HELSINKI UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering

ANNA HÄKLI

INTRODUCING USER-CENTRED DESIGN IN A SMALL-SIZE SOFTWARE DEVELOPMENT ORGANIZATION

Master's Thesis
August 10, 2005

Supervisor: Professor Marko Nieminen, D.Sc. (Tech.)
Instructor: Janne Sirén, M.Sc.

HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Computer Science and Engineering
Information Networks

ABSTRACT OF MASTER'S THESIS

Author
Anna Häkli

Date
August 10, 2005

Pages
VI+107

Title of Thesis
Introducing User-Centred Design in a Small-Size Software Development Organization

Professorship
User Interfaces and Usability

Professorship Code
T-121

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

Instructor
Janne Sirén, M.Sc.

This study discusses the challenge of introduction and implementation of user-centred design (UCD) in a small software development company. A case study was conducted in a Finnish software company, Spellpoint Group, where usability was considered an important but too often neglected quality attribute of software. The purpose of this study was to introduce the principles of UCD to the company and to enhance software developers' knowledge and practical abilities in usability engineering.

At first, the current state of the company's work processes and UCD knowledge was analyzed. The analysis confirmed that software development in the company does not follow any specific process model and that usability and UCD are more known as concepts than integrated working methods. General attitude towards usability in the company was positive, however, and the need for more information was admitted. The upcoming training programme was regarded interesting and welcome.

The understanding gained through the analysis was combined with a review of related work in the field of usability research and UCD. Characteristics of the UCD process were examined and several usability engineering techniques were presented. Also, challenges in and obstacles to UCD integration in an organization were discussed. Based on these findings, a suggestion was made for the company about how to include usability in software development projects. The suggestions addressed both projects with limited resources and projects with more flexible settings. To share the UCD knowledge in the organization, a training programme was planned for the company personnel. The training was organized in two half-day workshops where the participants simulated one iteration cycle of a miniature software development project, using the usability engineering techniques presented in an introductory lecture.

The effect on participants' learning was measured after the training. The results showed that the training enhanced the developers' understanding of the need to consider usability from the beginning of any project and increased their skills in designing and evaluating user interfaces. The training also raised the team spirit among the employees and helped in forming a common vocabulary of the topic. To better support UCD integration in upcoming software projects, management initiative and adequate resources are needed.

Keywords
User-centred design, usability engineering process, lightweight usability methods, usability training, process integration challenges, software development

TEKNILLINEN KORKEAKOULU Tietotekniikan osasto Informaatioverkostot	DIPLOMITYÖN TIIVISTELMÄ

Tekijä	**Päiväys**
Anna Häkli	10.8.2005
	Sivumäärä
	VI+107

Työn nimi

Käyttäjäkeskeisen suunnittelun edistäminen pienessä ohjelmistokehitysorganisaatiossa

Professuuri	**Koodi**
Käyttöliittymät ja käytettävyys	T-121

Työn valvoja

Professori Marko Nieminen, TkT

Työn ohjaaja

Janne Sirén, FM

Tämä tutkimus käsittelee käyttäjäkeskeisen suunnittelun käyttöönoton haasteita pienessä ohjelmistoyrityksessä. Tutkimuskohteena oli suomalainen ohjelmistoyritys Spellpoint Group, jossa käytettävyyttä pidettiin tärkeänä mutta usein laiminlyötynä ohjelmistojen laatuun vaikuttavana tekijänä. Tämän tutkimuksen tavoitteena oli esitellä käyttäjäkeskeisen suunnittelun periaatteet yrityksessä ja parantaa ohjelmistokehittäjien asiantuntemusta ja käytännön taitoja käytettävyystyössä.

Tutkimuksen aluksi kartoitettiin yrityksen työmenetelmien ja käytettävyysosaamisen nykytila. Analyysi vahvisti, ettei yrityksessä ole vakiintunutta ohjelmistokehitysprosessia ja että käytettävyys ja käyttäjäkeskeinen suunnittelu ovat tutumpia käsitteinä kuin käytännön työmenetelminä. Yleinen asenne käytettävyyttä kohtaan oli kuitenkin myönteinen ja lisätiedon tarve tunnustettiin. Tuleva koulutusprojekti otettiin yrityksessä kiinnostuneena vastaan.

Nykytilan analyysin tueksi nostettiin tuloksia aiemmista tutkimuksista käytettävyyden ja käyttäjäkeskeisen suunnittelun alalta. Kirjallisuuskatsaus käsitteli käyttäjäkeskeisen suunnitteluprosessin piirteitä ja esitteli useita käytettävyystyön menetelmiä. Myös käyttäjäkeskeisen suunnittelun käyttöönoton haasteita ja ongelmia eriteltiin. Näihin tutkimuksiin nojautuen yritykselle esitettiin ehdotus käytettävyyden sisällyttämisestä ohjelmistokehitysprojekteihin. Ehdotus huomioi sekä resursseiltaan rajalliset projektit että sellaiset projektit, joissa liikkumavaraa on enemmän. Käytettävyysosaamisen lisäämiseksi yrityksen henkilöstölle järjestettiin kaksipäiväinen koulutus, jonka työpajoissa osallistujat simuloivat yhden suunnittelukierroksen pienimuotoisesta ohjelmistokehitysprojektista, käyttäen aloitusluennolla esiteltyjä käytettävyysmenetelmiä.

Osallistujien oppimista mitattiin koulutuksen jälkeen. Tulokset osoittivat, että koulutuksen myötä ohjelmistokehittäjät ymmärsivät paremmin kuinka tärkeää on huomioida käytettävyys aina projektin alusta saakka. Koulutus myös paransi ohjelmistokehittäjien taitoja suunnitella ja arvioida käyttöliittymiä. Koulutus paransi yhteishenkeä työntekijöiden kesken ja auttoi muodostamaan yhteisen käsitteistön aihepiiristä. Jotta käytettävyys voitaisiin paremmin huomioida tulevissa projekteissa, on yrityksen johdon tehtävä aloite ja huolehdittava riittävistä resursseista käyttäjäkeskeiseen suunnitteluun.

Avainsanat

Käyttäjäkeskeinen suunnittelu, käytettävyysprosessi, kevyet käytettävyysmenetelmät, käytettävyyskoulutus, prosessien yhteensovittamisen haasteet, ohjelmistokehitys

FOREWORD

Before I started this study I still did not really acknowledge how much work other people around the world have done in the field of usability engineering. In only two decades an enormous number of articles, books, and design guides have been published, and I have only been able to glance through a small fraction of them. This study has opened my eyes to the wide range of studies and interesting research that has been done before and is constantly going on.

This thesis is the result of one spring that I spent thinking about the chances and challenges of user-centred design in software development. I was lucky to work for a company that was willing to take a chance in this project. I encountered enthusiasm and openness to my ideas and thoughts, and I still want to believe that usability can make its way to real business life outside universities and international conferences. There are so many of us that constantly suffer from bad design in computer programs and any technical devices that it is impossible to think this could go on forever. Yet, the task is not simple and there is still a lot to do. Small steps and lots of devotion – that is the key to better results.

Many special people have contributed to this work. First, I want to thank Professor Marko Nieminen for supervision and interest in my research topic. The valuable comments that I received helped me in forming a coherent structure for this thesis and in focusing on the most significant parts of the study. My instructor Janne Sirén has proficiently taken care of all practical arrangements at Spellpoint Group and suggested several relevant adjustments to my text. I also want to thank all my work colleagues for their participation in this study. A special thanks to Tuomas for a good sense of humour and for the practical tips about the art of writing a Master's Thesis. I also thank Elina Jormanainen for commenting an early draft of my work.

I also want to address all the great people at the HUT that have helped me in my usability studies. Course assistants Laura Turkki and Mika P. Nieminen taught me the hard discipline of doing research and writing reports – I appreciate your effort. I thank Sirpa Riihiaho for inspiration and motivation in usability. I also send all my best to my fellow eXtreme Usability students – thanks for sharing the experience and good luck and success to all of you in your future work! Johanna, Laura, and Kaisa: you have been my soul mates and shared the distress of this work; I could not thank you enough.

I am also grateful to my family for their love and encouragement. Mom and Dad – you knew I would make it this far. Finally and most of all, I want to thank my loving fiancé Antti for patience, mental support, technical backup, and a thorough proof-reading of my final text. You share my drive for best results and yet help me keep my feet on the ground. Thank you for being there for me.

Helsinki, August 10, 2005 Anna Häkli

CONTENTS

TERMINOLOGY AND ABBREVIATIONS

Here are listed and explained the terms and abbreviations used in this thesis.

Heuristic evaluation	A usability engineering technique where user interface is evaluated against known design principles, heuristics. The goal of the evaluation is to find usability problems in the interface.
Prototype	A test version or a demo of a program. Prototypes can be used for testing and evaluation of a user interface and usability of a program.
Software	A generic term for programs running on a computer. Software consists of instructions given in a specific programming language for a computer to perform certain operations.
Software developer, software engineer	A professional with usually technical background and/or education who is responsible for the technical design and implementation of software products or systems. Both terms are used interchangeably.
Task	The term task is used to refer to user's actions. A task is a set of discrete actions taken to reach certain goals.
Usability	A quality attribute of a computer program or other system. A usable program is productive, efficient, and pleasing to use and it supports the user in achieving their goals.
Usability engineer, usability specialist, UCD specialist	A professional, regardless of his or her education or job description, who is experienced in usability engineering and is responsible for good user interface design, practicing of usability methods, and the implementation of user-centred design in the organization. All three terms are used interchangeably.
Usability engineering	The process where usability is systematically built into a product.
Usability testing	A usability engineering technique where user interface is evaluated with a test user. The user performs given tasks with the system, and problems found in interaction are recorded and later analyzed. Usability testing is one of the core methods of usability engineering.

User	A person who uses a program or interacts with a computer system. Users are always natural human beings with personal differences, goals, and needs that should be considered in software development. Synonym to the term *end user* also found in literature.
User-centred design	User-centred design refers to a systematic approach to usability development where users and their needs are acknowledged in all phases of the product design process. User-centred design is an iterative process that aims at the best possible usability experience for a certain product with employment of certain usability engineering methods.
INUSE	A European Union research project for assuring the usability of interactive systems or web sites.
ISO	International Organization for Standardization
KESSU	A research project aiming at implementing user-centred design in software development organizations.
OOSE	Object-oriented software engineering; a recent approach to software development that has some similarities to usability engineering methodology.
UCD	User-centred design
UE	Usability engineering
UI	User interface
UML	Unified Modeling Language. UML is a standard notation for documenting object-oriented system specifications.

1 INTRODUCTION

1.1 Background

For some time now, usability has tried to make its way to the work processes and business culture of technology companies. In the competitive business environment of the 2000s, customer acceptance and satisfaction are becoming an essential factor in product marketing. An increasing number of customer companies base their purchase decision at least partly on the ease of use of a product (Rohn, 2005). It is no longer enough to deliver software than functions if it has poor usability.

Although benefits of usable software have been widely recognized, user-centred design (UCD) is rarely implemented thoroughly (Hakiel, 1999). Probably most clearly this can be seen in small companies where the work tasks are not separated: the job descriptions of software developers and system designers are not clearly differentiated, and they companies rarely have usability engineering (UE) professionals specialized in user-centred system development. Instead, software engineers typically both design and implement the software. Without a clearly stated goal for user-centeredness and adequate knowledge in usability engineering techniques, small software companies easily fall short in their attempts for more usable products.

Establishing a separate usability engineering division in a company may require a large investment in equipment and personnel, and can be seen available only for large companies. A typical software company is still relatively small, and it is natural to pose a question: Is it possible for small software companies to engage in usability engineering and to pay attention to usability in their development process? Moreover, can it be done without excess costs while being feasible also in those companies where no specialized UE personnel is available?

At Spellpoint Group, a Finnish software service company based in Espoo and Heinola, usability is seen as an important factor when designing software. Employing 15 people at the time of this study, the company was interested in improving its UCD skills in software development. Although some effort had been made before, there was a feeling at Spellpoint that usability had not managed to make its way to every-day work as well as it should have. Software projects are usually carried out with tight schedules and by the terms of the customer. Usability is considered difficult to sell. Under limited resources in time and without practical experience in usability engineering, usability too often has to give way to functional requirements. During spring 2005, a training project was organized at the company for introducing user-centred design practices for the company's personnel.

1.2 Why Usability?

Writings about the reasons for practising usability engineering have been published extensively. Productivity, efficiency, and safety are recognized criteria for a system, but moreover, satisfaction and enjoyability are acknowledged as critical factors especially in customer products for leisure. Usability has a strong ground on business environments as well, and is closely related to the wider concept of work ergonomics. As an example, a study of bank clerks showed that a program with bad usability had a

major effect on the well-being and motivation of employees (Sinkkonen, Kuoppala, Parkkinen, & Vastamäki, 2002).

Investing in usability will produce many substantial benefits, both for the buyer and the vendor. From the customer's point of view, good usability increases user productivity and decreases user errors and training needs (Marcus, 2005*)*. Usable products also facilitate learning and acquisition of new systems, and increases user satisfaction (ISO 13407, 1999). For the vendor, the benefits include increased sales through customer satisfaction, decreased customer support costs, increased savings from making changes earlier in the design lifecycle, and reduced cost of providing training (Marcus, 2005). A few more facts can be added: When information is gathered about the users to back up decision-making, the system requirements are based on research results instead of opinions (Beyer & Holtzblatt, 1998). Well-thought system requirements also shorten the time needed to agree about the user interface and system features with the customer (Sinkkonen et al, 2002). Good usability also helps in creating long relationships with customers, and serves as a sales argument in project negotiations (Alanne, 2002).

User interface is the only part of the program that is visible to the user and as such it is essentially important when considering the overall user experience and satisfaction. As much as 50 to 80 percent of all software code may concern user interface (Faulkner & Culwin, 2000), and still it is usually the most overlooked part of a system. Although functioning perfectly from a technical point of view, a program does not meet its requirements if it does not do what the user wants it to. Thus, it is vital for the program to match the users' expectations and meet their needs.

Researchers have also reported large economical and time savings for both users and developers of well-designed software (see e.g. Bias & Mayhew (Eds.), 2005; Nielsen, 1993). As cited in Marcus (2005), "'One [well-known] study found that 80 percent of software life-cycle costs occur during the maintenance phase. Most maintenance costs are associated with "unmet or unforeseen" user requirements and other usability problems' (Pressman, 1992)". By exploring the users and their needs more in detail at the very beginning of a software process, many unexpressed needs may be detected and action can be taken to include them in the design.

As another example, an estimate has been presented that 60 percent of product usability is related to user interaction with the system, and 40 percent to system look and feel (Ames, 2001). Problems with look and feel can be fixed at late stages in the design process, but issues with interaction are typically costly and difficult to correct (Ames, 2001). The earlier the problems are detected, the easier it is to correct them. Resources in development are saved when the same work is not done twice (Sinkkonen et al, 2002).

It is easy to make changes in the beginning of a project when few resources are bound and the plan is still flexible and modifiable. The later new problems occur and the initial plan is modified, the more difficult and expensive it becomes. Figure 1, adopted from Erlich and Rohn (1994; as cited in Marcus, 2005), illustrates the relationship of design alternatives to costs. The cost of making changes to the design increases exponentially as the development progresses.

An often-cited usability cost–benefit ratio of 1:10–100 was presented by Gilb (1988), stating that correcting a problem when a system is in development will cost ten times as much as fixing it in the design phase. Moreover, after the system has been released,

fixing the same problem will cost 100 times as much as fixing it in the design. (As cited in Marcus, 2005; pp. 19)

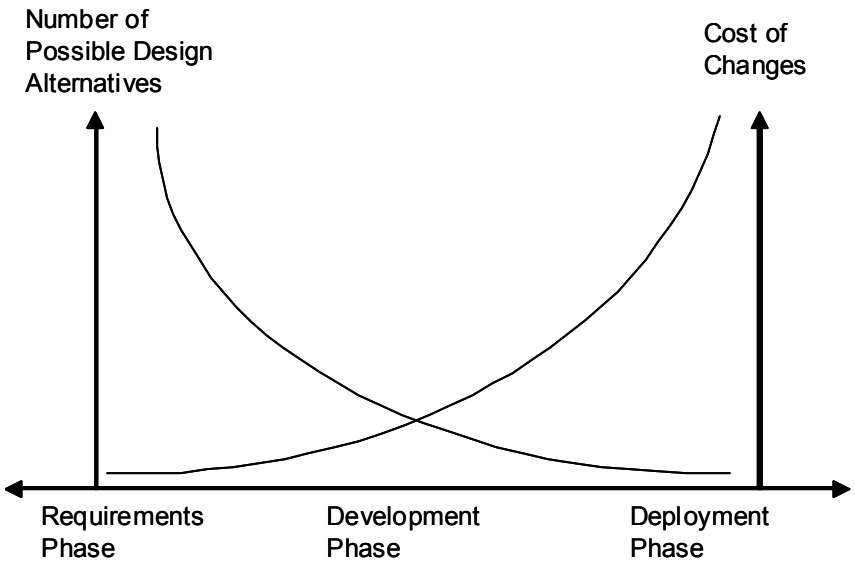

Figure 1. The number of possible designs decreases as the costs to make changes increases (Erlich & Rohn, 1994; as cited in Marcus, 2005; pp. 23).

1.3 Objectives of the Study

The main objective of this work was to establish a common user-centred method infrastructure at Spellpoint Group and to enhance software developers' UCD abilities in all phases of software development. Knowing the lack of a specified software process in the company, discussion about an appropriate user-centred process was also needed. A practical goal was to arrange a usability training programme for the company's personnel. There was a wish from the company's part that the upcoming training would take into account how usability has been introduced earlier in the company and build the new contents on that base.

To be able to decide which actions to take in the upcoming training, the current state of UCD knowledge was first analyzed. Targets for the actual usability development project were set later after the current state analysis.

The main objectives for the current state analysis were formulated as follows: What kind of training does the company need? Which aspects of software development should be addressed in the training?

To answer these questions, a set of supporting questions were listed:

1) What is the present knowledge base of usability and user-centred design methods among the company's personnel?
2) What is the present way to develop software at the company?
3) How does user-centeredness show up in every-day work?
4) What are the main problems in the current way of working?

By answering these questions, an answer to the main objectives would be found. Forming an understanding of the current state of the company was considered important to fully understand the needs of the organization as a whole and to be able to address the most critical phases in software development. Based on the overall picture of the company, the training contents were selected.

Because used-centred design approach was a new concept to the overall organization, the first actions towards usability recognition did not primarily target business strategy or marketing sectors. The main goal was to enhance software developers' knowledge and practical abilities to take end user needs into account from the very beginning of every software project. All the company's personnel participated in all phases of the training project: it was seen important to share the same information with everybody in the small organization where tasks and job descriptions are not fully separated. Yet, neither the integration of user-centeredness into the marketing material nor company's strategy setting or market positioning was addressed during this project.

1.4 Structure of This Thesis

So far, Chapter 1 has introduced the background and the motivation for this study, and discussed some reasons for usability engineering. Also the goals of this study were presented.

Chapter 2 gives an overview of user-centred software development. Some focal concepts of the research area are discussed in short. After the concepts, user-centred design is examined from a process point of view.

