuously, and the product owner and scrum master both thought it useful that it increased the visibility of what is being developed, both within and outside the team.

Observation

The most significant communication problems related to usability occurred when the designer was not available to answer the developers' questions for longer periods of time. Suggesting changes to the implementation only

after it was done sometimes caused disagreements, as the developers experienced that they had done the implementation according to their best knowledge, and had already moved on to new tasks. A better understanding of the users and of usability in general throughout the team would likely have resolved some of these communication problems.

4.3.8 Summary

Several new usability practices were tested in the case project. These included a dedicated interaction designer on the team, paper prototyping one sprint ahead of development, and usability testing sprint results. In general, these practices were very successful, although there were some issues. Paper prototyping one sprint ahead of development was well received, because it was cost-efficient, encouraged better design, and allowed ideas to be validated before implementation. It also gave the team and stakeholders better visibility of the direction of development. Usability testing sprint results produced many useful findings, and analyzing the tests together gave the whole team better insight into user behavior and usability problems in the product. Problems in the case project concerning usability included insufficient user research in the beginning and communication issues.

Chapter 5

Recommendations and conclusions

5.1 Introduction

This chapter answers the research question by presenting a set of recommended practices for improving the usability of products created by the company. The recommended practices, which are listed in Section 5.2, are based on the findings of the literature review, the current state analysis and the action intervention. Each practice is further divided into sub-practices that clarify how it should be implemented. In addition, each practice has a section describing the justification of the practice in the research. Section 5.3 discusses how the practices should be combined, and Section 5.4 concludes the research.

5.2 Recommended practices

5.2.1 Include an interaction designer on the team

Assign the role of interaction designer to one member of the team

This person should have the necessary skills for the job, and should work at least half-time, but ideally full-time, with usability activities on the team. The interaction designer role is an addition to the standard Scrum roles of product owner, scrum master and developer. The skills and focus required are sufficiently distinct to warrant making it a separate role, rather than a specialization of the developer role. However, the interaction designer should still be considered equal as a team member, neither superior nor inferior to the developers.

Make the designer responsible for the usability of the product

The interaction designer should be the primary person responsible for the usability of the product, in the same way the product owner is responsible for the business value and the developers for the technical implementation. However, in cases where there are significant trade-offs between usability and functionality or other quality attributes, such as security, it is still the responsibility of the product owner to decide what is most important for the product as a whole. Also, even if the interaction designer has primary responsibility, the entire team should understand that the work of every team members ultimately affects the usability of the product.

Make a developer designer if no specialized designer is available

The developer should work at least half-time on usability activities instead of development. While it is not ideal to have an interaction designer without the specialized skills, it is better than having none at all. Some usability benefits can already be gained just by performing usability activities at all, and the basics do not take that long to learn. However, this requires that time is reserved for the activities, as they cannot be done as an afterthought.

Justification

Having a dedicated interaction designer role on the team is widely supported in the literature. Beyer, Holtzblatt, and Baker (2004) state that agile teams require that design skill is available. Nielsen (2009a) states that usability should be part of the agile team, and Gulliksen et al. (2003) and Kreitzberg and Little (2009) recommend that teams include a usability champion. Nielsen (2007) states that, while a dedicated usability specialist is to be preferred, having developers take on usability work is much better than having no one responsible for usability. Finally, Federoff and Courage (2009) found that designers should work on at most two agile teams at a time.

The current state analysis supported the idea of an interaction designer role. Reported problems included too little usability knowhow and a lack of investment in usability. These problems are at least partially caused by no one being responsible for the usability of the product. The designer role was also positively received in the action intervention. It was found that the role made it possible to focus on usability and to dedicate sufficient time and effort to usability activities.

5.2.2 Understand the users before development begins

Create and communicate a clear product vision

A precondition to understanding the users of a product is to have a clear vision of what the project should be. This includes understanding who the target user are, what user needs the product attempts to fulfill, and in what general way the product attempts to do so. Creating the product vision is primarily the responsibility of the product owner, but the interaction designer should help the product owner create and clarify the vision, and especially help keep the vision focused on actual users and their needs. The product vision should also state the usability goals of the product, i.e. the necessary level of usability to fulfill the product vision. Work on the product vision should be started before the project begins, and the vision should be clearly understood by all team members at the end of the first sprint.

Understand the target users before development begins

In the first sprint, before development begins, the focus of the interaction designer should be on understanding the target users of the product, including their goals, needs and behavior. Together with at least the product owner, the designer should start by identifying and prioritizing the different user roles of the product. The focus should then be on understanding the one or two most important roles. For new products, users that are representative of the roles should be interviewed, or the designer should observe how they currently solve the problem. For improvements to current products, the designer can instead observe how the users use the current product. Preferably, all observation should be done in the user's natural work environment, using contextual inquiry. The product owner should participate in most of the user research, and other team members may also participate to some extent.

Communicate the product vision and user research clearly to the team

The product vision and user research should be communicated clearly to the entire team. To communicate user research, techniques such as personas can be used. The most important parts of the vision and user research findings should be kept in a visible location, e.g. on a wall in the working area. The vision and user research findings should also be regularly updated whenever new important information about the target users is learned during the project.

Justification

Pichler (2010) emphasizes the importance of creating a clear product vision in agile projects. Budwig, Jeong, and Kelkar (2009) and Nielsen (2009a) recommend quarterly to annual design vision sprints to update the vision. Including end users early is among the most fundamental usability principles (e.g. Gould and Lewis, 1985). Beyer, Holtzblatt, and Baker (2004), Sy (2007) and Nielsen (2009a) all recommend some initial user research in agile projects before development begins, and point out that users should be observed in addition to listened to.