Chapter 3 describes the current state research of the company. In this chapter, the target company is thoroughly examined and background information needed for the training programme is discussed. At the end of the chapter, the goals for the rest of the study are formulated.

Chapters 4 and 5 present the related work and previous studies in the field of usability and user-centred design. Based on the current state analysis, relevant usability engineering techniques are presented, focusing on lightweight methods that best apply to a small software development organization's use. Chapter 5 reports some previous experiences in training UCD to software engineers, reflects on learning in organizations, and discusses the challenges in integrating UCD into work processes.

Chapter 6 returns to the case study and presents suggestions for employing UCD in the company's work practices, based on the literature review and the current state analysis. A process model is presented for the case company, and a lightweight methodology is suggested to be used in projects where resources do not allow a full-scale usability process.

Chapter 7 describes the arrangements and experiences from the usability training that was organized. At the end of the chapter, the learning impact of the training is reflected and discussion of the training arrangements is presented. Finally, Chapter 8 summarizes the results of the study and presents the final conclusions. The results are reviewed and the overall success of the study is considered. The conclusions also address the company future and propose some further actions for the company.

2 OVERVIEW OF USER-CENTRED SOFTWARE DEVELOPMENT

2.1 Concepts

To better grasp the field of study and help the reader to follow the writing, some basic concepts are presented. In the following sections, the concepts of usability, user interface, and user-centred design are discussed. Moreover, to link between user-centred design and our case study in a software development organization, the nature of software development and software business is reflected.

2.1.1 Usability and Quality in Use

Usability has several different interpretations and definitions, depending on the context where it is discussed. In common language, usability and ease of use can be regarded the same thing. In this thesis, usability is defined as a quality attribute of software: a usable program is productive, efficient, and pleasing to use and it supports the user in achieving their goals. This definition is based on the international standard ISO 9241, *Ergonomic requirements for office work with visual display terminals*, that defines usability 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" (ISO 9241-11, 1998). The clause "specified context of use" implies that usability must always be viewed in context: it is always dependent of the user's intentions and goals, and thus any universal statement of a system's usability cannot be given.

A more recent standard ISO 9126 *Software product quality* specifies usability as "the capability of the software product to be understood, learned, used and attractive to the user, when used under specified conditions" (ISO/IEC 9126-1, 2001). Usability is seen as one of six attributes that affect product quality, together with functionality, reliability, efficiency, maintainability, and portability. A broad concept of *quality in use* is introduced to cover "the capability of the software product to enable specified users to achieve specified goals with effectiveness, productivity, safety and satisfaction in specified contexts of use" (ISO/IEC 9126-1, 2001); this definition is nearly similar to the one of usability in ISO 9241-11, only a safety aspect has been added later. According to this definition, usability with its definition is one part of the overall goal that the system meets the user's needs (Bevan, 1999). Figure 2 illustrates the definition of usability as a part of software quality. In order to achieve quality in use, the software must meet external and internal quality requirements that influence the quality in use experience.

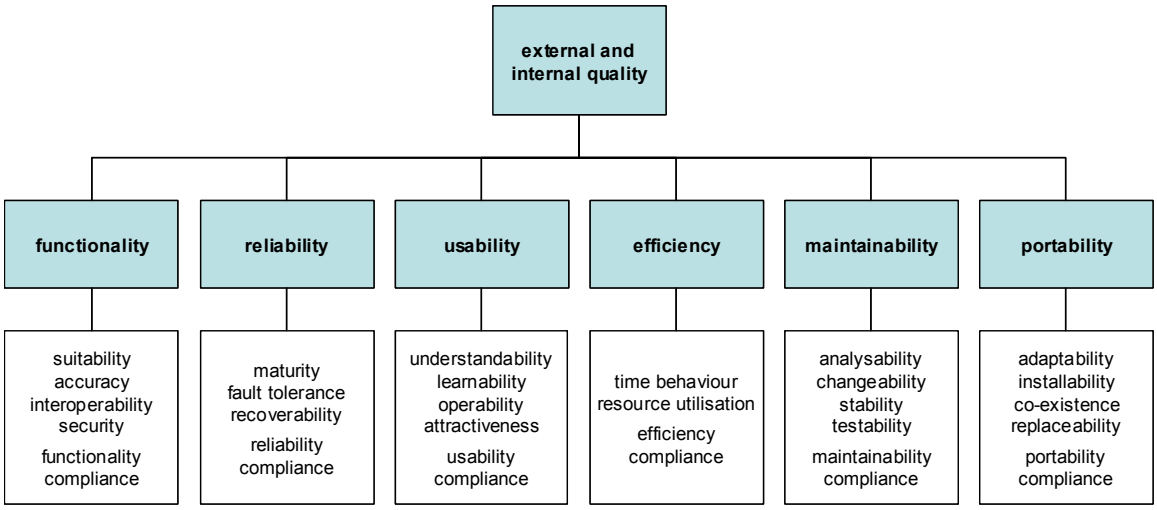

Figure 2. Usability as part of software quality (ISO/IEC 9126-1, 2001).

Apart from standards, other definitions for usability have also been presented. Gould and Lewis (1983) named four attributes for good usability: "(a) easy to learn; (b) useful, i.e., contain functions people really need in their work; (c) easy to use; and (d) pleasant to use" (Gould & Lewis, 1983; p. 50). The same attributes repeated ten years later, when Nielsen (1993) presented a hierarchical model of usability in relation to other system quality attributes. Usability, according to Nielsen (1993) can be described with five attributes: easy to learn, efficient to use, easy to remember, few errors and subjectively pleasing. Nielsen sees usability as a part of bigger concept called *system acceptability*. Apart from being socially acceptable, i.e. complying with the norms and standards of the society, the system must also gain practical acceptance. Practical acceptability, according to Nielsen, is composed of cost, compatibility, reliability and usefulness, which in turn can be broken up to utility and *usability*. Utility refers to the system's ability to do what is needed, and usability in turn questions how well users can use the system (Nielsen, 1993). Figure 3 below illustrates the hierarchy.

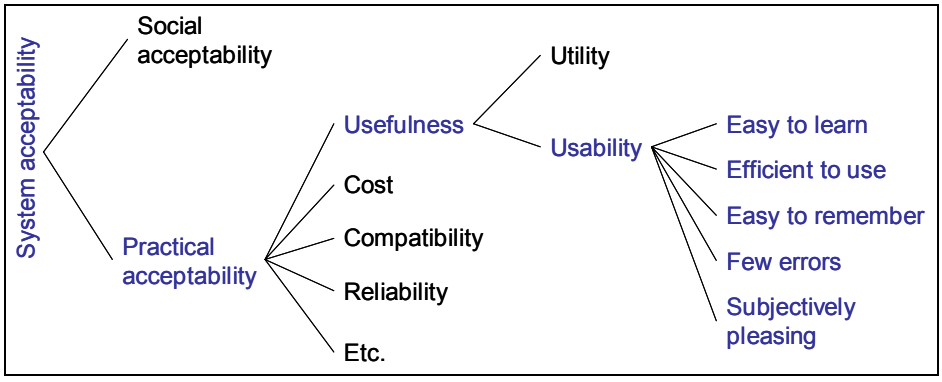

Figure 3. Usability as part of system acceptability. (Nielsen, 1993; colouring added)

2.1.2 User Interface

Generally, a *user interface* is any part of a system that a user comes into contact; either physically or perceptually. In a computer program a user interface consists of the layout of the information contents and functions available for the user. Graphical user interfaces (GUIs) are nowadays the most typical user interfaces, in contrast to text-based command-line user interfaces. Typical means of interaction with the user interface are keyboard, mouse, or touch screen, among others.

User interface design means the planning and implementation of the layout and user interaction with the interface. Sometimes the term *interaction design* is used to refer to the same activity. But, most importantly, user interface design is about designing the way to work with a system (Beyer & Holtzblatt, 1998). The structure of components and the actions available in the user interface dictate what a user can and cannot do, and how one must proceed in order to get a task accomplished. Thus, designing a user interface is about designing computer-supported work. A good design considers human factors and user needs for the system, and corresponds to users' natural ways of doing things.

2.1.3 User-Centred Design

The main philosophy behind *user-centred design* is the understanding of humans as users of computers and the need to facilitate this interaction to better support both work and leisure activities. UCD is more than a set of usability techniques or certain methods; it is an approach that tries to include usability engineering in all phases of system design. According to the international standard for UCD, ISO 13407, user-centred design is "a multi-disciplinary activity, which incorporates human factors and ergonomics knowledge and techniques with the objective of enhancing effectiveness and productivity, improving human working conditions, and counteracting the possible adverse effects of use on human health, safety and performance" (ISO 13407; as cited in UsabilityNet, 2003). Also the terms Human–Computer Interaction (HCI) and Computer–Human Interaction (CHI) are often used to refer to the wide range of human-centred activities in enhancing the interaction between man and computer.

2.1.4 Software Development and Software Business

Software is a generic term for programs running on a computer. *Software development* is the activity of designing and producing software products and computer systems. Nowadays, software is encountered in a majority of all kinds of technical devices, spanning from personal computers and web pages on the Internet to different kinds of mobile devices, home electronics, and cars. In this case study, the discussion will be restricted to the development of computer programs and web pages. The terms *software engineer* and *software developer* are used interchangeably to denote a person who develops software for work.

The case company of this study operates in the business of software services, which means offering other companies consultancy and various custom-made software development services. The business is very intense in nature and the development times are continuously getting shorter. As an example, one case study of systems development projects in 2000, reported by Maguire (2000), showed that the average length of a development project had diminished from 10–12 months to only 3 or less months. At

the same time, however, the complexity of systems and the number of lines of code is increasing (Maguire, 2000).

Pursue for cost-effectiveness, time savings, and increased productivity can affect the development process. Additional costs are cut and only the necessary actions are taken. Often, the customer has the last word when making decisions about resources and budget for the project. Small businesses must often work in the terms and needs of larger customers.

Fitting usability engineering into software development can seem a difficult task for small businesses with few resources. Large-scale improvements are not possible in a short time span, and the size of the company defines the economics of organizational development projects. Moreover, it is not possible to create a software process model that ignores the realities of every-day business. There is still a clear need to find a way for small and medium-sized businesses to incorporate usability engineering in the software design activities. The rest of this chapter will concentrate on forming an understanding of the practice of user-centred design. Later on, this thesis will address the topic of usability engineering from the perspective of small businesses, and tries to find solutions for practising usability engineering with fewer resources.

2.2 The Essentials of UCD

As introduced in the section 2.1.3, user-centred design is a multi-disciplinary approach to product development that aims at supporting user needs with the incorporation of wide variety of usability engineering techniques. Many software developers may have taken a course on basic concepts of user interface design and learned a few usability techniques to support the design work, but "few of them have an understanding of the complete UCD toolbox at a level that allows them to incorporate it into the whole software development lifecycle" (Seffah & Andreevskaia, 2003; p. 653). Because usability engineering is still a young profession, a common consensus has not yet been found about what the core skills for a UCD practitioner really are (Wilson & Rosenbaum, 2005). A number of writers have yet tried to define the essentials of UCD that every practitioner should know. The results of such studies vary from long to very long lists of different kinds of attributes.

Dayton et al (1993) presented an extensive list of universal attributes that would be required from a UCD specialist. The attributes were organized in three groups – knowledge, skills, and attributes that are harder to acquire – and they included among others knowledge in psychology and human cognitive processes, various usability engineering techniques and HCI standards, understanding of software development processes and implementation, skills of negotiation and team work, and personal attributes such as tenacity, flexibility, and empathy. Fully conforming to those requirements would certainly qualify for a UCD specialist, but it can be asked whether less would be enough for a software developer. Indeed, Karat and Dayton (1995) have later argued that formal knowledge of behavioural science or cognitive processes would not be highly necessary for successful application of usability engineering, unless designing systems in extreme conditions where human senses and perception are strained to the limits. Instead, they state, practical experience weighs most.

Based on Dayton et al's work among other sources, Seffah and Adreevskaia (2003) presented a skill map to be used in training software developers about usability – resulting in an equally extensive list of critical skills in UCD. Concentrating on software developers' point of view, however, this curriculum emphasizes a little more the various technical skills and the ability to communicate and work with multi-disciplinary teams along with usability engineers.

Wilson and Rosenbaum (2005) point out three general skills for a UCD practitioner. First, they mention the ability to consider the business goals of all stakeholders of a product. Second, a UCD practitioner should be able to interview and discuss with people effectively to gain valuable information. Third, one should continuously self-assess one's own skills and knowledge.

Judging from the arguments and conclusions of these studies, it is rather difficult to extract a small set of strictly defined factors or skills. It seems that the understanding of the big picture is most essential, rather than an individual skill or piece of knowledge. The process view of user-centred design repeats in the discussions, and thus looking closer at the process of UCD is justifiable.

2.3 User-Centred Design Process

Usability engineering is nowadays commonly seen in a wider perspective consisting of more than just separate techniques for usability testing or validation. The emphasis has shifted to consider the development process as a whole: user-centred design is often referred as an overall development process where users are taken into account from the beginning of the design process. Early focus in users in not a new idea, however: the need to include users in the process has been discussed in various writings since the mid-1980s (see e.g. Gould & Lewis, 1983 and 1985; Nielsen, 1993, Hackos & Redish, 1998; Mayhew, 1999b).

When talking about usability, it is important to understand the conditions in which the final product is going to be used. As Nielsen (1993) states, there are not necessarily only one right or wrong design solution; each design needs to be adjusted to the specific circumstances that may vary between projects. It is thus difficult to say exactly what makes an interface good, and detailed guidelines that would result in a good interface in all possible cases cannot be given in such varying project conditions. Instead, the usability engineering activities needed to arrive at good results are quite similar, and the process is well established (Nielsen, 1993). Adopting a user-centred design process, an organization will achieve a suitable design that respects the specific user needs.

The process of user-centred design, or usability engineering, has a cyclical form illustrated in Figure 4. This cycle is found in many other engineering fields, too, and is sometimes called iterative prototyping or spiral design in software development (Butler, 1996). It consists of four phases: analyze, design, build and evaluate. The cycle continues until the evaluation yields satisfactory results (Butler, 1996).

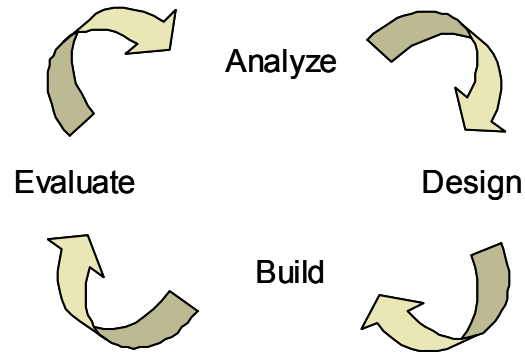

Figure 4. Iterative Cycle of Usability Engineering. (Butler, 1996)

The cyclical model presented above is repeated by a number of authors (e.g. Gould & Lewis, 1985; Nielsen, 1993; Mayhew, 1999b), others emphasizing some stages more than the others (Butler, 1996). Nevertheless, there is a common understanding that usability engineering is about iteration until the requirements are met. Unlike the conventional Waterfall model that is often discussed in software process literature (initially presented by Royce, 1970), user-centred design cannot possibly be seen as a straightforward process that gets it right the first time.

In the following sections, user-centred design process is further discussed and a couple of variations of the generic process model are presented.

2.3.1 Four Principles for Usability

Gould and Lewis (1983; 1985) were early to present general principles that characterize user-centred design process. The principles were initially written in 1983 and were then refined in later works. According to their most recent work (Gould, Boies, & Lewis, 1991; p. 75), a UCD process consists of four activities:

Early Focus on Users: Designers should have direct contact with intended or actual users. Methods for collecting data about users are, for example, interviews, observations, and surveys. The goal is to *understand* – not identify, describe, or stereotype (Gould & Lewis, 1983) – their characteristics, including behaviour, attitudes, and anthropometrics.

Integrated Design. All aspects of usability affecting the total user-experience – user interface, help, training, and documentation – should be designed in parallel, and their management should be centralized.

Early – and Continual – User Testing. Successful design is dependent on empirical testing, including observing user behaviour, gathering and analyzing feedback, and committing strongly to design improvements and problem solving.

Iterative Design. The results from testing must affect the design process. An iterative process of implementation, testing, feedback and evaluation must be repeated to improve the system.

The authors state that following the procedure has proved to lead to useful, usable, and likeable systems (Gould, Boies, & Lewis, 1991).

2.3.2 International Standard for User-Centred Design

User-centred design process was documented in an international standard in 1999 (ISO 13407). This standard, Human-Centred Design Processes for Interactive Systems, lists four user-centred design activities that should start at the earliest stages of a project. Figure 5 illustrates the iterative sequence in which these activities are employed.

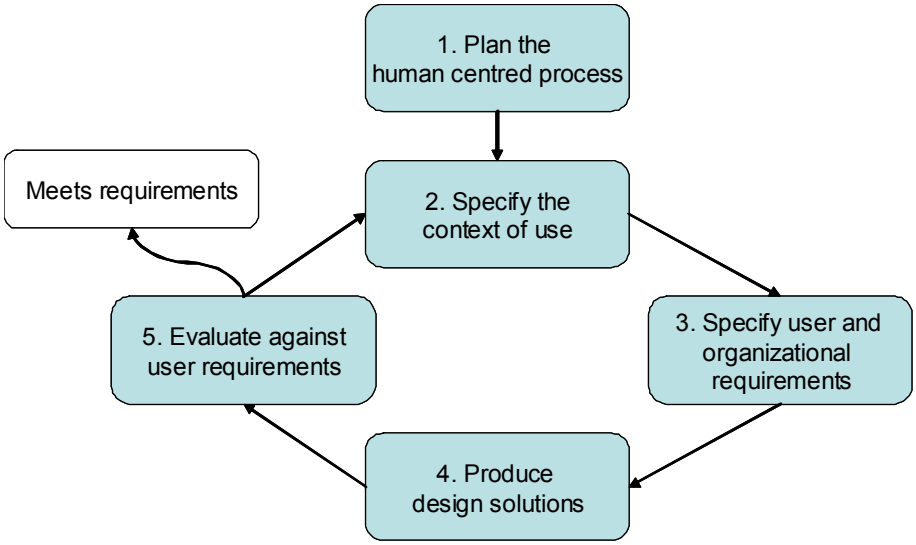

Figure 5. User-centred design process. (ISO 13407)

The process starts with the recognition of a need for and planning of a user-centred process. From this stage the process is iterated through four activities that are:

- Understand and specify the context of use
- Specify the user and organizational requirements
- Produce design solutions
- Evaluate designs against requirements

At the end of each iteration cycle the design is contrasted with the requirements, and the iteration is repeated until the objectives are met and the requirements satisfied.

ISO 13407 does not dictate the methods or practices that should be used in each stage; the organizations are free to choose methods that are applicable for their purposes. The standard serves well as a generic model for the process, leaving enough space for organizations to successfully apply it in their varying project conditions.

2.3.3 KESSU Project Model

Timo Jokela (2001) modified the ISO 13407 standard and proposed another UCD model with small improvements. Initially the model was named KESSU project model according to the research project; in an article one year later (Jokela, 2002) this name was no longer used. The process model is illustrated in Figure 6.