The current state analysis found that too much focus on quick fixes and short-term gains as well as a difficulty to create and communicate a unified product vision were problems in the company. Regular vision and user research sprints are a possible solution to this, as they temporarily change the focus from short term to long term. The action intervention also found that a clear product vision was beneficial, but that the usability goals were somewhat lacking. There was also insufficient user research early in the project, and information about the users was not communicated clearly enough to the developers.

5.2.3 Design primarily using iterative paper prototyping

Design every user story as a paper prototype

Before development on a user story begins, it should be designed as a paper prototype. The paper prototype should be hand-drawn, and it should include the level of detail necessary to communicate all significant aspects of the design. The prototype should be based on realistic data, and it should cover all steps necessary to complete the user story. Multiple user stories may be designed using the same prototype. The paper prototype should primarily be created by the interaction designer, but the product owner should regularly participate to ensure that the prototype actually satisfies the user stories. The scrum master and developers may also participate in the paper prototyping.

Discuss the prototype with the team and other stakeholders

When a paper prototype has been created, it should be discussed at least with the product owner, and preferably also with other team members as well as any relevant domain experts or other stakeholders outside the team. Discussions with the product owner and domain experts should focus on whether the prototype satisfies user needs, uses the correct terminology, and

implements the domain logic correctly. Discussions with the developers should focus on whether the prototype is feasible, how much development effort it requires, and whether there are better technical solutions to any part of it. Based on the discussions the prototype should be improved and, if necessary, discussed again.

Test the prototype with two or three representative users

It is not enough to discuss the paper prototype; it should also be tested with representative end users. Once a sprint, the prototype should be tested with two or three end users that are representative of the user role or persona the prototype is designed for, and improved based on the findings. The tests should be simple exploratory tests, about one hour in length, where the user first performs realistic tasks based on the user stories covered in the prototype while thinking aloud, and after that is asked to comment on the prototype in general. The designer and product owner should participate in the tests, and the tests may be recorded so that they can be viewed by other team members and stakeholders afterwards. If it is not possible to recruit representative users, it is better to test the prototype with company employees than not at all. In that case, the participants should neither be developers nor directly involved in the project.

Communicate the prototype clearly to the developers

When the prototype is done, it should be communicated clearly to the developers. The designer should walk through the prototype with the developers, and answer any questions they have. After that, the prototype should be put in a visible location, for example on a wall in the working area. The prototype should be annotated with notes explaining interaction that is not obvious. The different parts of the prototype should also be marked with the user stories they cover, so that the connections between stories and prototype are clear. Also, in cases where the navigation or workflows in the prototype are complicated, these should be communicated with additional artifacts, such as navigation maps and workflow diagrams.

Justification

Gulliksen et al. (2003) state the importance of simple design representations. Rubin and Chisnell (2008) explain how exploratory usability tests can be performed early to test high-level aspect of the design. Beyer, Holtzblatt, and Baker (2004) and Nielsen (2008a) emphasize the importance in agile projects of testing the design before it is coded. Federoff and Courage (2009)

explain how a prototype can work as the specification for developers, and Meszaros and Aston (2006) praise paper prototypes for making the product vision tangible.

The current state analysis found that one of the strengths of the company was that it employs internal experts that can provide useful feedback on the product. This feedback can be immediately utilized if it given based on a prototype, before development. In the action intervention, paper prototyping was a useful and cost-effective method for designing the product and evaluating the design. Paper prototypes were useful for visualizing the direction of development, and aided in estimation and development. The prototype wall also worked well.

5.2.4 Design one sprint ahead of development

Design a paper prototype of every user story before a sprint begins

Paper prototypes covering every user story that may be developed in a sprint should be created, discussed and tested in the sprint prior to the development. The prototype should be completed before sprint planning, so that the sprint can be planned based on the prototype. As the final estimate of how much can be included in a sprint is done during sprint planning, there will need to be prototypes for a few extra stories that may not fit into the sprint. After the sprint has begun, only minor changes may be made to the prototype. Any changes that significantly affect development effort should be included in the design for later sprints. However, the prototype may be elaborated on during the development sprint, for example by creating a high-fidelity prototype to communicate visual design.

Do not develop any user stories in the first sprint

Because every user story should be designed before a sprint begins, no user stories should be developed in the first sprint of a release. This is necessary to allow sufficient time for design before development begins. The developers should still be present during the sprint, so that they can discuss the prototype and estimate effort. The developers may also perform activities that do not produce user-visible functionality, such as preparing the development environment, planning the architecture, refactoring the current code and, if necessary, fixing bugs in the previous release.

Update, estimate and re-estimate user stories as the design progresses

As the prototype progresses during the sprint prior to development, the user stories and their acceptance tests should be updated by the product owner to reflect the most recent understanding. When the sprint begins, the user stories and prototype should be in agreement. Also, the user stories should regularly be estimated and re-estimated by the developers as the prototype progresses. A more accurate prototype allows developers to give better estimates, and this in turn allows the product owner to better plan the stories that will fit into the sprint, and the designer to know how many stories need to be designed before the sprint begins.

Justification

The practice of designing one sprint ahead of development was first proposed by Miller (2005) and Sy (2007), and has since become one of the most widely recommended agile usability practices in the literature. It is recommended e.g. by Budwig, Jeong, and Kelkar (2009), Kreitzberg and Little (2009) and Nielsen (2009a), who includes it as one of the two most important recommendations for ensuring good usability in agile projects. It is also recommended by agile author Cohn (2010) as well as Federoff and Courage (2009), who specifically state that it is beneficial for enterprise software.

One of the problems found in the current state analysis was that too little design was done before user stories were developed. The action intervention found that designing one sprint ahead allowed for enough time to prototype, discuss, test and iterate on designs. It made it easier for the design work to take into account the sprint as a whole, rather than just individual stories. And, critically, it allowed the product owner to update, add, remove and reprioritize user stories based on the new information gained through prototyping and prototype testing.

5.2.5 Usability test sprint results with representative users

Test the usability of sprint results with two or three representative users

In addition to testing the paper prototype, the usability of the working product should also be tested. This makes it possible to find usability problems that cannot be found in paper prototype testing, such as most problems related to interaction, visual design and performance. Once a sprint, the product increment developed in the previous sprint should be tested with two or three end users that are representative of the appropriate

user role or persona. Simple assessment tests should be used, in which the user is given realistic tasks based on the user stories developed in the sprint, and asked to perform them while thinking aloud. Compared to prototype testing, the focus should be mostly on observing the user's behavior while interacting with the product, and not on getting the user's opinions. As with prototype testing, if it is not possible to recruit representative users, it is better to test the prototype with company employees than not at all, as long as the employees are not developers or directly involved in the project.

View and analyze the tests with the entire team

The entire team should view the usability tests of the working product, either live (without disturbing the tests) or from recordings afterwards. This gives the entire team, including the developers, the possibility to better understand the behavior of the product's users. It is also easier to feel empathy for the user and to understand the importance of various usability problems after viewing an actual user interacting with the product. After viewing the tests, the entire team should together analyze them to come up with a list of the most important usability problems. In this way, each team member thinks through the usability of the product and the problems they observed.

Take into account findings when designing for the next sprint

Any important usability problems found in the tests should be taken into account when planning the following sprint. If the problems are so important that they endanger the usability goals of the product, they should be written as new tasks and prioritized at the top of the product backlog. These should then be included when designing the prototype for the next sprint. If the usability problems are less important, they may be written as new user stories or included into existing user stories, and prioritized separately by the product owner.

Justification

A common recommendation in the literature is to test the usability at different levels of development, both with low-fidelity prototypes and working code. Rubin and Chisnell (2008) recommend assessment tests, in addition to exploratory tests, to evaluate the usability of lower-level operations and aspects of the product. Krug (2006) recommends three users per test, but states that even testing with one user is much better than not

testing at all. Beyer, Holtzblatt, and Baker (2004) also state that, in addition to testing before coding, the code should also be usability tested when needed. Finally, Ferreira, Noble, and Biddle (2007a) found the completion of sprints a valuable opportunity to test the usability of working software early in the project.

The current state analysis did not find anything to directly support usability testing of sprint results, other than the perceived low usability of current products. The action intervention, however, found the practice useful. Testing the working product found several important usability problems, and it allowed the team to view users struggling with the product they had created. However, it was also found that important usability problems need to be put on top of the product backlog, to ensure that the test results are taken into account.

5.2.6 Support development through regular communication

Have the designer regularly available for feedback

Because a paper prototype is a low-fidelity design that leaves open some assumptions and cannot communicate certain aspects of the design at all, it is important that the designer does not simply hand off the prototype to the developers. Rather, the designer should be regularly available throughout the development sprint to support the developers, by explaining and elaborating on the design and by answering questions about the design and the best way to implement it. The designer should participate in the daily scrum meetings, so that there is at least one moment reserved for team communications each day. Participating in the same meeting every day also improves team cohesion. The designer should sit with the team or close to it, to lower the communication threshold. Finally, the designer should follow up how story development progresses, and comment on and suggest improvements to stories while they are still in development, rather than criticize them after they are done.

Spread usability knowledge to the entire team

The designer cannot be constantly available to help development, so it is important that the entire team learns the basics of developing usable products. The designer should be responsible for spreading knowledge about usability to all members of the team, either through education or through involving team members in usability activities, such as user research, usability tests, prototyping sessions or design studios. In turn, the

designer should also strive to learn about the development of the product and how development considerations affect the design.

Justification

The importance of designers and developers working as a team and communicating frequently is stressed by multiple authors. Miller (2005) state that daily interaction was critical to success and Nielsen (2009a) state that designers should be co-located with other team members. Cohn (2010) considers it essential that designers are part of the team and also focus on the current sprint, in addition to designing ahead. Finally, Ungar and White (2008) and Federoff and Courage (2009) emphasize the benefits of including the entire team in the design, thereby spreading usability knowledge to all team members.

The current state analysis found that one of the strengths of the development process concerning usability was teamwork and frequent communication. The action intervention also found communication critical, with problematic outcomes when the designer was not sufficiently available. Insufficient developer education in usability and involvement in the usability activities also caused some problems in the project.

5.3 Combining the practices

To implement all the recommended practices successfully, it is necessary to combine them into a consistent whole. This can be visualized in several ways. Figure 5.1 shows how the practices are combined from the point of view of the project schedule. In it, the responsibilities of the interaction designer are show in a separate track, next to the product owner and the developers. The activities performed by the designer vary depending on the sprint, with the first, second and last sprint of a release containing different usability activities than the middle sprint.

The schedule shows that implementing all the usability practices in a release requires that the release is at least four sprints long. With a four sprint release, the highest priority user stories are designed in the first sprint, developed in the second sprint, usability tested in the third sprint and improved in the fourth sprint. Although the recommended practices ensure that every user story is prototyped and the prototype is tested, only the stories from the first sprints will be usability tested in code. Including more sprints in a release will allow the team to test the usability of a larger part of the working product.