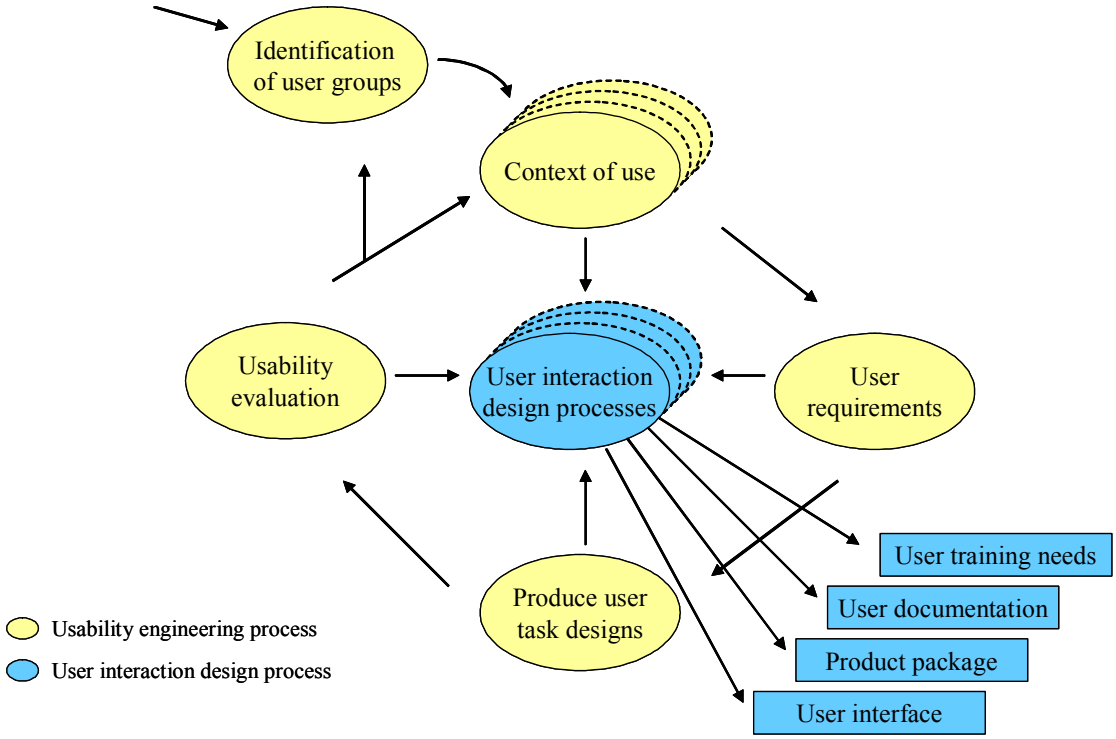

Figure 6. UCD Process Model (Jokela, 2002)

Jokela's model is defined through outputs that should be produced at each phase. Like the ISO 13407 standard, the model does not state either which techniques should be used to arrive at the outputs. In this model, a UCD process starts with identification of the intended users. For each user, the context of use is specified by defining user characteristics, their tasks, and the technical, organizational and physical environment in which they will use the system. If the environment of use or the tasks performed with the system that is currently in used differs from the new system to be produced, the descriptions should cover both of them. User requirements state the required performance of the product against the specified context of use, and define possible guidelines or restrictions that should be considered when designing an interface. The process of user task design produces descriptions how the users will carry out tasks with the new product. The actual interaction with the product is designed in the user interaction design process, and it covers the total user experience including the user interface, user documentation, training, and product package. The product design is then evaluated against the user requirements and by collecting qualitative feedback of the system performance. (Jokela, 2002)

When comparing this model with the ISO 13407 standard, Jokela's model introduces six processes instead of four. The main differences between the two models are the separation of user group identification and context of use phases, and the division of the UI design process in two. In contrast to the ISO 13407 model, here the context of use phase has multiple instances – one for each user. The separation of user group identification was done to emphasize the aspect that the context of use can vary between different users (Jokela, 2002). The separation of the UI design and user task design activities was done to emphasize the finding that the development of UI design solutions actually consists of two different processes: usability engineering process and UI design process. The visual interface and software is created by the designers in the

centre of the model, but the process of designing user's work flow is a UE process by nature. As Jokela (2002) states, the UI design process takes place every time a user interface or user documentation is produced; the usability engineering processes are there only to support that activity. In a company with little or no tradition in UCD, the user interface design process exists but the usability engineering processes are missing (Jokela, 2002). To make the development process user-centred, the usability engineering processes must have an effect on the final product, that is, they must be integrated to the work traditions of the designers.

The KESSU model was initially produced for a UCD process assessment tool. The model concentrates on measuring the extent to which user-centred design is practised in a software development process, and assessing the quality of the usability engineering output in each phase. Also, the integration of the UE outputs to the development is examined. The method will produce a qualitative assessment for improvement action.

The basic idea in the assessment is to concentrate on the outputs of the process. The individual techniques used to produce the output are not of interest here as long as they produce appropriate results. The UCD process outcomes by process phase (Jokela, 2002) are listed in Table 1. The more extensively the output is produced, the better quality it is and the more its results are integrated into the system design, the better the assessment.

Table 1. UCD process outcomes (Jokela, 2002)

Project phase	Output
Identification of user groups	User group definitions
Context of use	User characteristics User accomplishments with old and new system Descriptions of user tasks with old and new system (if different) Operational environment (old and new)
User requirements	Usability requirements User interaction design requirements
User task design	User task descriptions
Usability evaluation	Formative evaluation (= qualitative feedback) Summative evaluation (= comparing to the user requirements)
User interaction design	Interaction and visual designs User documentation Training assets Product package

2.4 Three Dimensions of UCD

A company's capability in practicing user-centred design can be seen to consist of three dimensions (Jokela, 2001): 1) UCD infrastructure, i.e. the "extent to which a product development organisation has resources to plan and effectively and efficiently implement UCD in product development projects"; 2) Performance of UCD in product development projects, i.e. the "extent to, and quality with, which UCD processes are carried out and the results subsequently integrated into product designs in individual product development projects" and 3) Usability in business strategy, the "extent to which the business benefits of usability are understood and utilised at strategic level

planning in a product development organisation." (Jokela, 2001; pp. 150–151). Figure 7 illustrates the dimensions. A very similar definition was repeated by Venturi and Troost (2004), with the name UCD integration; additionally, they include an element of communication and culture.

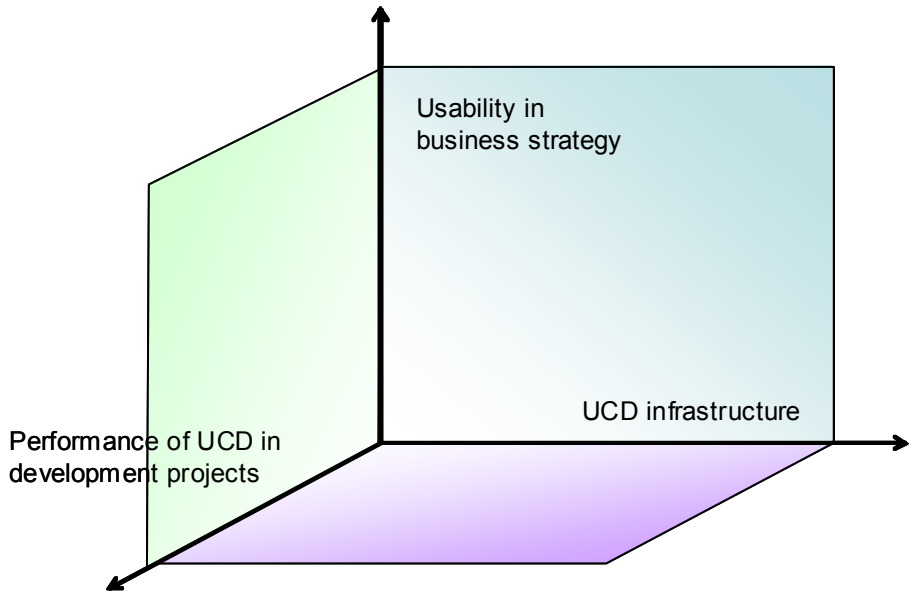

Figure 7. Dimensions of Usability Capability (Jokela, 2001)

Explained in common language, Jokela's definition states that UCD infrastructure consists of skills, methods and tools to implement UCD in product development, as well as awareness and commitment of the personnel towards UCD. The performance dimension reflects the quality and success of implementation of UCD in development. The third dimension points out the need to include usability in a company's strategic planning process and to enforce commitment to UCD among the management. Thus, to successfully exploit UCD in a company, there needs to be an infrastructure to engage in usability engineering, ability to integrate the UCD skills in practice, and management's support and willingness to assure resources to the process and motivate the personnel to respect the users at all stages of the project.

As simplified it is, the three-dimensional model clearly indicates that usability engineering requires an amount of effort to be successfully exploited in software development. Nevertheless, building up usability skills starts from the base axis – increasing the UCD infrastructure. Based on the current state analysis, a number of suitable usability engineering techniques will be discussed in Chapter 4.

3 CURRENT STATE ANALYSIS AT CASE COMPANY

To begin with the case study, a thorough analysis of the current state of usability knowledge and user-centeredness of the software process at Spellpoint Group were analyzed. The findings and analysis of the current state were then used as a basis for a training programme for the company's personnel. The purpose of the current state analysis was thus to find out about the pre-existing usability knowledge and abilities of the employees, as well as the attitudes and opinions towards usability and UCD.

This chapter describes the current state analysis conducted at the case company and presents the results of the study.

3.1 Starting Point for the Study

Usability as a theory is not new to Spellpoint Group. At the time of foundation in 2000, the company selected respecting of users' goals as one of their three values in software development. The initial interest in usability came from two partners who had studied some usability at university level. User interface design was originally one of the company's design services, but lately it has not been in Spellpoint's core business.

The company has actively tried to remind its employees of its values in respecting the user. All personnel of the company have participated in a two-hour user interface lecture about usability and interaction design held by the partners. This training has been organized twice for a larger audience, both times with similar contents: first in November 2003 and the second time in January 2005. When arranged the first time, the lecture was combined with a practical exercise where a user interface was designed for a hypothetical navigation system used in taxis; the second training was solely lecture-based. Since the spring of 2005 the user interface lecture has been part of the orientation programme for the company's new employees. All employees have also received a Finnish translation of Alan Cooper's book "The Inmates Are Running the Asylum" (1999) that presents some of the usability theories discussed in the user interface training.

The author of this thesis had the opportunity to participate in the user interface training session held in January 2005, and to make observations. The lecture introduced the philosophy of usability, discussed affordances and constraints related to usable products and software, and emphasized user goals as the basis for UI design. User profiling and use cases (see section 4.1) were presented as ways to improve software usability. A few appropriate design patterns were introduced, and some hints and tips for better user interface design were given. At the end, a few words were said about usability testing, concentrating mainly on user testing at product validation phase. The lecture was an interesting opening speech on introducing usability into software development. It raised a lot of discussion among the participants, but was considered rather short and generic.

In 2003 the company acquired the business of a small software company founded in 1985, Unifile Oy, which was joined as part of the group. Merging of two working cultures as one is still somewhat under work. Usability has been introduced to the employees of former Unifile only after the acquisition.

3.2 Targets for Data Collection

From this point the current state of UCD at Spellpoint Group was analyzed. It was supposed that the employees were at least in theory somewhat aware of usability, and that some basic questions about usability could be presented without thoroughly explaining what usability is as a concept. More in contrast, the author was interested in hearing how the employees themselves would define usability. The employees, however, were not expected to have adopted all the theory presented in that one lecture. One part of the current state analysis was thus to see what kind of impact the previous training has had on working methods and attitudes within the company.

In order to understand the current state of working methods and usability engineering at the company, different kinds of data were collected. Several interesting study areas were identified:

- Characteristics of working environment
- Characteristics of software development process
- UCD methods used in software development
- Employee's conceptual knowledge of UCD
- Employee's practical UCD abilities
- Attitudes towards usability and usability engineering

In addition to these pre-selected data, other interesting observations and comments were collected. Some more details about the data attributes are provided in Appendix A.

3.3 Selection of Study Methods

3.3.1 Data Collection Methods

To acquire the data listed in the previous section, a set of data collection methods were selected. Employee *interviews* were identified as potentially the most fruitful method for qualitative data. *Questionnaire* and *observation* were used as secondary methods for backing up findings from interviews. Questionnaire also allowed gathering some quantitative data. *Artefact analysis* was used to collect objective background information from written-form company documents. A practical *design exercise* method was developed to gain some real-life data about user-centeredness in UI design activities. The following sections present the data collecting methods more in detail.

Interview

The interview was planned as a semi-structured interview that follows a common structure decided on in advance. In a semi-structured interview the interviewer is yet allowed to present clarifying questions and continue on an interesting topic outside the initial structure. Also, the order of the questions does not need to be strictly followed. (Faulkner, 2000)

This form of interview was considered the most appropriate: same topics were discussed with all interviewees, but the structure was loose enough to allow some variation between the interviews. The possibility for modifications was seen essential

because job descriptions were different between the interviewees, and certain phases of software development were emphasized differently in the interviewees' work. Thus it was possible to discuss relevant parts of the interviewee's work more in detail while passing the irrelevant phases faster. The prepared structure is shown in Appendix B. The questions were written in Finnish as it was the native language of the interviewees.

It was considered essential to interview everyone in the company's personnel, for two main reasons: First, the employees had different job descriptions and having worked in varying projects they had different viewpoints and starting points to the UCD project. Secondly, we did not want to exclude anyone from the project. Reported experiences prove that participating in an interview is an important motivational factor later in the project (Jokela, 2001), and it was seen essential to create a positive attitude towards to the project in the company. Because the company was small, interviewing all employees instead of selecting only part of them was not unfeasible, and there was reasonable potential to reveal interesting details.

Questionnaire

The questionnaire was designed to be filled in by the interviewee himself. The questionnaire consisted of two pages: The first page contained closed-end questions where the interviewee was asked to assess his own skills in user-centred software development, orientation towards usability, and motivation for usability training. The questions were answered in a scale from one (very little) to five (very much), with an additional possibility for an "I don't know" -answer. An open-ended question about good usability was presented to measure the answerer's understanding of usability as a concept. The second page consisted of a list of certain UCD methods, and the answerer was asked to tell whether he has used or heard of them before. The questionnaire is shown in Appendix C. Like the interview, also the questionnaire was implemented in Finnish.

Artefact Analysis

An artefact analysis is a data collection method where information about people and their work is gathered by analyzing artefacts from their work. Typical artefacts can consist of documents, notes, job aids, reports or photographs (Hackos & Redish, 1998). In this study, artefact analysis was used to analyze work lists and some written work instructions, company policies, and other written material produced during software projects.

Design Exercise

Because software development and user interface design is practical in nature, some hands-on data was wanted, too. All employees had participated in a training session about user interface design practices earlier and we wanted to measure whether the employees would use the methods they had been taught. A simple design exercise was planned to measure the user interface design practices.

Observation

Observation as a research method means watching and following study subjects while they work, and recording the findings for later analysis. Observation can be participating where the observer actively participates in the work, or done from outside without interfering in the actions of the observation subjects. Observation can be done publicly, where the study subjects are informed about the observer, or secretly without informing the subjects about the observation. Observing in secret provides potentially more reliable data, as the subjects do not start changing their behaviour because of the observer; this method involves some ethical considerations, however.

While working in the company, the author participated in every-day work among other employees. This made it possible to observe working habits and the conditions where the work is done. It was decided not to inform the employees about the observation as it was done informally during work days. This was also a way to ensure reliable data. A few still pictures were also taken from the company's premises.

Summary of Data Collection Methods

Table 2 summarizes the data collected and the methods that were used.

Table 2. Data collection and methods

Data	Methods				
	Interview	Questionnaire	Artefact analysis	Design Exercise	Observation
Employees' conceptual knowledge of usability and UCD	x	x			
Characteristics of working environment	x				x
Characteristics of software development process	x		x		(x)
Documentation produced during software development (process outcomes)	x		x	(x)	
The extent to which UCD methods are used in software development	x	(x)	x		
Employees' UCD abilities in practice	x	x		x	
Attitudes towards usability	x	x			(x)
Other data about UCD and usability engineering practices	x		x	(x)	(x)

The five selected methods were considered sufficient to produce the information needed for the analysis. As shown in Table 2, each piece of data was directed with at least two methods. This allowed comparison between the methods and diminished the risk for data bias. Interviews and artefact analysis were judged as potentially the most fruitful data collection methods, and were thus used in gathering data for most questions of interest. All methods except the questionnaire would potentially produce other interesting information that could not be anticipated in advance. The purpose of the analysis phase was to open-mindedly examine all findings and pieces of information to form an understanding of the current state as thorough as possible.

Most data gathered was qualitative in nature. It was not seen as important to be able to make a formal, numerical assessment of the company's UCD maturity – mainly because the company is only beginning to take its first steps towards more user-centred design. Most important was to find out which areas and phases of software development need to be concentrated on and where new methods and tools could be integrated into present working styles. The data collection was also somewhat restricted by the confidentiality of customer projects: working methods or project characteristics involving any customers could not be reported. The allocated time for this study did not allow following an entire project from beginning to the end, and data about project work had to be collected mainly based on interviews and written documents.

Discussion of Data Collection Methods

Jokela (2001) has thoroughly studied process assessments for the purpose of improvement action. As an efficient and successful assessment method he suggests a workshop style of data collection. The participants of a certain product development team are gathered together and the development process is discussed through together. In his studies a workshop of this kind was a rapid and productive way to perform a process assessment. This also allows for presenting the assessment results straight after the workshop for everybody who is present. (Jokela, 2001)

Jokela's (2001) workshop method was first contemplated for use in this study, but was soon discarded for practical reasons. The main reasons for not performing the current state assessment using a workshop method were that, firstly, full personnel participation was needed. It would have been more difficult to organize a workshop where everybody could participate at the same time, than to arrange individual interview sessions. Secondly, Jokela's (2001) method suggests discussing certain recent projects with the project team in question. The small size of project teams (often 2–4 people) would have made this approach difficult. The author also needed time for analyzing the collected data. Using the workshop method, the results should have been presented immediately after the workshop. Naturally, the drawback in selecting individual interviews was that it required more time than one workshop.

3.3.2 Analysis Methods

To handle the large amount of data to be collected, an analysis plan was made beforehand. All interviews were transcribed using the notes and mp3 audio recordings. Questionnaire answers were summarized and averaged where possible. To further organize the data, a set of analysis tools were selected. Affinity diagramming was used for organizing and combining information. In addition, the user-centeredness of the organization was measured using the INUSE maturity model (Earthy, 1998).

Affinity Diagram

The data gathered from interviews, work list analyses, and inquiries were combined using affinity diagramming. An affinity diagram is a bottom-up approach for grouping facts and issues, and it can be used as a group decision-making tool when handling large amounts of information gathered in field studies (Hackos & Redish, 1998). An affinity diagram is built from sticky notes that are grouped together to form hierarchical entities (Beyer & Holtzblatt, 1998). The notes contain individual findings from user data, one

per note. An affinity diagram is used for classifying and organizing information, and it raises common structures and themes out of individual notes (Beyer & Holtzblatt, 1998). Beyer & Holtzblatt (1998) suggest grouping the notes without predefined categories to ensure open-minded solutions and finding of unforeseen groupings. Figure 8 illustrates an affinity diagram.