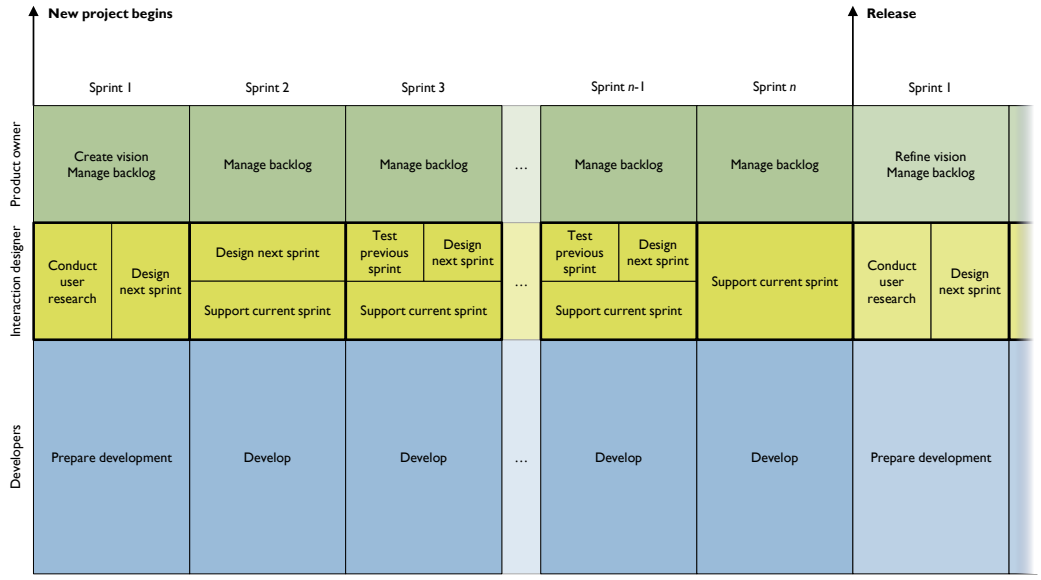

Figure 5.1: Combining the recommended practices in the project schedule. In a release with *n* sprints, the interaction designer conducts user research in sprint 1, designs using iterative paper prototyping in sprints 1 to *n*-1, and tests the usability of sprint results in sprints 3 to *n*-1. Sprint *n* does not include usability testing, as there is no more time in the release to take into account the findings.

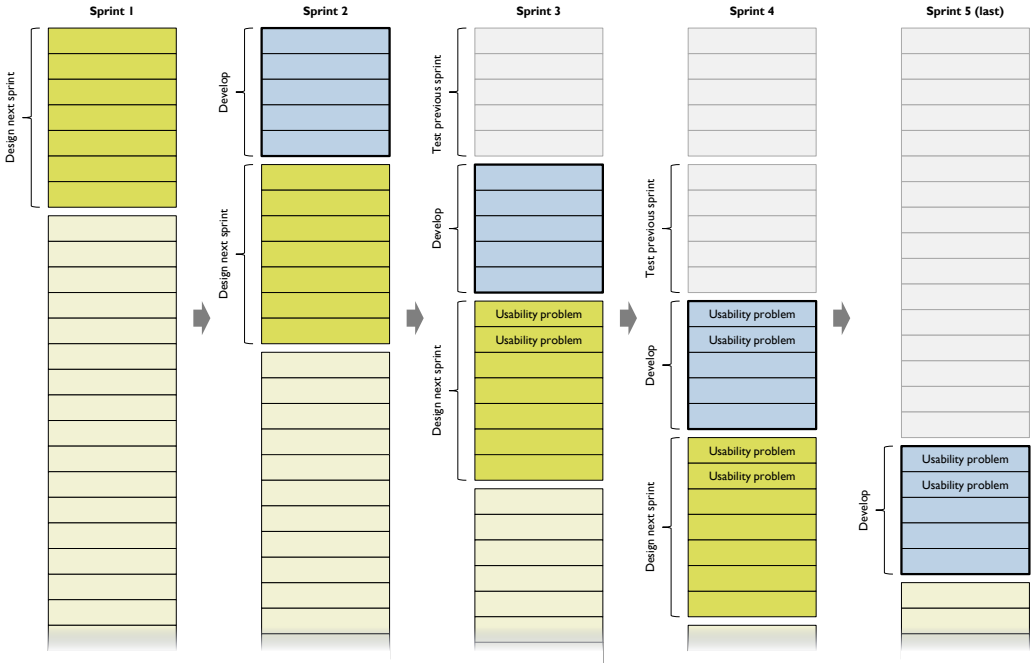

Figure 5.2: The recommended practices from the point of view of the product backlog, in a five sprint release. User stories are designed in the sprint before development and usability tested in the sprint after development. Important usability problems found in testing are included in the design for the next sprint. Stories developed in the last two sprints are not usability tested in the release.

Another way to visualize the practices is shown in Figure 5.2. This figure shows how user stories in the product backlog are designed, developed and usability tested as the release progresses. In the case of a five sprint project, stories developed in the second and third sprints will be usability tested and improved in the same release, while stories developed in the last two sprints will not, unless additional sprints are added that focus solely on fixing usability problems. However, in agile development, the most important stories are in general developed first, so this process ensures that the most important parts of the product have been tested. The figure also shows that somewhat less new functionality can be developed from the fourth sprint onwards, as part of the design and development effort will be put into fixing usability problems found in the tests.

5.4 Conclusions

Improving the usability of the products developed by the case company requires the successful integration of interaction design and agile software development. This requires the adoption of usability practices that not only promote usability, but also fit well with agile methods and principles and are welcomed by stakeholders. Through a review of the literature on agile usability, an examination of the current state of usability in the company, and the testing of new usability practices in a case project, this thesis has arrived at an answer to the research question in the form of six recommended practices.

First, each team should include an interaction designer role responsible for the usability of the product. Second, during the first sprint of every release the entire team should focus on creating a clear product vision and understanding the goals, needs and behavior of the target users of the product. Third, paper prototyping should be the primary method of design, and prototypes should be discussed, tested and communicated clearly to the team. Fourth, design should be done one sprint ahead of development, and should be intertwined with the process of managing the backlog and estimating user stories. Fifth, sprint results should be usability tested after each sprint, and the test findings should be taken into account when designing for the next sprint. Sixth, there should be regular communication between the designer and the rest of the team, to ensure that the design is implemented successfully and that knowledge about usability is spread throughout the team.

Chapter 6

Discussion

6.1 Limitations

Several limitations should be taken into account when interpreting the results. First, the action intervention was performed within only one case project with one team in one company. This makes it difficult to assess which results are applicable only to the particular context and which are more general in nature. It is possible that the personalities and skills of the people involved play a large part in how they implement and react to different practices, and that the results therefore would have been different in another team or another company.

Second, the role of the researcher as an employee at the company may have affected the research. The researcher had a considerable interest in the success of the action intervention and the project overall. This is an inherent weakness in the action research methods, because of its dual scientific and practical goals. To the extent possible, efforts were made to minimize the risk of this affecting the research, as the long term benefits of answering the research question were of greater importance to the company than any short term benefits the research may have bestowed on the case project and team.

Third, the selection strategy for the interviews, especially the current state analysis, may have affected the research. While an effort was made to select interview participants with diverse viewpoints, a secondary goal was to find participants that may have important insight into the questions covered by the interviews. It is possible that this may have skewed the results to emphasize the benefits of investing in usability and the problems in the current software development process.

6.2 Implications

The results of this thesis contribute to an overall positive outlook for introducing interaction design in agile organizations, with the goal of improving the usability of products. While at first it may appear that some of the recommendations are not agile in nature, in fact most of the practices fit well with agile values and principles, and many of the apparent conflicts are only superficial. This is noticed by comparing the recommended practices with the values in the agile manifesto (Beck et al., 2001):

- *Individuals and interactions over processes and tools*. Giving responsibility over usability to the interaction designer role is a way of promoting the skill and motivation of individuals, and regular face-to-face communication and collective learning promotes team interactions. None of the recommended practices put an excessive emphasis on processes or tools; simple solutions are preferred.
- *Working software over comprehensive documentation*. Creating a paper prototype before development is different from traditional comprehensive documentation. A paper prototype is lightweight and shares many of the benefits of working software, including, crucially, the ability to gather realistic feedback.
- *Customer collaboration over contract negotiation*. Customer collaboration is enhanced considerably by having actual end users participate in the development process in user research as well as usability tests of both the prototype and the working product. Observing users in addition to listening to them makes collaboration more fruitful.
- *Responding to change over following a plan*. Responding to feedback from usability tests is one way of responding to change. And while up-front envisioning and user research are planning ahead, both focus on user needs and other factors that change slowly compared to design and implementation.

The results and recommendations are also in agreement with much of the literature on agile usability. Many of the most commonly suggested practices in the literature, such as parallel tracks (e.g. Sy, 2007), little design up front (e.g. Beyer, Holtzblatt, and Baker, 2004), low-fidelity prototyping (e.g. Snyder, 2003) and close collaboration (e.g. Cohn, 2010), are supported by the research. This contributes to the idea that, while there are still many unanswered questions in the field, some of the main findings of the past decade are widely applicable.

6.3 Future work

Based on the findings of this thesis, there are many possibilities for further research in the field of combining interaction design with agile methods. In addition to the general theme that the field needs more empirical research and still lacks any controlled experiments (Silva et al., 2011), there were many topics and specific questions raised during the research that would benefit from more examination. These include:

- *Sprint lengths*. How do various sprint lengths affect the recommended practices? How different are two-week or even one-week sprints from four-week sprints from the point of view of interaction design?
- *Up-front design*. How much time should be spent on design up front, and how is this affected by the type of product, the project and other parameters? In which circumstances should the entire team be involved from the start, and in which is it better to restricts initial design to a smaller group, e.g. the product owner and interaction designer?
- *Usability goals*. What is the best way to set and follow up usability goals in agile projects? Can the goals be followed up in the usability tests, even if only a few users participate during each sprint? Can concrete metrics be used? How would metrics during product development differ from metrics on live products?
- *User experience*. How does a broader focus on user experience instead of usability affect the practices? In particular, when, how and by whom should visual design be done? What other aspects of user experience should be considered?

6.4 Final comments

Whenever specific practices on how to do something in agile development are proposed, it is important to keep in mind a few things. First, agile is a mindset more than any specific process or collection of practices. Without fostering an agile mindset in a company, it will be difficult to achieve the full benefits of specific practices. Second, agile is about motivated and self-organizing teams. A company should weigh the benefits of imposing new practices against the drawbacks of reducing the control employees have over their own work. Third, agile is about continuous improvement. The day any single set of practices are seen as final is the day a company is no longer agile. Keeping these considerations in mind while implementing the recommendations will ensure the greatest benefits.

References

Ambler, Scott W. (2008). "Has agile peaked?" In: *Dr. Dobb's Journal*. URL: http://drdobbs.com/architecture-and-design/207600615.

Avison, David E. et al. (1999). "Action research". In: *Communications of the ACM* 42.1, pp. 94–97.

Beauregard, Russell and Philip Corriveau (2007). "User experience quality: a conceptual framework for goal setting and measurement". In: *Lecture Notes in Computer Science* 4561, pp. 325–332.

Beck, Kent and Cynthia Andres (2004). *Extreme Programming Explained: Embrace Change*. Upper Saddle River, NJ, USA: Pearson Education, Inc.

Beck, Kent et al. (2001). *Manifesto for Agile Software Development*. URL: http://agilemanifesto.org/.

Beyer, Hugh and Karen Holtzblatt (1998). *Contextual Design: Defining Customer-Centered Systems*. London, UK: Academic Press.