In this study, the purpose of the affinity diagramming was to gain understanding of software development practices, and to find insights and repeated phenomena in the process. The data was also to be used in forming a general process model about work habits at the company.

Figure 8. Affinity diagram.

INUSE Human-Centeredness Model

A model for assessing an organization's usability maturity was developed by an EU research project INUSE (Kuutti, Jokela, Nieminen, & Jokela, 1998). The project produced a set of techniques for assuring the usability of interactive systems or web sites (EUSC, 2000). The techniques cover a wide range of quality attributes, including system usability, ergonomics, organizational processes, and UCD capability (EUSC, 2000).

Because the case company in this study does not have an established systematic usability process, assessing the development processes as such would have been futile. However, assessing human-centeredness of the organization was considered relevant in the analysis, and the INUSE model was tried in practice.

The usability maturity model for human-centeredness presents a six-step scale for maturity. It is intended for organizational assessment or process improvement. The six steps model an organization's progress towards deeper understanding in usability. The model is based on management activities taken on each level. (Earthy, 1998)

According to the model (Earthy, 1998), the organization's maturity in respect to human-centeredness defines the appropriate ways to communicate about human-centred issues. The report states:

> An organisation with a low rating is unlikely to be able to conceive of the processes necessary to bring about the highest levels of maturity. However, they will be able to understand the benefits of the next level of maturity and will be able to see how to extend what they do now in order to improve or move up a level. (Earthy, 1998)

The descriptions of the six levels are presented in Table 3 below.

Table 3. Six usability maturity levels (Earthy, 1998).

Level X: Unrecognised. The need for a human-centred process is not recognised. If systems are received with varying degrees of satisfaction by their end users this does not cause concern. There are no positive human-centred attributes at this level.
Level A: Recognised. The organisation recognises that there is a need to improve the quality in use of its systems. The organisation has a development process and produces systems. Members of the organisation understand the business benefit of producing usable products.
Level B: Considered. The organisation makes its staff aware that quality in use is an important attribute of its products and it engages in awareness raising and training to make its staff aware that quality in use can be improved by taking account of end-user requirements during development of the product.
Level C: Implemented. Human-centred processes are fully implemented and produce good results. End-users or suitable representatives are involved in specifying and testing systems. Suitably trained staff are available as required to perform the processes which take account of user issues. Techniques are employed which are appropriate to the system, stage in the development and the end users.
Level D: Integrated. Human-centred processes are integrated into the quality process and systems lifecycle of the organisation. The systems and human-centred lifecycles are managed to ensure that the results of the human-centred processes produce improvements in all relevant work products. The required time and resources are provided for revision to improve quality in use. Information derived from human-centred processes is in a suitable format to be easily assimilated by relevant staff.
Level E: Institutionalised. The quality in use of whole ranges of systems is coordinated and managed for business benefit. The culture of the organisation gains benefit from being user and human centred. Human-centred processes are used within the organisation to improve the quality in use of the processes, tools and methods used and developed by the organisation for its own use. Quality in use defects are treated on equal terms with other system defects. Human-centred skills are regarded on a par with engineering skills.

Each level is defined by a set of attributes. Each attribute is defined by one or more management activities that must take place in order to achieve each level. A maturity assessment is done by collecting relevant data about the organization and contrasting the information of the work processes against the maturity attributes. The extent to which a certain attribute is considered in the organization is assessed on a four-point scale: not achieved, partially achieved, largely achieved, and fully achieved. For an organization to achieve a certain level of maturity, the attributes of the lower level must be realized fully, and the ones of the pursued level largely or fully. (Earthy, 1998)

The attributes for the maturity levels and corresponding management activities are listed in Appendix D. Having only made first initiatives towards user-centred design, the case company of this study is clearly still on lower levels of maturity. Thus, the two highest levels of maturity are left uncovered.

3.4 Study Arrangements

The current state analysis consisted of three parts: First, the work environment was studied by observing the office space and collecting artefacts of the work. These were used to form a rough picture of the work. Second, interviews and work list inspection were used to form an understanding of current work processes and working methods. Third, interviews and inquiries were used to assess current usability knowledge in the organization and to gather qualitative data of employees' opinions and attitudes.

All interviewees were informed about the study beforehand in the company's monthly meeting in January. They were told about the schedule of the project, and the goals and motivation of the project were presented. The author advised all employees to contact her at any time if they had any comments or questions about the project. The monthly meetings were also used later in spring for informing the employees about the progress of the study and presenting the results. The meetings made it also possible to follow-up the data collection sessions and share ideas and feelings with others.

3.4.1 Environment Observation

Working in the company, the author participated in every-day work with other employees, and therefore was a natural part of the software development organization. The author's experiences and observations of the working environment were used in describing the company's social and physical working environment. The observations gathered with this method were separated from other data to retain the objectivity in the study.

The environment observations were made informally while working in every-day projects at the office. The author observed other people work, listened to conversations, and gathered notes about interesting events and other details. The physical workplace was also examined. Other co-workers were not informed about the observations to ensure natural behaviour and thus unbiased results. The observations were gathered in a notebook and small notes and combined later to form an analysis of the environment. The observations were restricted to company's main office in Espoo; observations were not conducted in the one-man side office in Heinola.

3.4.2 Artefact Analyses

A set of written artefacts were collected to gather some background information about working methods and principles at the company. Artefacts were collected from company documents and other documentation produced during the software projects. The most important artefacts were detailed work lists that employees fill in while working in the project.

Work List Inspection

The author was given work lists of 13 projects: six software projects and seven web projects. The lists contained detailed descriptions of every task done during the projects, with the duration and description of every work task. Employee numbers and other references to them were removed from the lists beforehand to protect possible personal information in the lists. The total number of employees in the projects, and their contribution of the total project hours were given, however.

The work lists were analyzed using the task descriptions written by the employees themselves. Design and planning, implementation, testing, documentation, and customer contact and meeting tasks were picked up from the lists and marked with different colours. These phases or actions were then compared and further analyzed to form an understanding of the work tasks during a project. The division between above-mentioned tasks was not always possible, because implementation and testing are not fully separable work tasks and because different employees use different wordings in describing what they have done. However, in most cases the descriptions were detailed enough for reasoning what has been done.

In addition to qualitative data about project work, some numerical data were collected, too. The project lengths, working hours spent on the projects, the number of employees participated, and their share of the project hours were counted and analyzed. Also, the durations of the project phases were counted and compared to figure out what their proportions in the project were.

Company Document Inspection

For document inspection, the author went through the digital archive of general company documents. Also the company's web pages were examined. As a result, a small set of documents were collected for analysis.

3.4.3 Interview Sessions

An interview session was organized with employees and the management, 14 in total, to get an overall picture of the current state of the company. Although all employees do not participate in software implementation and code writing, they may be involved in the first phases of planning and requirements definition, or participate in customer relationship management. Thus it was decided that the same questions would be presented to all employees no matter what their position or job description was.

Before deciding on the final questions and structure of the session, a preliminary interview was held for one employee. This semi-structured interview was held two months before the actual data collection as this employee was to be absent the rest of the spring and was not to be able to attend to the training sessions later in spring. This one-hour interview gave some insight into the problem in hand, and helped in forming the final questions for other interviews.

After the preliminary interview, a short questionnaire and a design exercise were added to the interview sessions. The interview questions were slightly modified, and a few more were added after analyzing the interview results. All the other sessions (13) were held within a fortnight, in the beginning of March 2005. Three and a half hours were

reserved for each interview session, allowing one or two interviews per day. The interview sessions took place in a meeting room in the company's main office in Espoo, except for one that was held in the company's side office in Heinola.

The Course of the Interview Sessions

A meeting room was reserved for the session. Refreshments such as coffee, tea and cold drinks, and some snacks were served in the sessions to make the session comfortable for the interviewee. At the beginning of the session, the reason and goals for the study were explained, and the interviewee was told that this study was a part of the author's diploma work at university. The main goals, i.e. understanding the work practices and current knowledge in usability at Spellpoint were named. The interviewee was told that the details of the discussions would remain confidential, and that individual employees' answers would only be reported without names in the study results. He was explained that instead of measuring an employee's individual performance, the company as a whole was the target of the study, and all comments and opinions would help in forming the big picture of the working methods. Any questions regarding the study were answered if presented by the interviewee. After that, the session's agenda was presented: First, a design exercise would be given. Then, there would be a short questionnaire, and finally a set of interview questions. The employee was advised to ask for a break at any time if he felt that the session was too long.

Exercise

A design exercise was given to the interviewee in the beginning of the session. White paper, pencils, an eraser, and a ruler were available, and there was also a laptop for the employee's use. The instruction was to design a currency converter tool for a Finnish bank's web pages on the Internet. The employee was advised to draw or write a description of the tool: what it looks like, how it is used, and what it can be used for. If one wanted, one could use the computer, too, but hand-drawn was as good. It was stressed out that an outline of the user interface is enough and graphical details are not important. For that reason any colour pencils or equivalent were not supplied. The technical implementation was not relevant in this exercise and the employee was advised not to think about it. The instruction was loose enough to give the employees free hands to design the tool as they considered best. Any instructions for applying the usability methods familiar from the user interface lecture were not given to the interviewee to avoid affecting the results. The interviewee was only told that all material produced during the designing would be collected for later analysis.

The interviewee was told that the author would leave the room and wait outside the meeting room and then return after half an hour to ask if the design was ready. He was advised that if he had any questions or problems during the exercise, he could call the author at any time. He was also advised to call if he was ready with the design already earlier than 30 minutes. Finally, the employee was asked if he had any questions about the instruction. When the employee felt that the instructions were clear, the author wished him luck with the design, and told the exact time when she would return to the room. It was considered important that the interviewer left the room for that time, to ensure that an observer's presence would not affect the employee's performance.

After 30 minutes the interviewer returned to the room and asked if the employee had finished his design. The interviewee was allowed to request some more time to finish the design if required. Three interviewees were given five more minutes, and one used ten extra minutes. Five interviewees were ready earlier, using only 15 to 25 minutes to the exercise. The time used thus varied between 15 to 40 minutes. The interviewee was then asked to present his design and to describe how he ended up with this solution. He was also asked how he felt about the exercise in general.

Questionnaire

After the design exercise, a short break was suggested and it was held if the employee wanted. After that, a questionnaire was given to the employee. He was reminded that individual questionnaire results would not be reported in the results and that the main goal was to gather information about the company as a whole. He was advised to ask if he did not understand the questions but otherwise he could fill in the questionnaire by himself.

Interview

After the questionnaire an interview was held. The interviewee was asked for a permission to record the interview. He was told that the tapes would only be used as backup notes and later analysis, and they would not be played for other people.

The largest topic discussed in the interviews was the work flow and phases of a typical project. To avoid overgeneralizations and talking at an abstract level, the interviewee was encouraged to think of a recent project whenever possible and to give real examples. The interviewer presented several additional questions about interesting things to get the most out of the interviews. Even though the same questions were gone through with all interviewees, the contents of the interviews depended somewhat on the position and job description of the interviewee.

The interviews lasted from 30 to 100 minutes depending on the interviewee's activeness in comments and discussion. After the interview, the employee was thanked for his time. He was told that the results from the interviews would be presented in the company's monthly meeting on the 1st of April 2005 and that according to a preliminary plan the upcoming trainings would be organized in April or May.

The first interview session in March served as a pilot and one day extra was left between it and the other sessions for analyses, and to allow small modifications to the exercise, questionnaire, or interview questions. After the pilot session, a few wordings were changed in the questionnaire questions to clarify them. The 30 minutes' time for the design exercise was seen to be possibly too short and it was decided that the employees can have five or ten minutes of additional time on request. The interview questions were not modified.

3.5 Study Results

This section presents the results gathered during the current state study. For better traceability, individual results are presented for each data collection method. In section 3.6 the results are combined and final conclusions about the current state at the

company are presented. The results from the two pilot interview sessions (January and March 2005) were also included in the analysis.

3.5.1 Artefact Analysis Results

Company Documents

Corporate look is highly appreciated. This can be seen from well-designed web pages and various document templates for documents and presentations that are available for all employees. The work processes are not that acknowledged: there are no statements about recommended software process models, nor any specific guidelines for software design. One formal instruction was found: a HTML Coding Standard (Spellpoint Software, Spellpoint Interactive, 2003; internal document) that aims at quality code in Internet pages. Project documents and files are stored using a shared disk space for the employees. There is not, however, any archive for sharing best practises or hints and help with others.

Work Lists

The analyzed work lists consisted of six software projects and seven web projects. The web project lists contained all hours spent on the projects; however, in five software projects the project manager's hours were not included. This was due to the fact that the project manager in the projects in question was the old CEO of the former Unifile Oy, who worked in the projects as an outside consultant after the Unifile acquisition. Consequently, the work lists of those projects covered mostly the implementation, but the hours concerning especially customer contacts and the manager's contribution in product specification and design were not available. Despite the fact, the work lists provided useful qualitative information about the course of typical projects; the quantitative results can only be taken as suggestive.

In practice the project teams are often rather small. In the six analysed software projects the average project team size was 3.66 employees. In the seven web projects the average was slightly bigger, 3.85 employees. The work list statistics showed that in software projects the main responsibility of the implementation is often largely a single person's responsibility. In two out of three projects, more than 90 percent of the implementation had been done by one software developer. In other projects there had been two employees sharing the responsibility. In web projects the division of work does not show as clearly, but also in those projects a single key responsible can be found: in five out of seven projects one person had been responsible for more than 50 percent of the work done.

The lengths of the projects vary a lot. In the thirteen analyzed projects the lengths varied from 1 to 13 months. On average, the lengths of software projects were somewhat longer than web projects (6.67 vs. 5 months), but the hours committed to software projects vary more than in web projects. In half of the software projects the implementation phase took more than 450 hours; the rest three were less than 80 hours'

projects.[1] Web projects span from 43 to 163 hours, with five out of seven projects consisting of less than 80 hours. Considering that the total lengths of the projects measured in months did not differ that much, it can be stated that the implementation phase is more intense and time-consuming in software projects.

The implementation of a typical software product starts without a clear design phase. A clear end for planning cannot be identified from the work descriptions; instead, the implementation is started parallel to planning and specification. The implementation is tightly related to testing and together they seem to form a cyclical pattern: after implementing a piece of functionality, the software is tested and refined until it works correctly. Changes and a large amount of small fixes are done on the grounds of customer feedback.

According to the work lists of six software projects, two projects involved some written documentation other than comments in the source code, and additionally a user manual had been written in one other project. Another of the two projects had started with a written specification of more than ten hours, and the hours used in documentation in various phases of the project were close to 35 hours in total. In the other project, documentation was not as extensive, and the descriptions wouldn't reveal what the document was about. Three projects involved no documentation. In the seven analyzed web projects, the written material seemed to involve only bids and business contracts.

All projects contained smaller or larger breaks in the middle. Most typically a break is caused by a customer, whose feedback or response to a certain feature is waited for. Holiday seasons in summer and winter also cause breaks to the projects. In software projects, the phase of implementation and testing (that are quite inseparable) is rather coherent; instead, breaks are typical in phases of planning and final integration. Web projects typically include demos or design suggestions that are reviewed by the customer. In those projects the implementation phase is more discontinuous and breaks are taken every time customer input is needed. In web projects, testing as a separate action does not stand out as strongly as in software projects; nevertheless, some testing has been reported in five out of seven web projects. In the analyzed software projects, only one minor project did not include any specified testing.

The interpretations of the project activities were made based on the written description of the task. The terminology used for certain tasks or activities naturally varied between the employees, and the lengths of the descriptions ranged from one-word statements to longer sentences. The work listings had been written in English in all but three earlier projects. There is a reason to expect that the use of foreign language in reporting may have had a minor effect on the thoroughness or precision of the descriptions.

3.5.2 Interview Results

The interviews consisted of three major topics that addressed common knowledge of usability, the software development process in the company, and usability engineering in software development. This section presents the results from the interviews.

---

[1] Note that the lengths of five software projects do not include customer contacts and other project administration.

UCD Knowledge

Usability and user interfaces were discussed at the beginning of the interviews. All interviewees had an understanding about usability to the extent that they could name some attributes related to usability, and they understood that it is important that a program is easy and satisfactory to use. Commonly, "easy to use" and "usability" were understood as synonyms. User interface as a concept was generally more vaguely definable. Software components, "button systems", and graphical user interfaces in general were discussed, and almost everyone recalled a particular example from the company's user interface training: that a door handle is also a user interface. Rather surprisingly many interviewees were unsure whether a web page can be regarded as a user interface: the term seemed to be related more with stand-alone PC applications (programs) or any other physical tools such as a hammer, a thermos jug – or door. One interviewee commented that there is an expectation from the customer that web pages are visually pleasing, but he wasn't sure whether it has to do with usability. At the other extreme, the technical explanation for user interface, "an interface between man and machine," was offered in several interviews.

Below, some comments about usability and user interfaces in general are given. The translations for the quotes are presented in end notes.

I1: *"Silloin kun jollain asialla on hyvä käytettävyys, se ohjaa itse itseään silleen, että teet alitajuntaisesti asioita, ilman että välttämättä huomaat että tekee. Sun ei tarvitse keskittyä itse siihen."* [2]

I11: *"Hirveintä on, että ohjelma tekee jotain, mitä se ei kerro mitä teki, että menettää kontrollin. Siitä tulee tyhmä olo, että teinks mä jotain väärin."* [3]

The interviewees generally agreed that it is important to consider usability in all software products. Four of them added that when designing an interface for their own use only, usability is not an important factor, though. All interviewees' unanimous opinion was however that whenever there are many different users who do not know the system, it is important that it is easy to use.

Usability engineering and UE techniques were not known by many interviewees. Those who had included some usability in their studies had heard about user testing, but yet had not tried it themselves. The two techniques presented in the previous UI training, namely user profiles and use cases, were best recalled: nine interviewees mentioned thinking about users or user profiling as a design technique, and seven interviewees mentioned users' goals or use cases. Usability evaluation as a topic was quite unknown. Seven interviewees were able to tell that a user can be asked to look at or try a program, either with given test tasks or without. The very method of usability testing was not known to all of them, however; some of the interviewees actually presented user testing as a hypothetical way to evaluate usability, and added that they had never used or even

[2] "When something has good usability, it steers itself in such a way that you do things subconsciously without necessarily paying attention to it. You don't need to concentrate on it."

[3] "The worst thing is that a program does something but it doesn't tell what it did, that you lose control. It makes you feel stupid, like, did I do something wrong."

heard of anyone using such a technique. Interviewing users about a program was also mentioned by a couple of interviewees. One employee recalled participating in a situation where program was tested with a user and later the user had been interviewed. Usability testing was surprisingly not very well remembered, even though it had shortly been referred to in the UI training, too.

Software Development

Most of software produced at Spellpoint is tailored for customers' needs. In the past years there has been some in-house software development as well, but the main focus is currently on customer projects. A typical project team consists of two to four people, including project manager and one or more persons in implementation.