Beyer, Hugh, Karen Holtzblatt, and Lisa Baker (2004). "An agile customer-centered method: rapid contextual design". In: *Lecture Notes in Computer Science* 3134, pp. 527–554.

Budwig, Michael, Soojin Jeong, and Kuldeep Kelkar (2009). "When user experience met agile: a case study". In: *CHI EA '09 Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*. New York, NY, USA: ACM, pp. 3075–3084.

Chamberlain, Stephanie, Helen Sharp, and Neil Maiden (2006). "Towards a framework for integrating agile development and user-centred design". In: *Lecture Notes in Computer Science* 4044, pp. 143–153.

Cohn, Mike (2004). *User Stories Applied*. Boston, MA, USA: Pearson Education, Inc.

— (2010). *Succeeding with Agile*. Boston, MA, USA: Pearson Education, Inc.

Cooper, Alan (2004). *The Inmates are Running the Asylum*. Indianapolis, IN, USA: Sams Publishing.

Cooper, Alan, Robert Reimann, and Dave Cronin (2007). *About Face 3: The Essentials of Interaction Design*. Indianapolis, IN, USA: Wiley Publishing, Inc.

Dybå, Tore and Torgeir Dingsøyr (2008). "Empirical studies of agile software development: a systematic review". In: *Information and Software Technology* 50.9-10, pp. 833–859.

Federoff, Melissa and Catherine Courage (2009). "Successful user experience in an agile enterprise environment". In: *Lecture Notes in Computer Science* 5617, pp. 233–242.

Ferreira, Jennifer, James Noble, and Robert Biddle (2007a). "Agile development iterations and UI design". In: *AGILE '07 Proceedings of the AGILE 2007*. Washington, DC, USA: IEEE Computer Society, pp. 50–58.

— (2007b). "Up-front interaction design in agile development". In: *Lecture Notes in Computer Science* 4536, pp. 9–16.

Ferreira, Jennifer, Helen Sharp, and Hugh Robinson (2011). "User experience design and agile development: managing cooperation through articulation work". In: *Software: Practice and Experience* 41.9, pp. 963–974.

Fox, David, Jonathan Sillito, and Frank Maurer (2008). "Agile methods and user-centered design: how these two methodologies are being successfully integrated in industry". In: *AGILE '08 Proceedings of the Agile 2008*. Washington, DC, USA: IEEE Computer Society, pp. 63–72.

Garrett, Jesse James (2011). *The Elements of User Experience: User-Centered Design for the Web and Beyond*. Second Edition. Berkeley, CA, USA: New Riders.

Goodwin, Kim (2009). *Designing for the Digital Age: How to Create Human-Centered Products and Services*. Indianapolis, IN, USA: Wiley Publishing, Inc.

Gould, John D. and Clayton Lewis (1985). "Designing for usability: key principles and what designers think". In: *Communications of the ACM* 28.3, pp. 300–311.

Gulliksen, Jan et al. (2003). "Key principles for user-centred systems design". In: *Behaviour & Information Technology* 22.6, pp. 397–409.

Hall, Roger R. (2001). "Prototyping for usability of newtechnology". In: *International Journal of Human-Computer Studies* 55.4, pp. 485–501.

ISO 9241-11 (1998). *Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability*. Helsinki, Finland: Suomen Standardisoimisliitto.

ISO 9241-210 (2010). *Ergonomics of human-system interaction – Part 210: Human-centred design for interactive systems*. Helsinki, Finland: Suomen Standardisoimisliitto.

Jokela, Timo and Pekka Abrahamsson (2004). "Usability assessment of an extreme programming project: close co-operation with the customer does not equal to good usability". In: *Lecture Notes in Computer Science* 3009, pp. 393–407.

Jordan, Patrick W. (2002). *Designing Pleasurable Products*. London, UK: Taylor & Francis.

Kreitzberg, Dr. Charles B. and Ambrose Little (2009). "Agile UX development". In: *MSDN Magazine*. URL: http://msdn.microsoft.com/en-us/magazine/dd882523.aspx.

Krug, Steve (2006). *Don't Make Me Think! A Common Sense Approach to Web Usability*. Second Edition. Berkeley, CA, USA: New Riders.

Leffingwell, Dean (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*. Boston, MA, USA: Pearson Education, Inc.

Medlock, Michael C. et al. (2002). "Using the RITE method to improve products: a definition and a case study". In: *Usability Professionals Association, Orlando, FL, July, 2002*.

Meszaros, Gerard and Janice Aston (2006). "Adding usability testing to an agile project". In: *AGILE '06 Proceedings of the conference on AGILE 2006*. Washington, DC, USA: IEEE Computer Society, pp. 289–294.

Miller, Lynn (2005). "Case study of customer input for a successful product". In: *ADC '05 Proceedings of the Agile Development Conference*. Washington, DC, USA: IEEE Computer Society, pp. 225–234.

Moløkken, Kjetil and Magne Jørgensen (2003). "A review of surveys on software effort estimation". In: *ISESE '03 Proceedings of the 2003 International Symposium on Empirical Software Engineering*. Washington, DC, USA: IEEE Computer Society, pp. 223–230.

Nelson, Elden (2002). "Extreme programming vs. interaction design". In: *FTP Online*. October 3.

Nielsen, Jakob (1993). *Usability Engineering*. San Francisco, CA, USA: Morgan Kaufmann.

— (2003). "Return on investment for usability". In: *Jakob Nielsen's Alertbox*. URL: http://www.useit.com/alertbox/roi-first-study.html.

— (2007). "Should designers and developers do usability?" In: *Jakob Nielsen's Alertbox*. URL: http://www.useit.com/alertbox/own-usability.html.