In many projects the initial specifications are received from the customer, either in written or oral form. The specifications – or "specs" – are often negotiated with the customer and the vendor company can also influence the final agreement. Especially in larger projects or in cases when the customer is unsure of what they want, the specification is done in tight co-operation with the customer. Often the specification is very loose and generic at first, stating only the main purpose for the project, and it is further adjusted in the course of the project. However, the specification document is usually produced in some form, more or less formal. According to the interviewees, the specifications are very technical in nature, mostly listing the needed features and often already including a plan for database implementation. Technical constraints such as compatibility with other systems or safety requirements may also have been included in the specifications. In web projects the customer may have a clear vision of the layout, or they can give the graphical designer free hands in making one or more suggestions.

Customer is contacted often in a daily basis during the specification. Customer contacts are mostly managed by the project manager who is also responsible for the specification. The software developers are usually not in direct contact with the customer, unless when participating in project work at customer premises.

The implementation of the software typically starts already before the software project has been thoroughly specified. Due to tight schedules, the specification and implementation phases often overlap. The implementation may involve a quick demo version: in several interviews it was stated that a customer often wants to see a preliminary version of the software before they can fully decide what they want. One software developer said that by making a quick functional demo the customer is shown what the company can deliver fast. The demo version is rarely planned in advance but more than once it has ended up in use at a customer, when the customer was happy with the demo version. Typically the specification is revised if problems occur or some details have not been fully specified in advance.

> *I9:* *"Kun asiakkaat testaa, tulee palaute että tämä kohta pitää tehdä toisin, tämä kohta ei toimi. Silloin tietää tarkkaan että mitä pitää tehdä ja tarkentuu se työ. Tulee niin kuin speksit jälkikäteen."* [4]

[4] "When customers test [the program], we get feedback that this feature needs to be done differently, this does not work. Then you know exactly what to do and the work gets better specified. You kind of get the specs afterwards."

Non-functional mock-up demos are not really used for presenting design ideas to customers. More than one interviewee stated that showing the design too early is problematic; the customers would only comment on irrelevant details. Also, when something has been shown to the customer, they soon get used to the design and may resist further changes in the user interface even when they clearly are needed. One interviewee even said that it is the task of the project manager to anticipate what the customer wants.

Software development is incremental in nature. The software is implemented in small increments that are tested as soon as they are ready. The customer is contacted on a regular basis for further specification, feature discussions and information about the project's schedule. More contacts are taken when parts of the software are nearly finalized and they are sent to the customer for acceptance tests. Once accepted, the customer may soon take the software in use. Later increments bring new functionalities and additional features to the software.

There is no specialized testing division in the company; usually the developer tests his own code for functionalities. Usability or user interface is not tested separately beyond functional elements. At the time of testing the project becomes less intensive as customer input and comments need to be waited once in a while. The software is revised and improved until an acceptance is gained from the customer.

According to the developers, sometimes the customer is unhappy with the results when they finally see the functional program. Usually the software designs are not shown to the customer before there is something to test. If the user interface or the functionalities of the program are not what the customer wanted, even large modifications may be asked. Sometimes, some new specification needs to be done – this has sometimes happened even at the end of a project.

The customer contacts are still retained after the delivery, and further updates are made later. New update requests may come after years of use, for example when the customer changes the operating system on their computers.

In short projects the work may be very straightforward, instead. After short – if any – planning the program is written and quickly tested. Typically these programs do not require much customer involvement and contacts are less frequent. These projects are typical one-man projects with little or no documentation.

Documentation is not an established part of every project, and according to the interviewees, too often the only documentation produced is the source code with few comments. There are no formal instructions in the company about what to document. A technical documentation or user manual is written if the customer is willing to pay for it. Often this is not the case as the customers have tight budgets. The software developer who has written the program writes the documentation. Sometimes the initial specification document may also evolve to a technical document by updating details during the implementation.

One software developer said that writing the documentation or a user manual is difficult, because he does not always know what the customer wants to do with the program. He went on to describe:

> *I8: ”Joissakin tapauksissa on jopa niin, että meillä ei ole aavistustakaan siitä – – miten sitä ohjelmaa käytetään, joka me ollaan asiakkaalle myyty. On ollut tapauksia, jossa olen ollut näkemässä, miten asiakas käyttää sitä ja ollut ihan ihmeissäni, että eihän sitä noin ajateltu.”* [5]

Another interviewee told that missing context information also makes program testing difficult: when use cases are not known it is difficult to know what functionalities to test for potential problems.

Usability at Work

When asked how usability and user-centeredness show up in daily work, many answered quickly that surely it shows up every day. Nevertheless, few interviewees were unable to give any real examples. Unanimous opinion was, however, that everybody is willing to try their best and no one does bad usability on purpose. A few interviewees mentioned sometimes thinking about users when designing an interface or writing use cases during the design process. However, they were not able to name many specific projects where those techniques had been used. Currently, any usability testing or other evaluations are not done at the company. One interviewee named the company's own web pages as an example of good design and quality that shows up in daily work. Usability and general software quality seemed to link in many interviews.

It was brought up repeatedly in the interviews that it is difficult to enhance software usability within the short time that is usually reserved for the implementation. According to the interviewees, the biggest reason for not putting more effort in usability is that customers are not willing to pay extra for that. Technical constraints can also stand in the way for better design. One interviewee explained:

> *I1: ”Jos haluaisin tehdä sellaisen [ohjelman], jota on mukava käyttää, se on teknisesti usein ylivoimaisen raskas tai työläs toteuttaa. Jonkun muun pitäisi mitata onko se kannattavaa, mutta – – jos annat työmääräarvion, että se on tunti että tää saadaan toimimaan ja viikko että tää on hyvä, niin usein sanotaan, että otetaan se tunnin juttu. – – Toisaalta päätökset eivät ole minun ongelmani; jos halutaan päättää että tehdään huono, niin tehdään.”* [6]

It was, however, pointed out from the management that company's abilities in usability have been brought up in project negotiations whenever suitable. Also software

[5] “Sometimes we don't even have a clue – – how the program that we've sold to the customer is used. There are cases where I have seen how the customer uses it and I've been astounded that this wasn't the way we planned it.”

[6] “If I wanted to make a program that is nice to use it is often technically too heavy or time-consuming to implement. Someone else should measure whether it pays off to implement, but – – if you give an estimate of the workload saying that it takes one hour to make it work and one week to make it good they will often choose the one-hour thing. – – On the other hand, decision making is not my problem; if someone decides to make it badly, then we will.”

developers themselves told that they try to pay attention to usability aspects when they can – still aware of their limited skills in making rational decisions between good and bad design. One employee stated that in his opinion, usability is given more attention than functional testing of software.

The interviewees reported many practical problems related to usability and software development in general. Many of them also related to working with customers.

> *I1:* *”Siis, kyllähän niissä [ohjelmissa] on ollut aika monessakin käyttöliittymä, mutta [en osaa sanoa] onko niitä* suunniteltu *vai onko ne vaan roiskaistu ne nappulat sinne… On tehty enemmän siltä pohjalta että on aluksi tehty ohjelma ja sitten on lyöty nappuloita sen mukaan, mitä se sisältää tai mitä halutaan. Lähestymistapa on ollut sillä tavalla nurinkurinen.”* [7]
>
> *I9* *” – – se on itsessään ongelma kun [asiakkaalta] ei ole kerrottu mitä halutaan ja on liian huonosti speksattu, eikä asiakaskaan tunnu aina tietävän mitä haluaa. Sen tuntee probleemaksi, kun pitää itse kehittää mielessään, että asiakas varmaan tahtoo tätä. Joskus se osuu oikeaan ja joskus menee kauas metsään.”* [8]
>
> *I7:* *“On asiakkaita, jotka saattaisivat kummastua suuresti, jos sanotaan että tehdään joku suunnitelma tai määrittely. Tai ainakaan eivät maksaisi siitä.”* [9]

An interesting misconception about the use of UCD methods was revealed when the interviewees were asked whether any usability engineering methods were used in the company: more than one interviewee stated that they thought so even if they had not done it in person. An interviewee told:

> *I6:* *“Koulussa kirjoitettiin juuri 20 sivua kahviautomaatista. Nyrkkisääntönä on, että ilmiöistä keksitään mitä tarpeita tai toiveita käyttäjillä on ja tehdään niistä valmistettavalle tuotteelle vaatimuksia. En tiedä tehdäänkö Spellpointissa, mutta luulisin että tehdään.”* [10]

In general, the attitude towards usability was good. The interviewees agreed on that usability is an important part of software quality, and everyone was able to recall examples from their own life where a program had been difficult to use. The contents of

---

[7] “I mean, surely most of them [programs] include a user interface but [I can’t tell] if they’ve really been designed or if the buttons have just been thrown there… We’ve mostly first done the program and then put buttons there according to what it consists of or what we want. The approach has been backwards that way.”

[8] “It is a problem when [the customers] have not told what they want and the specs have been done poorly, and the customer doesn’t always seem to know either what they want. I feel it’s a problem when you need to figure it out in your own head that this is probably what the customer wants. Sometimes [the design] goes right and sometimes it goes wrong.”

[9] “There are customers who would be surprised big time if we told that we will make a plan or a specification. Or at least they wouldn’t pay for it.”

[10] “At school [university] we wrote 20 pages of requirements for a coffee machine. The rule of thumb was that phenomena are used to figure out what kinds of needs or wishes users have and to make requirements out of them. – – I don’t know if this is done at Spellpoint, but I think it is.”

the company's user interface lecture had vanished from the minds of most interviewees – even some of those who had attended to it only two months earlier were not able to recall what exactly was discussed. The interviewees noted that when the methods have not been used in practice, they feel remote from daily work and difficult to use. No one expressed being uninterested in further training in usability; instead, most interviewees pointed out that they could use some revision.

I8: *"Voi, mä haluaisin oppia enemmän, omat tekeleet nolottavat hirveästi."* [11]

I11: *"Musta olisi inhottavaa että ihmiset kärsisivät siitä päivittäin, että joutuvat käyttämään mun tekemää ohjelmaa."* [12]

I3: *"Uskon, että [firman aikaisemmalla] koulutuksella on ollut vaikutusta työtapoihin, olen kiinnittänyt enemmän huomiota asioihin."* [13]

I1: *"Kertaus olisi paikallaan jossain muodossa, koulutuksesta on jo pari vuotta ja asiat ovat unohtuneet. Niitä [oppeja] ei ole oikeasti käytetty, vaikka ehkä pitäisi enemmän käyttää."* [14]

In the end, the general opinion of the company's state in knowledge of usability was unclear. The responses of the interviewees varied a lot. Some saw that the skills and knowledge in the company were very good, when others thought that a lot needs still to be done.

I5: *"Työpaikalla asenne käytettävyyteen on aika positiivinen. Joku saattaa kysyä, että mitä olet mieltä tästä jutusta, ja se on positiivista."* [15]

I13: *"En henkilökohtaisesti usko, että hyvän käytettävyyden soveltaminen on sillä tasolla millä sen pitäisi olla. Kun asiakas ei halua ja sisäisissä ei ole tarpeeksi tekijöitä. Se on huono tekosyy."* [16]

I4: *"Käytettävyyteen on käytetty suhteellisen paljon aikaa, uskon että enemmän kuin muissa firmoissa ja etenkin näin pienissä."* [17]

---

[11] "Oh, I would like to learn more, my own work embarrasses me terribly."

[12] "It would be horrible that people suffered every day from having to use a program I made."

[13] "I believe the training has had influence in working habits, I've paid more attention to things."

[14] "I could use a revision in some form. It's been a couple of years since the training and things have been forgotten. They [methods] have not been really used, although maybe we should use them more."

[15] "At work the attitude is positive. Someone can ask 'what do you think of this thing' and that is positive."

[16] "I don't personally believe that the implementation of good usability is at the level where it should be. The customer doesn't want, and in-house projects don't have enough people. That's a bad excuse though."

[17] "We've used relatively lot of time in usability, I believe more than in other firms and especially as small ones as ours."

I8: "Se [käytettävyys] on vähän sellainen lapsipuoli. Siihen ei suhtauduta ylenkatseella mutta ehkä olankohautuksella. Siitä on puhuttu niin vähän, että en ole varma, että ollaanko noloina kun tää on vähän tällaista vai suhtaudutaanko humpuukina vai jotain siltä väliltä. Ei kauhean vakavasti oteta päivittäisessä työssä huomioon." [18]

3.5.3 Questionnaire Results

The interviewees' attitudes towards usability and previous knowledge of the topic were measured also with a questionnaire (see section 3.3.1). The main implication of the questionnaire results was that usability was highly appreciated by the interviewees and that they were motivated in getting more information on the topic.

Skill Assessments

The interviewees were asked to estimate their own abilities and skills in taking users into account in software development on a five-point scale (1 = poor, 5 = very good). This question resulted in an average grade of 3.38 with the median in 3. Those respondents who do not usually participate in software development found the question a bit difficult, though. When asked to assess the abilities of the company as a whole, the average grade raised to 3.53 with the median in 4. The answers reflect the interviewees' general view that the company knows good usability. Quite interestingly, the respondents were not very sure of their personal skills although they are the only ones responsible for usability of the software produced. Yet it is true that in a small organization the competencies on individual level are emphasized and can have a big effect on the perceived quality at company level as well. On the other hand, the assessments show the employees' respect in the company to deliver quality products from usability viewpoint as well.

Usability and Training Interests

The respondents were asked to name factors that affect good usability. The three most important factors named were clarity (seven mentions), intuitiveness (six), and simplicity (five). The first two were also ranked as the most important factors. Other attributes mentioned were speed, functionality, logicalness, consistency, and efficiency, among others. Quite interestingly, three respondents mentioned meeting the user's goals as the most important factor – as presented in the company's former user interface training.

Usability was regarded as very important part of software among the respondents, with an average grade of 4.8 of five (= very important). Twelve out of thirteen respondents said that they were interested or very interested in learning more about user-centred software development, and eleven respondents were interested or very interested in learning new usability techniques or methods. The job descriptions of the respondents affected these ratings: those who did not see themselves as being part of every-day

[18] "It's [usability] kind of a stepchild. We don't scorn it but maybe just shrug at it. We've talked so little about it that I'm not sure whether we're embarrassed when this is what it is, or if we think it is nonsense, or something in between. It [usability] isn't paid very much attention to in daily work."

software development showed a little less interest in training. On average, the interest rate for both questions was as high as 4.3 proving that there is significant interest in the topic. The average grades rise to 4.57 for new information and 4.71 for new methods, if the administrative personnel and other people not involved in software engineering are left out of the count.

Usability Engineering Techniques

At the end of the questionnaire the respondents were asked to think whether they had heard or used some of the usability engineering techniques presented. Eleven out of fourteen respondents (the pilot interview from January also included in the total count) reported knowing and having used use cases sometimes as a design technique, and the rest three had also heard about them. User profiles were also recognized by all respondents, with seven of them having used them somewhere. The differences between user groups and user profiles or use cases and use scenarios were not very clear to the respondents and thus the results are rather vague for these techniques. The same applies to the technique of environment analysis: the respondents were unsure of its meaning but commented that they could guess what it means without really knowing. Including descriptions of the techniques would probably have made the answers less ambiguous, but other interesting details would have been missed that way. For two respondents, use cases were familiar from UML notation that is used in modelling object-oriented software structure. One respondent said that the user groups and user profiles are familiar from the context of operating systems – apparently referring to access rights or customized user preferences not related straight to usability. Overall it seemed that the techniques were somewhat more familiar from studies than real work done at the company, and that the terminology does not seem to be very established.

Conclusions

The questionnaire results implicate that usability is not systematically perceived to relate to clear set of any specific techniques. The respondents were able to name certain attributes related to usability, but conceptual understanding and practical usability engineering skills are probably vaguely mastered. This result is hardly surprising, because none of the respondents were usability specialists, and the main part of their current job does not include usability engineering.

3.5.4 Design Exercise Results

The design drawings from the 13 interviewees were analyzed based on the actual solution and the way the designers described their solutions. Considering the short time they were given, many had not had time to draw or write all details on paper, but they completed their design when describing it to the author. Both the verbal descriptions and the design on paper were used in the analysis.

Design Principles

When asked how they ended up with the design solution, ease of use and intuitiveness were offered as main goals in many answers. Four designers told that they had aimed at a simple and fast solution without excess features, and two more added that they wanted the tool to be so easy that everyone knows how to use it. Three designers reported that

they had either searched for similar tools on the Internet or remembered seeing one before, and that they had used those tools as an inspiration or for benchmarking. Two more told that they had thought about going to see some examples but had not done that after all.

Although not specifically asked for, three employees wrote user profiles and use cases (see section 4.1) on a paper. Two to four different users had been identified, including travellers, businessmen, bank customers, and ordinary people who buy things on the Internet. Altogether in seven interviews the employee reported having at least thought about some users or use cases during the design, but four of them had not written them down.

Solutions

Examples of two different design solutions are shown in Figure 9 and Figure 10 below. The simplest solution for the tool included two text fields – one for each currency – and two drop-down menus for the currency name. With small variations in either the arrangement of the components or possible titles placed next to the fields, altogether seven designers had ended up with such a solution. The exchange rate between the currencies was shown next to the result in altogether four designs.

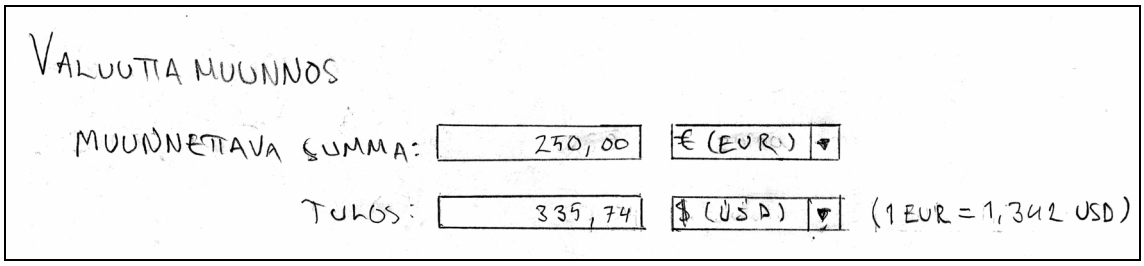

Figure 9. Simple solution with basic functionality only.

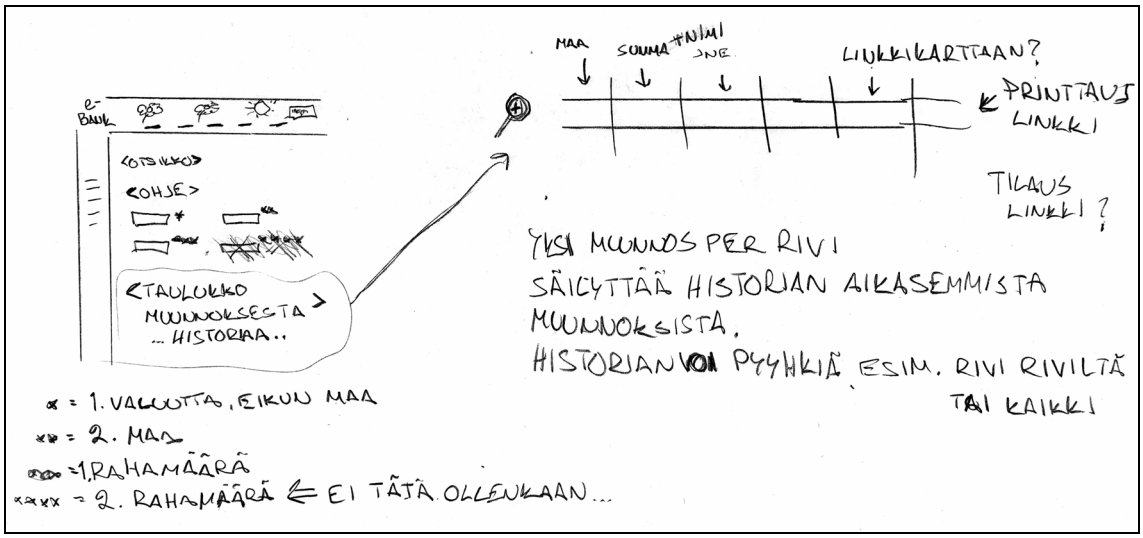

Figure 10. Solution with advanced features; the conversion history is shown in a result table.

Six designs provided a clear way to search currencies by the country name, and one of them also allowed selecting the country from a map. One design even supported reading a user's mind or eye movement for selecting the currencies.

Shortcuts and Help

Four designs offered some kind of shortcuts to often used currencies. In one design there was an anticipating typing functionality that offered a first match as the currency or country name was typed. None of the 13 designs required typing the currency name or abbreviation in any field as the only choice for use. Altogether seven designs made the conversion automatically without an additional "Convert" button. Three designs worked both ways so that any of the currency amounts could be modified and the tool would convert it to the other currency. The rest worked only one way.

Four designs included a short instruction how to use the tool. Three of them also provided a button or link for more instructions, and additionally one other design included such a link. Most interestingly, none of these four designers providing help were software developers by job description. Only two designs allowed the user to select a language.

Additional Features

Altogether seven designers had come up with some additional features or functionalities for the tool. One design showed consecutive conversions in a table maintaining a conversion history for the user; the history could be deleted and printed. Another design allowed making several conversions between the two currencies at a time, and labelling the conversions with descriptive names. Two designs allowed the user to choose more than one target currency in which to convert. Two solutions also included a possibility to print some kind of a reminder note from the system to better support trip planning: one with a map labelled with country and currency names, and another with a small currency conversion matrix. Furthermore, one design integrated the currency converter tool into an electronic banking service allowing the user to convert amounts from an account statement by mouse-clicking the figure. In addition, two of the before-mentioned converter tools came with an extra service for bank customers that allowed pre-ordering foreign currencies in a bank office. Two designs included a chart of the exchange rate of the currencies and one of them came with a possible economist forecast for the upcoming price development; two more designs had included a link to a list of current exchange rates.

Conclusions

Being a simple tool with very little functionality, the design exercise did not produce remarkably poor design solutions. Most tools were nicely usable with a clear and simple user interface. Some minor yet identifiable problems were the use of three-letter currency abbreviations instead of currency or country names, missing search functionalities, and lack of help in using the tool. Also, the drawings were somewhat ambiguous and rather small, making their interpretation difficult. Without the verbal explanation from the interviewees, parts of the designs would have been hard to understand. Clearly the limited time available may have caused some of the ambiguity, and the instructions did not clearly demand finalized drawings either.

More interesting than the actual final design was the reasoning behind it. Straightforward and fast use was seen as important characteristic for such tool, which is a justifiable reason for cutting down extra features. The number of special widgets was not regarded as an automatic bonus for the design, but the arguments for explaining the design were also appreciated.

Thinking about users and use cases before drawing the design seemed to have a positive effect on the functional versatility of the tools. Those who had listed even some typical use cases had been able to find ways to facilitate them with shortcuts or some extra features. Those designs where users had not been thought about were clearly the simplest ones. Yet, it can be argued that if the goal was to design as simple tool as possible, it had been well achieved in those designs. Although providing extra features was not the primary goal in the exercise, it is interesting to notice that paying attention to possible users and their needs did in all cases lead to inventing more uses for the tool.

3.6 Analysis of the Current State

The results from interview sessions, artefact analysis, and observations were combined to form an analysis of the current state at the company. The work processes and working methods at Spellpoint were analyzed using the data collected from interviews and work lists. Individual observations about project work, methods and phases were collected from interviews and grouped according to project phases. The groups consisted of project specification, customer contacts and meetings, software and UI design methods, implementation, and testing. Comments or references to possible UCD methods such as user and task analysis, requirements definition, user interface design, and usability testing were thoroughly searched from the interviews. The data collected from the interviews were contrasted with task descriptions from work list. The work lists supported the observations gathered from interviews by providing numerical data and more details.

The work environment is described from both physical and social perspective. The work processes and working methods are then described, and usability at work is discussed. The personnel's knowledge in usability and UCD is reflected and a statement of the company's usability maturity is presented.

3.6.1 Work Environment

Spellpoint has two own offices in Southern Finland. The main office in located in Espoo and at the time of the study there were 9 people working there, some of them part-time. A small side office is located in Heinola, about 200 km away from the main office. One employee is currently working in the Heinola office.

In addition to Spellpoint's own office space, several employees are working for customers as a part of customers' software development teams. At the time of this study, four employees were working full-time and three employees part-time at customer premises in the Helsinki metropolitan area.

Physical Work Environment

The company's main office in Espoo consists of two open office rooms and a meeting room. The three spaces are separated with glass walls. The management shares one

room and the employees have their workstations in the other room. Large windows at both sides of the office make the rooms well-lit, and light wooden and aluminium colours used in office furniture give the office a fresh look.

In the open office space the employees' workstations are arranged in the form of a clover leaf, in a circular shape. The workstations are not reserved for one employee, and the employees can use any of the free stations. In practice, however, the employees working at the office tend to choose the same workstation every day.

At the time of this study, a small coffee table was positioned by the entry door. Later in the spring, after the current state analysis the company rented two more rooms – a work room and a separate coffee room.

Figure 11. Workstation circle at Espoo main office.

The one-man's office in Heinola is made up of two small rooms. Earlier there were two employees working there, but now the other room is turned into a coffee room with a coffee table in the middle. The other room is the actual office with two computer tables and PCs. Because of the physical distance between the two offices, daily contacts from Heinola office to project management and other company's personnel are restricted to e-mail and phone.

Social Work Environment

The open office space in main office allows free conversation between employees who are working in the same workspace. This makes it easy to ask for a colleague's opinion for a topical matter, or to have a briefing or a small design meeting with someone else working on the same project. Discussions about current projects are typical and absolute silence is not required. Yet, to enable employees to concentrate on their own work, sets of soundproof headphones are available.

A strict hierarchy cannot be identified in the company. A flat organization is a natural choice because of varying projects and a lot of individual or team-based working.

The geographical division of workplaces causes a social gap between employees working in different places. Those employees who work daily together have a tighter social relationship than what has formed between employees working in separate locations. A few employees have never worked at the main office, and thus the working habits and culture there are quite unknown to them.

Monthly meetings are organized to get the personnel together for information sharing and other topical conversations. These meetings also serve as a more informal get-together for employees. These meetings also help in forming a common social culture among the employees, which is especially worth investing in because of the physical separation of the employees. The company also organizes evening events for the personnel once in a while, to further develop good sprit in the company.

3.6.2 Work Processes

Project Model

The study proved that there is no established software process at the company; instead, projects are conducted somewhat from an ad hoc basis, depending on the schedule, budget, and other customer requirements. The projects could be described as incremental build-and-fix way of producing software. There is, however, a common pattern identifiable in most of the larger projects.

In Figure 12 a generalization of a typical software project is presented. The activities of planning, implementation, testing and documentation are shown with coloured bars. The number of customer contacts is illustrated with a sliding colour-bar: red means many contacts, violet few contacts. The phases with most contacting are highlighted with black arrows.

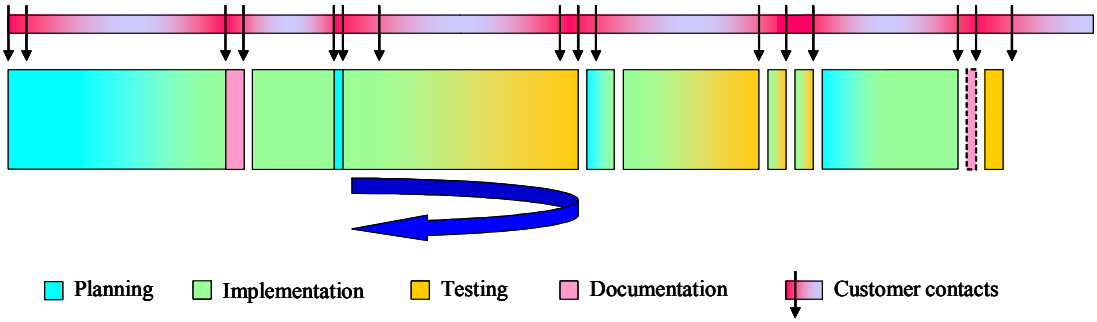

Figure 12. Course of a typical project.

Software development is done in small teams. Typically a project team consists of a project manager and one to three developers. In many projects one developer does more than 50 percent of the implementation. Some small projects can be carried out with one developer alone.

Each project starts with some kind of planning phase: formal or informal, depending on the scope of the project and on how specifically the customer can express their needs. Implementation phase overlaps specification in a way that a clear beginning for the

phase cannot be identified. If any documentation is written, it is usually a semi-technical product specification.

During the project, specifications are clarified and new features or functionalities may be added. Often, software is tested both by the developers and the customer. The customer typically refines the specifications when there is a first prototype to see and test. At this point a new iteration may be needed and the process loops backwards as illustrated in Figure 12 with the blue arrow.

Typically, the implementation phase is rather continuous with only minor interruptions when the specifications need to be clarified. Developers also need to share their time with more than one project at a time, and this may sometimes cause interruptions. The latter half of the project, however, is more fragmented. Getting test results, comments and other feedback from the customer takes some time and the project does not proceed as straightforwardly as in the implementation phase. This is illustrated with small gaps in the project's progress bar.

Customer contacts fade some time after the project accomplishment; however, projects do not have a clear end and minor improvements can be done months after the so-called last delivery. Documentation is not very systematic at the end of the project; a user manual or setup instructions are delivered only if the customer has specifically asked for them.

The process model presented above applies to larger projects with some complexity and specification needs. There are also smaller projects, typically extensions to existing software or minor systems that can be implemented by one developer alone. These projects do not quite follow the above-described model; instead they include little planning, next to no documentation and a straightforward implementation phase that ends with a standard test for functional bugs. These projects do not typically involve much customer contacts other than in the beginning and the end of the project.

Considering the small size of the company and the number of employees, it is natural that the process is still rather informal and unstructured. The company intends, however, to improve project management and to make the development process more controlled at some time frame. Major weaknesses relate to specification and testing, and maintenance of old software is sometimes considered difficult.

Usability in Projects

As was already known when starting the current state study, usability engineering is not systematically practised at the company. Usability as a concept or theory is known and the employees have recognized the need to produce more usable software. However, the desire to make better software has not turned into managed user-centred design practices.

In few projects, potential users and use cases have been thought through. Recently, in an in-house software development project and one large customer project, user interface was designed and documented before implementation. In most cases user interface design does not show up as a separate design phase.

In web projects the layout of the Internet page is designed before functional implementation. Easy navigation and organization of informative contents are given a thought. Especially in web projects, the graphical appearance of the pages and visual effects are also considered important.

Software usability is not tested besides functionalities and software developers were uninformed about available testing methods. User manuals are not written by default unless customers are willing to pay for them.

3.6.3 Current Knowledge and Understanding of Usability

Everyone in the company has heard the term usability before and was able to give some kind of an opinion of factors that affect the usability of a product. In their minds, usability relates both to software and mundane products. Also the basic concept of user interface is understood. Usability as a topic has been somewhat discussed recently, mostly in informal context at coffee table or lunch conversations, but also in some larger projects.

Usability engineering and UE techniques are less known to the personnel. The knowledge varies between the employees, depending on whether they have included some usability in their studies before. An understanding of UE process and practical skills in usability engineering are understandably lacking, though, because usability engineering is not currently practised in software projects. Nonetheless, the employees have learned certain good practices in UI design and know some basic principles that could be applied in the design.

3.6.4 Attitudes towards Usability

A general attitude towards usability among the interviewees is good. Usability is regarded important whenever someone else is going to use the software. Also, experiences of good and bad usability have made the interviewees appreciate ease of use and intuitiveness of a software product. Although lacking a thorough understanding how to build usability into a product, the employees are willing to learn more about usability and how to take it into account in UI design. The employees also show a strong belief that the software produced in the company has rather good usability and that everyone will do their best for good usability within the limits of project resources.

Human-Centeredness Assessment

The company's attitude towards usability was formally assessed using the INUSE human-centeredness scale (Earthy, 1998; see section 3.3.2). Based on the information gathered about the organization's work methods and individual interviewees' attitudes, each management practice was assessed using the four point scale (Earthy, 1998):

N	Not achieved: There is no evidence of achievement of the defined practice.
P	Partially achieved: There is some achievement of the defined practice.
L	Largely achieved: There is significant achievement of the defined practice.
F	Fully achieved: There is full achievement of the defined practice.

The assessment results are summarized in Table 4.

Table 4. Management practices in the company.

Level	Attribute	Management practice	Assess-ment
A: Recognised	A.1 Problem recognition	A.1.1 *Problem recognition.* Management and staff are aware that there is a need to improve aspects of the systems under development concerned with their use.	F
	A.2 Performed processes	A.2.1 *Information collection.* Information is collected which could be used to take account of user requirements.	L
		A.2.2 *Performance of relevant practices.* Practices are performed which could be used to include information about user requirements in the system or service.	L
B: Considered	B.1 Quality in use awareness	B.1.1 *Quality in use training.* Staff are made aware that quality in use is a particular attribute of a system which can be improved.	F
		B.1.2 *Human-centred methods training.* Staff are made aware that quality in use is achieved through the use of a series of human-centred processes during the development and support/use of a system.	P
		B.1.3 *Human-system interaction training.* Staff are made aware that human-centeredness covers the total system, not just the user interface or the physical ergonomics.	P
	B.2 User focus	B.2.1 *User consideration training.* Staff are made aware that the needs of the end users of the system should be considered when developing or supporting the system.	F
		B.2.2 *Context of use training.* Staff are made aware that end users' skills, background and motivation may differ from developers or system support staff.	F
C: Implemented	C.1 User involvement	C.1.1 *Active involvement of users.* The development process ensures understanding of user needs through user involvement in all development phases.	N
		C.1.2 *Elicitation of user experience.* The design solution is shown to stakeholders and they are allowed to perform tasks (or simulated tasks).	N
		C.1.3 *End users define quality-in-use.* Systems are tested using measures of quality in use derived from end users.	N
		C.1.4 *Continuous evaluation.* Early and continual testing is an essential element of the development methodology. The process is based on the necessity for feedback from users.	N
	C.2 Human factors technology	C.2.1 *Provide appropriate human-centred methods.* Select and support methods for the elicitation of user input at all stages in the lifecycle.	N
		C.2.2 *Provide suitable facilities and tools.* Suitable facilities and tools are provided for quality in use activities.	N
		C.2.3 *Maintain quality in use techniques.* Ensure that methods and techniques are reviewed for suitability and that state-of-the-art user interface technologies are used as appropriate in developing new systems.	N
	C.3 Human factors skills	C.3.1 *Decide on required skills.* Identify required competencies and plan how to make these available in order to facilitate multi-disciplinary design solutions.	N
		C.3.2 *Develop appropriate skills.* Development of appropriate skills in human-centred staff either by training or by job experience.	P

Level	Attribute	Management practice	Assess-ment
		C.3.3 *Deploy appropriate staff.* Skilled staff are involved and effective in all stages of development as and when required.	P

At least some effort has been made in all of the practices defined in levels A (recognized) and B (considered). The problem of usability has been recognized in the management, and initiative has been taken to spread information about usability and users' needs in the organization. A full employment has been achieved in the practices Problem recognition (practice A.1) and User focus (B.2).

The process of background information gathering at the beginning of the project was investigated in interviews. The conclusion was that some background information is gathered in most projects to back up technical decision-making and cost estimates. As discussed in section 3.6.2, the focus is often in functional and technical requirements. The smallest projects may be conducted solely based on the instructions and requirements received from a customer organization. This information-gathering process could be extended to include human-centred information-gathering; thus the processes needed for information-gathering (attribute A.2) can be regarded as largely existing.

In level B two practices still need focusing: The importance of usability as a quality attribute has been emphasized in the organization (B.1.1), but the focus has mostly been on user interface elements and interaction in general. The process view in usability engineering (B.1.2) has not been brought out in training, and neither has the training very much emphasized the need to consider user-centeredness at the system-level beyond individual interface components or design patterns (B.1.3). Especially, the contribution of user documentation, training assets, and the product packaging to the overall user-centeredness of the system has not been discussed.

At the level C (implemented) the only achievements relate to UCD skills development. Some effort has been made to train the personnel in usability methods (C.3.2): a practical exercise was used in the first user interface design training in 2003. There are a few people (including the author) in the company with knowledge in usability engineering gained by university studies, yet most of them have no practical experience. These people are, however, typically involved when usability needs to be considered, and thus the practice C.3.3 can be seen as partially achieved.

Based on the above-mentioned assessments, the company can be currently placed at the level A for *recognized* usability. Remarkable part of the practices in level B for *considered* usability is already acknowledged, and the company could probably achieve that level with moderate investment in further training. Considering the small size of the company and the hectic nature of software service business, the company has taken a good start towards user-centred software development.

3.7 Discussion of the Analysis

The main findings from the company's current state analysis were:

There is no established software process model at the company. The schedule and budget for the project are often dictated by the customers. Projects are carried out by small project teams with one or two software developers responsible for most of the implementation. Software is typically developed in increments and the development often involves re-specification and modifications to the software.

Usability is recognized as an important factor in software quality. In practise, usability does not always get enough attention. Especially in the specification phase user interface solutions should be given a thought. Usability engineering techniques are not systematically used in projects, but UI details are sometimes refined for better usability.

Employees are interested in getting more information. The company is willing to invest in usability training, and the general attitude towards further training in usability is good among the personnel. The atmosphere in the company is favourable for usability engineering, if the knowledge can be shared in the organization.

The results of the current state analysis were presented to the company for possible feedback and discussion. This session did not produce criticism against the analysis, and it was concluded that the analysis reflected well the current state of the company.

The data collected during the current state analysis revealed a few issues other than strictly related to usability. Project management, work load, and marketing efforts, among others, were also addressed. Because these problems were outside of the scope of this study, they were not further discussed during this project. All findings were recorded however, and were reported to the company management for later improvements.

3.7.1 Study Methods

The data collected with interviews was comprehensive and covered more than well the research questions about current working processes and employees' knowledge in and attitudes towards usability. Although the responses to individual questions varied a lot depending on the interviewee's job description, the results showed a clear convergence at the end of the interviews, and it can be stated that the data presents very reliably the current state of the company. Generally, all employees' attitudes were constructive and they were interested in developing the working methods.