— (2008a). "Agile development projects and usability". In: *Jakob Nielsen's Alertbox*. URL: http://www.useit.com/alertbox/agile-methods.html.

— (2008b). "Usability ROI declining, but still strong". In: *Jakob Nielsen's Alertbox*. URL: http://www.useit.com/alertbox/roi.html.

— (2009a). "Agile user experience projects". In: *Jakob Nielsen's Alertbox*. URL: http://www.useit.com/alertbox/agile-user-experience.html.

— (2009b). "Discount usability: 20 years". In: *Jakob Nielsen's Alertbox*. URL: http://www.useit.com/alertbox/discount-usability.html.

Nielsen, Jakob (2011). "Parallel & iterative design + competitive testing = high usability". In: *Jakob Nielsen's Alertbox*. URL: http://www.useit.com/alertbox/design-diversity-process.html.

Norman, Donald A. (2002). *The Design of Everyday Things*. New York, NY, USA: Basic Books.

Pichler, Roman (2010). *Agile Product Management with Scrum*. Boston, MA, USA: Pearson Education, Inc.

Rapal Oy (2012). *Our Company*. Accessed March 23, 2012. URL: http://www.rapal.fi/en/our-company/.

Rubin, Jeff and Dana Chisnell (2008). *Handbook of Usability Testing*. Second Edition. Indianapolis, IN, USA: Wiley Publishing, Inc.

Runeson, Per and Martin Höst (2009). "Guidelines for conducting and reporting case study research in software engineering". In: *Empirical Software Engineering* 14.2, pp. 131–164.

Schwaber, Ken (2004). *Agile Project Management with Scrum*. Redmond, WA, USA: Microsoft Press.

Schwaber, Ken and Mike Beedle (2001). *Agile Software Development with Scrum*. Prentice Hall.

Schwaber, Ken and Jeff Sutherland (2011). *The Scrum Guide*. URL: http://www.scrum.org/scrumguides/.

Sefelin, Reinhard, Manfred Tscheligi, and Verena Giller (2003). "Paper prototyping – what is it good for? A comparison of paper-and computer-based low-fidelity prototyping". In: *CHI '03 extended abstracts on Human factors in computing systems*. New York, NY, USA: ACM, pp. 778–779.

Shine Technologies (2003). *Agile Methodologies Survey Results*. Melbourne, Australia: Shine Technologies Pty. Ltd.

Silva, Tiago Silva da et al. (2011). "User-centered design and agile methods: a systematic review". In: *AGILE '11 Proceedings of the 2011 Agile Conference*. Washington, DC, USA: IEEE Computer Society, pp. 77–86.

Snyder, Carolyn (2003). *Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces*. San Francisco, CA, USA: Elsevier.

Sy, Desirée (2007). "Adapting usability investigations for agile user-centered design". In: *Journal of Usability Studies* 2.3, pp. 112–132.

Ungar, Jim and Jeff White (2008). "Agile user centered design: enter the design studio – a case study". In: *CHI '08 extended abstracts on Human factors in computing systems*. New York, NY, USA: ACM, pp. 2167–2178.

VersionOne (2010). *State of Agile Survey 2010*. Atlanta, GA, USA: VersionOne Inc.

Walker, Miriam, Leila Takayama, and James A. Landay (2002). "High-fidelity or low-fidelity, paper or computer? Choosing attributes when

testing web prototypes". In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 46.5, pp. 661–665.

Yin, Robert K. (2009). *Case Study Research: Design and Methods*. Fourth Edition. Thousand Oaks, CA, USA: SAGE Inc.

Appendix A

Current state analysis interview

This is the interview that was conducted as part of the current state analysis. The interview was conducted in Finnish with all participants. Below is a translation of the original Finnish interview. The participants were first read the introduction and then asked the interview questions.

A.1 Introduction

- This interview is part of my master's thesis on improving usability in agile software development
- Participation is voluntary
- Participation is anonymous, but in practice there is a risk that you may be recognized based on your work description or answers
- If you have any questions about the research now or at a later stage, you may ask me
- In this interview, usability is defined as a product quality attribute that describes how easy, efficient and satisfying the use of a product is for the end user
- This interview is part of the first stage of the research
- The objective of this stage is to examine the current state of usability at Rapal, and especially what problems are present in the current processes and methods

A.2 The interviewee

- What do you do for work at Rapal?
- What kind of work have you been doing lately?
- Can you give an example?

- How well do you think you understand the software development process at Rapal?
- How well do you think you understand usability?
- How about usability practices and methods?
- How important do you personally consider usability?

A.3 Current products

- How usable do you think the products of Rapal are?
- What strengths do the products have concerning usability?
- What weaknesses do the products have concerning usability?
- How do the usability of the products affect your own work?
- How has the usability of the products changed lately?
- How good knowledge do you think we have about the usability of the products?
- Do you think more or less effort should be put into usability?
- How would Rapal benefit from a greater investment in usability?
- What drawbacks could there be in a greater investment in usability?

A.4 The development process

- How would you describe the software development process at Rapal in general terms?
- How is the usability of products taken into account in the development process?
- What specific usability practices are used?
- How well do you think the development process at Rapal promotes usability?
- How do Scrum practices affect usability?
- How does the product owner role affect usability?
- What are the strengths in the software development process concerning usability?
- What are the weaknesses in the software development process concerning usability?
- Can you give examples of weaknesses or problems?
- What do you think is the cause of these problems?

A.5 The Scenario project

- How is usability taken into account in the Scenario project?
- How is this different from previous projects?
- What are the strengths of the project concerning usability?
- What are the weaknesses of the project concerning usability?
- Can you give examples of weaknesses or problems?
- What do you think is the cause of these problems?