Although detailed and extensive, the analyzed work lists contained one problem to the research: In five out of six software projects the working hours of the project manager were not included in the work lists due to him working as an outside consultant in those projects. In the company, the project manager is the main responsible for customer contacts and software specification negotiations, and as a consequence, the work lists cannot be considered representative in what it comes to the planning and specification phase and customer contacts. Had the study relied only to the data gathered in work lists, these two fields of software development would have appeared too light relative to total working hours. This deficiency was recognized, and supplementary information was gathered in interviews. The results acquired this way were sufficient for forming an

understanding of the work processes, but unlike in other parts of software process, here the possibly biased interview data could not be fully contrasted with more objective data from work list.

The questionnaire was not gone through with the interviewee, which caused some problems in analyzing the responses. Most importantly, it proved difficult to analyze to what extent the usability engineering methods were familiar to the employees, because the answers were ambiguous. Many respondents were unsure if they knew the meaning of the concept, either because they only knew it in practice, or they had used another term for it earlier. It is clear that some terms carry different meanings in different scholars, and that the same method may have several names. Going through the questions with the interviewee would probably have resulted in less ambiguity, and allowed the employees to explain the terms as they understood them. This problem was identified too late to change the questionnaire, however. The difficulty in analysis was acknowledged in the analysis phase and the concepts definitions of the questionnaire were not used as primary metrics of employees' UCD knowledge.

3.7.2 Data Validity

When considering the results of employees' knowledge in usability and UCD, it must be remembered that one half of the employees had attended to user interface training only two months before the study. The topic may have been more clearly in the minds of these employees, comparing to the ones who had attended to the same training almost one and a half years ago. Moreover, as the employees were informed about this upcoming study beforehand, they may have subconsciously prepared themselves to the interviews by going through the lessons learned in the training. On the other hand, practical exercise was included only in the first training session in 2003, which may have strengthened the effect of the training back then. This would be reflected in the result that all three employees who wrote the user profiles and use cases in paper had attended to the first training where this method was particularly emphasized. In the end, it can be stated that the circumstances were rather neutral, because it would be impossible to reach a situation where all employees would have equal previous knowledge provided by the company. Moreover, the starting point of this study reflects well the instable situation that is constantly present in a company: without continuously on-going supplementary training to old employees, the newly-hired employees will always have received certain information more recently than others.

When reading the INUSE maturity assessment, it is worth noticing that the maturity is assessed based on the management activities in the company. In the case company the development process is not quite managed, and especially in smaller projects the usability initiatives may depend on the individual developer's attitudes and skills. The assessment was based on the general understanding gained from the interviews of the personnel. However, the interviews also showed that the attitude towards usability is good also among the employees; it is not only the management that is enthusiastic about the topic. This partly supports the assessment that usability is recognized in the organization as a whole, although the management activities are not fully consistent in all software projects.

3.8 Redefinition of Study Objectives

After the current state analysis the objectives for the rest of the study were defined.

- How to integrate usability into the current working practices?
    - What is an appropriate user-centred process for the company?
    - Which usability engineering methods are effective and non-costly, and could be used when time and money count?
- Which topics should the training focus on?
    - What are the essentials of UCD that a software engineer should master?
    - Which areas of software development should primarily be improved at the case company?
- Can user-centred design be taught by organizing an intensive training period?

To answer these questions, reference was searched from previous studies. Together with the Chapter 2, chapters 4 and 5 cover the theoretical background of the study. Based on the understanding gained from the literature, a model for usability engineering process was suggested for the case company, and a training programme was organized. Chapter 6 discusses usability engineering at Spellpoint Group, and Chapter 7 describes the experiences from the training programme in detail.

4 USER-CENTRED DESIGN METHODOLOGY AND TECHNIQUES

User-centred design aims at designing and producing a computer system by taking into account the needs and goals of the target users. Whether it is for work or leisure, using a program is always done for a specific reason. Each user has certain needs or goals they are trying to accomplish with the system. In a work environment, software is usually produced to help certain work that is currently done differently – either manually or with an older computerized system somehow insufficient for current requirements. Building a new software product always involves defining or redefining ways to perform certain tasks or procedures (Hackos & Redish, 1998). To support software design and decision-making, data about users and their needs for the system have to be collected.

In the following sections, methods for finding out about users and their needs for the new product are presented. The importance of specifying system requirements is then discussed. A couple of ways to present design solutions are examined, and after that, usability evaluation techniques are discussed.

It has been stated that under limited resources, insisting on using only the best usability methods may lead to using no methods at all (Nielsen, 1993). Many usability methods require an amount of money and certain expertise, which may not always be available. Usability engineering techniques do indeed vary in their needs for resources. In this chapter, attention is paid to techniques that could be used with fewer resources, as well.

To encourage at least some usability engineering to be practised, Nielsen (1993) suggested a set of four simple techniques that he called *discount usability engineering methods*. These include user and task observation, scenarios, simplified thinking aloud, and heuristic evaluation. These methods among others will be further discussed in the next sections.

4.1 User and Task Analysis

When designing software, it is important to understand who will use it. Designers are not the end users, and they cannot always tell what the users want to achieve with the program. The only way to avoid guessing is to do research on the users and their tasks. Early focus on users will guarantee best that user needs are acknowledged in the design process.

A system may have many different users: they can be for example individuals using it either at home or workplace, administrators or maintenance people, or those who install the software (Hackos & Redish, 1998; pp. 28). Different user groups need to be considered when analyzing the users.

4.1.1 User and Task Observation

Observing users working at their workplaces will reveal more details about their work than simply asking them to describe their work. More importantly, all work tasks cannot be described verbally. Visiting user sites is a good way to find out about the working methods with the present system, and may help in designing new solutions.

Why Field Studies?

Traditionally, many systems developers are used to gathering required data about the users and their tasks by organizing a requirements-gathering session with a user-representative (Hackos & Redish, 1998). Also a focus group method adopted from market research studies can be used, in which a group of 8 to 12 users or potential customers gather to discuss their working methods and the functionalities or attributes of a certain product (Hackos & Redish, 1998). Instead of visiting customer locations, these sessions are typically organized away from users' work, under normal meeting room conditions. Hackos and Redish (1998; pp.150–152) list several reasons why field studies provide more and better information.

First, the focus group sessions typically concentrate on functionalities more than purely usability. Starting from functional requirements too quickly leads to defining database elements, algorithms, and other technical solutions, leaving usability requirements ignored. Second, asking people to tell how they do certain things will not produce the same results as observing them doing it. People cannot always describe what they do, or they may oversimplify or generalize their description. Also, some essential actions may be so automated that people forget to mention about them. Third, the best way to solve problems is not always what the users say. In requirements-gathering sessions, the users will tell what they think they need. However, visiting their site locations and watching them work might reveal more innovative and better ways to do the work. People may be seeing only the obvious solution for a single problem and missing the big picture. (Hackos & Redish, 1998)

Two more problems relate to information-gathering outside customer locations. When inviting users to discuss their work, the user representatives eventually coming to the meeting are often managers, supervisors, or other people who think they know how the work is done. They may have done the work before, but the actual daily routines get forgotten soon after moving to other responsibilities. Watching the actual users' work will give a more realistic picture. The final problem relates to the arrangements of the meeting itself. Gathering a bunch of people together may lead to only few people dominating the discussion while others stay quiet. Some people with differing opinions might be unwilling to express themselves in a group situation, either because they do not want to be different, because they do not want to take responsibility of the decision, or they may feel being wrong. On site visits the users get to speak one at a time, and they will be heard as individuals. (Hackos & Redish, 1998)

The benefits of field studies are summarized in Table 5.

Table 5. Benefits of field studies (Hackos & Redish, 1998).

1. Focus on usability and work tasks instead of functionalities and technology.
2. Gathering first-hand information about user's work.
3. Avoiding misinterpretations of users' need.
4. Reaching real, representative users.
5. Avoiding bias caused by group dynamics.

Discount User and Task Observation

Information about users and their tasks can be acquired with many techniques. Video-taping, shadowing, observing, and interviewing users while they are working are all known methods (see e.g. Hackos and Redish, 1998), but the more data is collected the more it takes time to analyze it and extract the relevant information. The discount task analysis presented by Nielsen (1993) includes quietly observing the users as they work and collecting data without interfering. The observation aims at understanding the factors that affect the work and identifying user needs in relation to the system. While observing, notes can be taken about the various kinds of tasks the users perform, the order in which they do them, and the steps there are in accomplishing a task (Hackos & Redish, 1998). It is also interesting to observe how different users' tasks relate to each other and how the responsibility is shared among the users. When listing tasks, it is also important to recognize their frequency, criticality, difficulty and time to complete (Hackos & Redish, 1998). These factors can have a large effect on the requirements for the system.

Wixon et al (2002) present a similar Discount User Observation (DUO) method to reduce the costs and time required for user and task analysis. The DUO method includes collecting detailed notes from observations, asking minimal clarifying questions and recording contextual information using a digital still camera. Asking clarification is useful when analyzing a work task that is difficult to understand only by silently observing, and the user is the best person to tell why he does certain things. Whatever the exact methods used in user and task observation, the goal is to gather sufficient information to be able to understand the conditions where the future system will be used.

4.1.2 User Descriptions

After identifying the users for the system, a representation of them is a useful tool for the designers. Two different approaches to user descriptions are presented: personas and user profiles.

Personas

Alan Cooper (1999) presents *personas* as a tool for describing users. Personas are detailed descriptions of users, created by the designers. They are not real people, but they are synthesized from observations of real people (Cooper, 2003). Personas have a name and age, and several details about their personal life and work. A picture of the persona strengthens the overall image. The more details, the more usable and credible the persona is. Although imaginary, the user characteristics are derived from existing users by combining, simplifying, and approximating. A good persona represents the average, not a marginal user.

The method of creating personas helps in understanding the users and their goals in software development. Cooper proposes that the persona becomes a living person in the minds of the developers and gives perspective to design assumptions. Instead of designing to everybody, personas help in designing for one user and thus reduce the need to include everything in the design. The software developer can ask, for example, "Would Rosemary want this functionality?" and use the imaginary persona as a decision tool. (Cooper, 1999; pp. 165–192)

User Profiles

A resembling approach to user analysis has been presented by Hackos and Redish (1998), who call these representations *user profiles*. What is different from Cooper's personas is that these user profiles are formed from real data collected from observations and interviews of actual people. User profiles can be written to describe a few representative users as exemplars of particular user types or user groups. User profiles could include photographs of users and their working environments, quotes from the users, narrative descriptions, and any other relevant material that would reveal who the users really are. A good way to illustrate the user profiles is a poster. Presenting the user profiles in narrative or visual description, instead of writing a simple characteristics list, will make them more memorable. (Hackos & Redish, 1998; pp. 306–308)

Remarks

The selection between these two user description methods is not very critical; the main reason for using such a description is to make the diverse users more understandable and tangible. One risk can be seen in Cooper's method, that is – overgeneralization. When the personas are imaginary, they may end up reflecting the writers own prejudices and become overly stereotypical.

4.1.3 User's Tasks and Goals

To build a successful user interface, it is necessary to understand the work that the users do and the goals they want to achieve with the program. For background information it is essential to find out how the work is currently done, and what problems the users are having with the current procedures. By examining the current work it is possible to identify ways to simplify the procedures and to help the users accomplish their tasks more effectively (Hackos & Redish, 1998). Automation should never increase work or cause more frustration than the current way of doing things (Beyer & Holtzblatt, 1998).

Analyzing the context of use also ensures that all relevant usability variables are considered when the product is specified (Maguire, 2001). Furthermore, when the context of use has been made clear, it provides a shared view among the members of the design team and facilitates group work (Maguire, 2001).

Different users have different goals and their needs for the system are thus different. Each user types should be considered separately, taking into account their previous experience in using similar systems, and the tasks they need to do at their work. Finding out user's goals has been extensively discussed by Hackos and Redish (1998); the topic cannot be covered here in such detail.

A user can have three different types of goals for using a system: life goals, experience goals and end goals (Cooper, 2003). Life goals represent users' personal aspirations and motivations, and help to understand *why* the user is doing something. Experience goals describe what users want to *feel* when using the product. End goals represent the user's expectations of the tangible outcomes of the interaction with the system. To satisfy user expectations, all these goals need to be considered. (Cooper, 2003)

The work or tasks of a user can be presented in various ways: Larger activities can be seen to consist of consecutive tasks that can be presented as work flows or task sequences. The many different uses for a system can be presented as task lists. By analyzing the relationships between the tasks, they can be found out to form task hierarchies. (Hackos & Redish, 1998). Organizing and categorizing the information will reveal the structure of the work as currently done.

Use Cases

Use cases are a design technique introduced in object-oriented software engineering (OOSE) methodology, presented by Jacobson, Christerson, Jonsson and Övergaard (1992; as cited in Mayhew, 1999). OOSE has a lot to do with usability engineering approach as it also starts from analyzing the real world as a basis for software modelling. Use cases are generalizations from a user's real tasks and they try to capture generalities across individual users doing similar tasks (Mayhew, 1999a). A use case describes a work flow as it is currently done and it is written in every-day language. Use cases present typical usages in abstract form; the instances of use cases observed in real world can include variations, bottlenecks or human errors that are not included in the use case representation (Mayhew, 1999a).

Use cases can be illustrated by using a Unified Modeling Language (UML) notation. UML was generated by Jacobson, Booch, and Rumbaugh (1999), and it is a standard modelling notation that can be used for documenting object-oriented system specifications. Figure 13 below illustrates use cases in UML notation using a simple example of a lift (Schach, 2002). In the example, there are two possible uses for a lift: pressing the lift call button outside the lift or a floor button inside the lift.

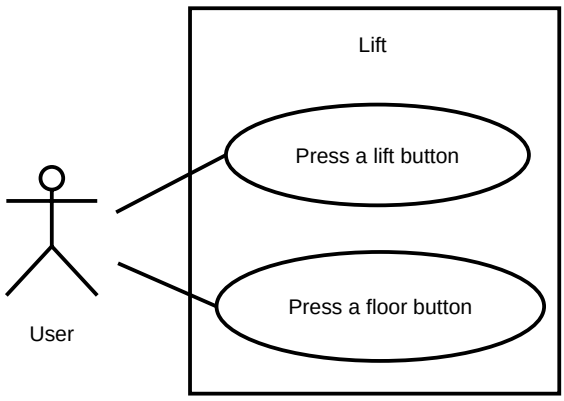

Figure 13. Use cases for a lift. (Schach, 2002; p. 370).

4.2 Requirements Definition

Defining system requirements is not purely a usability engineering method, but a fundamental part of any development process. Requirements definition is about identifying and indicating needs and constraints in relation to the context of use of the system (Jokela, 2002). Although all processes of user-centred design should be driven by business goals, it is of special importance at the requirements phase (Jokela, 2002).

Requirements should address the areas of business requirements, technical requirements, and user requirements (Faulkner, 2000). Business requirements are derived from

analyzing the organization and the users who will eventually use the system (Faulkner, 2000). They can also define the financial and legislative requirements (Jokela, 2002). Technical requirements define what the system must do, what the expected performance level is, and how the system should co-operate with other technical systems in the organization (Faulkner, 2000). The user requirements of a product are derived from the needs and goals of the users identified in user and task analysis (Faulkner, 2000), and are often specified in terms of effectiveness, efficiency, and satisfaction (ISO 9241-11, 1998). The user requirements should integrate all things that have an impact on the total user-experience: in addition to software, also user documentation, training, and packaging of the product should be covered when applicable (Jokela, 2002).

When deciding on the requirements, it pays off giving the various requirements to the customer for prioritization (Schach, 2002). Ranking the requirements by importance allows the development process to focus on the most relevant requirements early in the development and to ensure that they will be met. In case of changes in the initial schedule or other reductions in resources, the functionalities may be cut down starting from the requirements of minor importance. Also here the requirements statement serves as a decision-making tool.

Requirements are an essential part of the contract between customer and deliverer. Once agreed on, the requirements should not be changed without a written agreement between the parties (Faulkner, 2000). A clear statement of the requirements is necessary already from the business perspective, because it mostly defines the scope of the project and allows for estimating the time and the money required. Moreover, it serves the usability objective by guaranteeing that user requirements are not overridden on-the-fly. Also, without systematic control of the requirements, the project may suffer from 'requirements creep' where new requirements appear every now and then (Faulkner, 2000). Changing a requirement without a thorough examination of the consequences, even when asked by the customer, can cause serious problems in the development. Faulkner (2000) reminds, however, that requirements are not static and they most certainly evolve during the project; the requirements can be altered when enough care is taken to control and document the process.

4.3 Prototyping

Prototyping is a method used in iterative design. It means producing a test version of the program that can be used for testing and evaluations. A prototype can be either low-fidelity with little or no functionality, or high-fidelity where most or all features and interaction functionalities have already been implemented. The quality of the prototype is determined by the phase of the design process and the purpose of the prototype, that is, whether the prototype is made for introducing an early version to test users or for testing a fully-designed product for validation. It is recommended to use low-fidelity prototype for testing early designs because they are cheap to produce and easy to edit (Nielsen, 1993).

In this section, three low-fidelity prototyping techniques are presented.

Scenarios

Nielsen (1993) presents scenarios as a discount prototyping method. A scenario is a narrative description of a user performing certain tasks in order to achieve certain outputs. A scenario thus presents the actions taken by the user step by step, from the beginning to the end. A scenario is limited to one path only, and no real data is available. Scenarios are a very limited type of prototypes that can be used as a means to express and understand user interaction with a system (Nielsen, 1993). Faulkner (2000) points out that a system cannot be designed before the developers understand the user's tasks. Scenarios are a good way of conveying this information and putting users and designers on the same wavelength (Faulkner, 2000).

Paper Prototyping

A narrative scenario can be used to construct a prototype for early evaluation of a user interface design and to collect user feedback (Nielsen, 1993). If the design is tested with users, it is useful to implement the scenario in paper. In paper prototyping each screen or view of the program is drawn on separate sheets of paper. A user is asked to perform the task described in the scenario without help by using the prototype: to "click" with their finger and to "type" in text input fields. Conducting a user test with a paper prototype quickly reveals major problems in the design. A paper prototype is a good way to find problems especially in design concepts and terminology, screen layout, and the contents of the program. Paper prototype will also reveal problems in navigation and work flow, and it can be used to measure whether the functionalities satisfy the users' needs. (Snyder, 2003)

With paper prototypes the interaction with the prototype is not very natural and the prototype responds slowly if at all to user's actions (Snyder, 2003). The look and feel of the system is missing, and the prototype usually shows only some of the final functionality (Hackos & Redish, 1998). As an advantage, the task-oriented scenarios are often easier for the users to understand than function-oriented system specifications (Nielsen, 1993), and seeing the design on paper may encourage more suggestions as it seems more changeable (Hackos & Redish, 1998). Hackos and Redish (1998) also point out that testing with a paper prototype can help avoid unrealistic expectations of how soon the real product can be implemented; in contrast, a high-fidelity prototype that responds to user input – even if only restricted to certain paths – can seem quite ready for the users and require further explaining.