A.6 The future

- How do you think the way usability is taken into account will change at Rapal in the coming years?
- How would you like it to change?
- If the two answers are different, what do you think is the reason?
- How would you measure progress in improving usability?
- What risks do you see in the future in improving usability practices?
- Do you have any personal goals or wishes regarding usability?

Appendix B

Action intervention interview

This is the interview that was conducted as part of the action intervention. The interview was conducted in Finnish with all participants. Below is a translation of the original Finnish interview. The participants were first read the introduction and then asked the interview questions.

B.1 Introduction

- This interview is part of my master's thesis on improving usability in agile software development
- Participation is voluntary
- Participation is anonymous, but in practice there is a risk that you may be recognized based on your work description or answers
- If you have any questions about the research now or at a later stage, you may ask me
- In this interview, usability is defined as a product quality attribute that describes how easy, efficient and satisfying the use of a product is for the end user
- The objective of this interview is to examine how well usability practices and methods have worked in the Scenario project, as well as what have been their strengths and weaknesses
- It is important that you answer as honestly as possible without worrying about offending me or criticizing what we have done

B.2 Usability activities in general

- In the Scenario project, how much effort has been put into usability?
- How much effort has been put into understanding users?

- How much effort has been put into user interface design?
- How much effort has been put into usability evaluation?
- How well have the usability practices worked as an integral part of the software development process?

B.3 The interaction designer role

- How do you think the interaction designer role has worked as part of the team?
- What do you think the most important tasks of the role has been?
- How has the role affected your own work?
- How much do you think that you have done work related to usability?
- How much do you think that other team members have done work related to usability?
- Do you think there has been more benefit or harm in having the designer as a team member?
- Do you think the team would have needed more or less usability know-how?
- Do you think the responsibility for usability has been too much or too little concentrated in one person?
- What do you think would have been the consequences if one of the developers also had assumed the interaction designer role?
- What do you think would have been the consequences if there had been no interaction designer at all?
- Do you have any other comments about the interaction designer role?

B.4 The product vision and user research

- During the project, how good has your understanding about the product vision been?
- How good has your understanding about the target users and their needs been?
- How good has your understanding about the usability goals been?
- From where have you primarily received information about the product vision and target users?
- How well did you understand the product vision and user needs when development started?
- How has this affected your work during the project?

- What do you think would have been the effects if, before development began, there would have been a one month sprint focusing on clarifying the product vision and understanding the target users?
- Do you have any other comments about the product vision and user research?

B.5 The use of paper prototypes

- How do you think paper prototypes have worked as a design tool during the project?
- How well do you think paper prototypes communicate the design?
- How well have paper prototypes worked in combination with user stories and acceptance tests?
- Do paper prototypes include sufficient information about the design from the point of view of development?
- Do paper prototypes help with estimating user stories?
- Have you created paper prototypes?
- If so, what has your opinion of it been?
- What problems are there with paper prototypes?
- Do you have any other comments about the use of paper prototypes?

B.6 Designing one sprint ahead

- What have been the advantages and disadvantages when designing in the previous sprint?
- What have been the advantages and disadvantages when not designing in the previous sprint?
- What do you think has been a suitable level of design in the previous sprint?
- How has design and updating user stories worked in combination?
- Do you have any other comments about designing one sprint ahead?

B.7 Usability testing of sprint results

- What have been the advantages and disadvantages of usability testing the working product?
- How well did you understand the test results by watching the recording of the tests?

- How useful was analyzing the tests together with the team?
- How did the tests affect your view of the usability of the product?
- How did the usability tests affect your own work?
- Do you have any other comments about usability testing?

B.8 Communication within the team

- How well do you think communication related to usability has worked within the team?
- How have the daily scrum meetings affected communication?
- What do you think were the biggest communication challenges?
- How well has the project wiki worked for communication related to usability?
- How well has the paper prototype wall worked for communication?
- Do you have any other comments about communication?

B.9 Other

- Do you have any other comments about usability in the Scenario project?

Appendix C

Observation framework

This is the framework that was used as a basis for the observation during the action intervention.

C.1 Usability activities in general

- How much resources were dedicated to usability activities?
- How well did the usability activities integrate into the development process?
- How did the different stakeholders react to usability activities?

C.2 The interaction designer role

- What were the advantages of a dedicated interaction designer role?
- What were the disadvantages of a dedicated interaction designer role?
- How did the other team members relate to the role?
- How were responsibilities divided?
- How much skill was required for the role?
- How much time was required for the role?

C.3 The product vision and user research

- How well did the team understand the product vision?
- How well did the team understand the target users?
- How well did the team understand the usability goals?
- Which methods were used for understanding users?

C.4 The use of paper prototypes

- What were the advantages of using paper prototypes?
- What were the disadvantages of using paper prototypes?
- How did different stakeholders respond to the prototypes?
- How useful were the prototypes for the product owner?
- How useful were the prototypes for developers?
- How did paper prototypes compare to high-fidelity prototypes?
- How did prototype testing work?

C.5 Designing one sprint ahead

- What were the advantages of designing one sprint ahead?
- What were the disadvantages of designing one sprint ahead?
- What was the appropriate level of prototyping before sprint planning?
- How did paper prototyping work in combination with updating the product backlog?

C.6 Usability testing of sprint results

- What were the advantages of testing the working product?
- What were the disadvantages of testing the working product?
- How did testing the working product differ from testing a paper prototype?
- How did watching a recording of the test with the team work?
- What was the team reaction to analyzing the tests together?
- How many of the usability problems found got fixed?
- How much effort did usability testing require?

C.7 Communication within the team

- What part of usability-related communication worked well?
- What were the problems with usability-related communication?
- How much face-to-face communication was necessary to communicate the design?