Rapid Prototyping

Rapid prototyping means producing a prototype of the user interface by using some visual development tool. It is a useful method when the customer is unsure what they actually want from the system, or when there is a risk that the customer is losing interest. Rapid prototypes allow the user to evaluate the system before it is built, and user feedback can be used in further developing the system. (Faulkner, 2000)

Faulkner (2000) points out that the prototype should only be used to getting the interface right, though. It should never be used as a means to avoid proper engineering. Too often an interface is quickly hacked together as a demonstration and modified according to the users' needs. Then, instead of serving as a model for the system, the system is eventually built using the same prototyping tool that was used in creating the

prototype, or the interface prototype is bolted onto the system's functionality. (Faulkner, 2000)

4.4 Usability Evaluation

Evaluating user interface designs is an essential part of usability engineering and it aims at identifying problems in the design. Usability evaluation can be done with or without end-user involvement. To distinguish between methods involving users and those done by usability specialists or designers only, two different terms are typically used. When users are involved, the term *user testing* is used. To refer to testing or inspections done without user involvement, the term *usability inspection* is used. In some writings such as Nielsen (1993), the term *user testing* is used to refer to a single usability evaluation method, that is, the conventional UI testing with one user (see section 4.4.2). Here the term *usability testing* is used for that specific method.

In this and the next section two different usability evaluation methods are discussed: heuristic evaluation and usability testing. These are the two most common usability evaluation methods (Riihiaho, 2000), and they are also suggested in Nielsen's discount usability methods (Nielsen, 1993). Usability testing is discussed especially from the viewpoint of simplified thinking aloud, according to Nielsen's (1993) discount usability engineering.

4.4.1 Heuristic Evaluation

Heuristic evaluation is a usability inspection method that does not require end-user involvement. In heuristic evaluation the system is reviewed by a small set of evaluators against known design principles, or heuristics. As an output, heuristic evaluation produces a list of usability problems with references to the heuristics that were violated in the opinion of the evaluators. A running system is not required for the evaluation, and the method suits well in use also in the early phases of development where the user interface exists only on paper. (Nielsen, 1993)

Design Heuristics

Many different lists of design guidelines have been produced. For example, Smith and Mosier (1986) have listed almost a thousand guidelines, and Ben Shneiderman (1986) has written eight golden rules for interface design (as cited in Riihiaho, 2000). Molich and Nielsen's (1990) nine heuristics were proposed as an intellectually manageable set of principles (Riihiaho, 2000). According to the writers, these nine principles were more or less recognized in the user interface community, and were easy enough to be presented in a single lecture (Nielsen & Molich, 1990). These heuristics also explain the majority of problems that one can observe in user interfaces, and correspond to the generally agreed-on principles of good design (Nielsen & Molich, 1990).

In Table 6 a list of ten design heuristics is shown, following Nielsen's (1993) presentation. The nine heuristics initially suggested by Molich and Nielsen (1990) are the items 1–9 in the list. Later, Nielsen (1993) had added a tenth heuristic into the list; this heuristic is the rule number 10. Next, each heuristic is shortly examined following Nielsen (1993).

Table 6. Ten design heuristics (Nielsen, 1993).

1. Simple and Natural Dialogue
2. Speak the User's Language
3. Minimize the User's Memory Load
4. Consistency
5. Feedback
6. Clearly Marked Exits
7. Shortcuts
8. Good Error Messages
9. Error Prevention
10. Help and Documentation

1. Simple and Natural Dialogue: User interface should contain only relevant information that the user needs, at the time and place where it is needed. According to Nielsen (1993; pp. 115), "every additional feature – – is one more thing to learn, one more thing to possibly misunderstand, and one more thing to search through when looking for the thing you want." The information should be organized logically, and related objects should be grouped close to each other. The components of the user interface should be organized in such order that the user can perform tasks in the most suitable and effective way.

2. Speak the User's Language: In addition to implementing user interfaces in the users' native language instead of some other foreign language wherever possible, speaking the user's language also refers to terminology used in the UI. The dialogue should always use natural language instead of system-oriented codes (such as numerical codes), and use terms and concepts that are familiar to the target users. Non-standard meanings or uncommon shortenings should be avoided. The main objective should be to use the same concepts and terminology that the users would use; in a professional computer program the terminology could also include the domain-specific, specialized terminology.

3. Minimize the User's Memory Load: In general, people are better in recognizing things than remembering them. Thus computers should try to help users to perform tasks and not require remembering everything. Letting a user select items from a list or browse a menu are examples of user interfaces where user's memory load has been reduced. Also, providing an example of required input next to the field, for example a date field, helps users in filling them in. As an extreme example, users should not be required to read and remember text from one screen and use that information while editing another screen.

4. Consistency: In a good user interface, the same commands or actions will have the same effect everywhere in the system. Also, if there is same information or similar functional components in different screens, they should be on the same place and formatted equally in each screen. Consistent looks and structure of the interface speeds up the usage and reduces possible misinterpretations.

5. Feedback: A system should continuously provide feedback about its state and performance for the user. The feedback should not restrict to error messages; positive feedback is also needed to inform the users that their actions are being received and reacted to. In procedures that take a long time to process, the progress should be

indicated. At all times, the user should be able to tell whether the computer is still processing the information, or if the process has frozen because of a system failure. When considering response times, it is important to remember that too short a response time is also bad usability; if the computer reacts too fast, the user will find it hard to keep up with the feedback.

6. Clearly Marked Exits: Users will always make errors when using a system, and they should be provided with an easy way out of an undesired state. Also, navigating in a system should never lead to a state or view where there is no way out. An undo-functionality allows users to try unknown options without fear of losing their work. Exit requirement also applies to time-taking processes: a user should always be able to abort the execution of a procedure that takes more than some 10 seconds to process. Exits should be clearly visible in the system and they should not require users to remember obscure codes or key combinations.

7. Shortcuts: Providing accelerators for experienced users makes using the system more efficient and faster. Although the system should be as usable for novice users without knowing these shortcuts, they especially facilitate the use of frequently used functionalities. Typical accelerators include function keys, double-clicking features, or direct buttons for important functionalities. Also, having access to recent commands, for example in command-line based UIs, or reusing previous inputs or selections, will make the interaction faster.

8. Good Error Messages: Good error messages present the problem clearly and politely in natural language, are precise and detailed, and constructively help the user in solving the problem (Shneiderman, 1982; as cited in Nielsen, 1993). Guessing what the user actually wanted to do may result in constructive suggestions on how to correct the problem. Error messages should never accuse the user or use hostile language; the user will already feel bad about the error anyway.

9. Error Prevention: Even better than having good error messages is to avoid potential errors. Typically errors are caused by users, and error-prone situations or functions can be found with user testing (see section 4.4.2) or by logging errors when they occur in the field. Designing the user interface in a way that reduces errors, for example favouring selection over typing, will reduce potential errors. Another regular problem is using modes, where a certain action in one mode will cause different output than in another mode. When modes cannot be totally avoided, their state should be clearly shown to the user (Monk, 1986; as cited in Nielsen, 1993).

10. Help and Documentation: Although it is desirable that the system is so easy to use that no additional documentation is needed, this is not always possible. Additionally, users often wish to have documentation to enable them to familiarize to additional, less common features. A good help is readable, compact, and comes with good search functionality. Also, linking help to context and focusing on real tasks will make it more usable.

Later in 1994, Nielsen refined his ten heuristics based on an analysis of 249 usability problems (Nielsen, n.d.). This list is presented in Table 7, with comparison to Nielsen's old heuristics (1993). These heuristics mostly cover the same principles than those initially presented in 1990; mainly the wordings and the order differ. The older set of

design principles is used in this thesis because they were more familiar for the author; the selection of the wording of the principles is solely a matter of opinion.

Table 7. Ten usability heuristics (Nielsen, 1994) and comparison with old heuristics (Nielsen, 1993)

Heuristic	Corresponding heuristic in 1993
1. Visibility of system status	5
2. Match between system and the real world	2
3. User control and freedom	6
4. Consistency and standards	4
5. Error Prevention	9
6. Recognition rather than recall	3
7. Flexibility and efficiency of use	7
8. Aesthetic and minimalist design	1
9. Help users recognize, diagnose, and recover from errors	8
10. Help and Documentation	10

The Evaluation Process

In heuristic evaluation the system is evaluated against the design principles. The method of heuristic evaluation has been described thoroughly by e.g. Nielsen (1993). For best results, three to five evaluators should be used (Nielsen & Molich, 1990; Nielsen, 1993). Each evaluator goes through the user interface several times and compares the elements of the interface with a selected list of usability principles. Each evaluator should perform the evaluation alone to ensure independent and unbiased results. All findings are written down, either by the evaluator or an outside observer who can record verbal evaluations. (Nielsen, 1993)

Typically, an evaluation session for one evaluator can last for one or two hours. Longer sessions are not recommended, and the evaluation of large or very complicated systems could be slit up to several smaller evaluations. (Nielsen, 1993)

After the evaluations, the results are combined in one list. This can be done by the evaluators together, or if an observer has been used, he can quickly combine the results. After that, the evaluators should estimate the severity of the problems. Nielsen (1994; p. 49) suggests a five-point severity scale for the problems:

0 = This is not a usability problem at all
1 = Cosmetic problem only; fix if extra time
2 = Minor usability problem; low priority
3 = Major usability problem; important to fix, high priority
4 = Usability catastrophe; imperative to fix before release

Three factors should be considered when estimating the severity: how often the problem occurs, how hard it is to overcome the problem, and how persistently the problem will occur after the user has come to know about it (Nielsen, 1994).

Remarks

The study by Nielsen and Molich (1990) showed that combining the findings from several evaluations results in finding remarkably more problems. In their studies, the number of usability problems found grew substantially up until to five evaluators. Adding more than ten evaluators did not produce much benefit. On an average over six studies, a single evaluator would find only 35 percent of usability problems (Nielsen, 1993). It has been estimated that five evaluators would find up to two thirds of usability problems (Nielsen and Molich, 1990), and that for best efficiency three to five evaluators should be used (Nielsen and Molich, 1990; Nielsen, 1993). The results of heuristic evaluation will also depend on the expertise of the evaluators. Experienced evaluators will find more problems than novice evaluators, and optimal results will be achieved using double specialists, i.e. usability specialists that also have expertise in the domain of the interface (Nielsen, 1993).

According to Nielsen (1993), heuristic evaluation does not systematically generate fixes to the found problems. However, it is usually fairly easy to come up with an improvement idea, because the evaluation results clearly state the problem and the design principles that are violated (Nielsen, 1993). Nielsen (1993) also states that the solutions are often obvious after the problems have been identified.

Discount Heuristic Evaluation

Nielsen (1993) proposes heuristic evaluation as a discount technique for evaluating a user interface. Although some expertise is required in order to apply the principles correctly in all cases, even non-experts can find many problems. Those remaining problems that were left unrecognized could be revealed by a simplified thinking-aloud test (see section 4.4.2). Conducting the evaluations with more than one person looking at the interface will increase the number of problems found (Nielsen, 1993).

According to Riihiaho (2000), making valuable findings with heuristic evaluation does not require extensive experience or knowledge in usability. In her studies, also beginners proved to be good in finding usability problems as long as they understood that users and the context of use are the most important factors when evaluating a system. The results of evaluations will improve as the evaluators gain more experience in using the method. (Riihiaho, 2000)

4.4.2 Usability Testing

The term *user testing* is used in this paper to refer to all tests that involve users. User testing can be done with one or more users that have not participated in the system design. This section concentrates on presenting the *usability testing* method that is the most commonly used type of user testing (Riihiaho, 2000; Rosenbaum, Rohn & Humberg, 2000; Nielsen, 1993). At first, the standard method is discussed, and at the end of the section, a discount usability testing method is presented, following Nielsen (1993).

Usability Testing Method

Usability testing means letting a user try out the evaluated system or a prototype by performing given tasks with it. It is not about selling the product to the customer or

asking for an opinion; in contrast, usability testing is about finding out how easily people can learn and use the interface (Gould & Lewis, 1985). Users rarely can express their real opinion of the product before they have tried it themselves. If asked for a plain opinion, it is too often based on the fidelity and superficial appearance of the design (Nielsen, 2001).

Usability testing also aims at revealing the user's mental models about the system (Sinkkonen et al., 2002). Mental models are unconscious and non-verbal representations of the reality in human mind, and they are used to explain and anticipate a system's functionality when using a new system (Sinkkonen et al., 2002). The better the user's mental model matches the conceptual model of the system, the easier it is to learn to use it (Sinkkonen et al., 2002). In usability testing, users' mental models are explored asking the user to think aloud, i.e. say what he does, why he does it, and what he expects to happen when he does it, while performing a set of test tasks. This will provide the observers with knowledge about the reasons behind user's actions. Thinking-aloud method can also reveal problems that might otherwise stay unnoticed. This is especially true in situations where the user performs the task perceivably right but has difficulties to understand why the system acted the way it did (Nielsen, 1993). The results from usability tests typically focus on specific problems concerning effectiveness and ease of use, and can often be implemented quickly resulting in immediate improvements (Rosenbaum et al., 2000).

The proper technique and practical tips for usability testing have been discussed by Hansen (1991) and Nielsen (1993), among others. Here the method is shortly presented.

Test Plan

The first step in user testing is to decide what aspects of the user interface will be tested, and selecting the test tasks based on that. The test tasks should cover the most relevant usages of the system and specify precisely what the user needs to achieve. The test situation, test tasks, and the amount of help given should be the same for all test users to ensure similar conditions (Hansen, 1991). Usability testing also usually involves selecting metrics that will be used in assessing product usability (Dumas & Redish, 1994). The metrics may include, for example, the number of users who has had a specific problem, the number of wrong choices made, or the time to perform certain tasks (Dumas & Redish, 1994).

Test users are selected from the target user group of the system. Ideally, the system is tested with various different users that represent different groups of potential users of the system (Nielsen, 1993). Testing the user interface with several users will reveal more problems and help in analyzing their severity. The problems in using the system start clustering around certain features very soon, and they can be identified already with a second or a third test user (Sinkkonen et al., 2002). Before the actual tests, it is strongly recommended to perform one or two pilot tests to try the test arrangements and to verify that the test users understand the task instructions (Nielsen, 1993).

Test Situation

During the test, the experimenter gives test tasks to the user one at a time. The users should be given time to solve the task on their own, and the experimenter should only

intervene if the user seems to be getting desperate. If the user gets stuck in the task, the experimenter can give small hints to help the user on with the task (Nielsen, 1993). The test situation is preferably recorded with a video camera, and by observers taking notes during the test.

The user is asked to think aloud while performing the tasks. Nielsen (1993) makes a few notes about the thinking-aloud method: The method has the disadvantage that it slows down the user's performance, and usually performance measurements should not be done while using thinking-aloud method. On the other hand, while verbalizing their thoughts, users may concentrate more on certain critical components on the interface, and even learn the interface much faster than working silently; this should be taken into account when analyzing the results (Nielsen, 1993).

Usability testing is complemented with an interview after the test where additional information can be acquired from the user. Interviewing the user after the test tasks is a good way to find out about the user's feelings about the system, and to ask for clarifications for certain observations or comments gathered from the tests. The user may also be asked for any suggestions for improvement; although different users may give completely contradictory suggestions that cannot be implemented as such, they can provide interesting ideas to be considered (Nielsen, 1993).

There are some ethical principles that need to be kept in mind when conducting usability tests. The test situation will be quite stressful for the test user, as all their actions are being observed and recorded. The responsibility of the experimenter is to make the user feel as comfortable as possible. Nielsen (1993) has thoroughly discussed the proper way to conduct usability tests; his ethical instructions for usability testing are shown in Table 8.

Table 8. Main ethical considerations for usability tests. (Nielsen, 1993)

Before the test:
Have everything ready before the user shows up.
Emphasize that it is the system that is being tested, not the user.
Acknowledge that the software is new and untested, and may have problems.
Let users know that they can stop at any time.
Explain any recording, keystroke logging, or other monitoring that is used.
Tell the user that the test results will be kept completely confidential.
Make sure that you have answered all the user's questions before proceeding.
During the test:
Try to give the user an early success experience.
Hand out the test tasks one at a time.
Keep a relaxed atmosphere in the test room, serve coffee and/or have breaks.
Avoid disruptions: Close the door and post a sign on it. Disable telephone.
Never indicate in any way that the user is making mistakes or is too slow.
Minimize the number of observers at the test.
Do not allow the user's manager to observe the test.
If necessary, have the experimenter stop the test if it becomes too unpleasant.
After the test:
End by stating that the user has helped you find areas of improvement.
Never report results in such a way that individual users can be identified.
Only show videotapes outside the usability group with the user's permission.

Presenting the Results

After the usability tests, the results are gathered in a test report. If certain usability requirements were set before the test, it should be evaluated whether the requirements were met. The report should list the usability problems found and state their severity. A good way to present the results is to arrange them by severity. In addition to listing the problems, the test team should also suggest improvements on how the problems could be corrected (Riihiaho, 2000). The report should also include positive findings and comments gathered in the tests (Riihiaho, 2000); this way it can be ensured that praised features will not be replaced with worse ones.

Simplified Thinking Aloud

As a discount usability testing method, Nielsen (1993) proposes a simplified thinking-aloud method. It is a lightweight usability test conducted with only a few users. When the system is tested with few users, the test users should be selected to represent an average user. In simplified thinking-aloud method a test user is asked to think out loud while performing a given set of test tasks using the system. As discussed earlier, thinking-aloud method will allow the observer to gain understanding of the reasons *why* the user does what he does, instead of only seeing *what* they are doing. Nielsen states that even computer scientists can apply the method with a minimum of training and still succeed in finding many problems. To further simplify the method, the traditional way of recording the sessions on video tapes and analyzing them later can be replaced with only taking notes of the session. Although sometimes useful, watching and analyzing the videos is time-consuming and expensive. Nielsen suggests that this time is better used in testing the system with more users or running more iterations. (Nielsen, 1993)

4.5 Discussion of the Usability Engineering Techniques

In this chapter, a number of usability engineering methods have been presented, concentrating on lightweight methods that could be fairly easy to bring into use in a small organization, too. The methods cover a user-centred design process as a whole. User and task analysis, requirements definition, prototyping, and usability evaluation methods as upper categories were discussed, and suggestions were made about their use in software development.

Rosenbaum et al. (2000) asked 134 HCI specialists from different-sized companies to score usability engineering methods based on their experiences. They were asked to name the usability methods they had used in their company, and to estimate how big an impact the methods had on the company's strategic usability. Usability testing – whether in or outside a laboratory – was ranked the most effective method. According to the researchers, one of the reasons for the high ranking may lie behind the fact that usability testing lets many team members observe first-hand how users cannot interact with their product. The most used technique was heuristic evaluation, although it was ranked only sixth effective, after usability testing, field studies, and task analysis, among others. According to the researchers, this may be due to the fact that heuristic evaluation is relatively easy to conduct, and fast results can be given also in a tight schedule. Interestingly, the results showed no statistical difference between companies of different size; the same methods were appreciated in all-sized companies (Rosenbaum et al, 2000). In Vredenburg, Mao, Smith, and Carey's (2002) survey the

three most used methods were iterative design, usability evaluation (referring to user testing), and task analysis. These methods were also ranked high in their effectiveness.

Researchers have also compared different usability evaluation methods to see how similar findings they produce. In one study, heuristic evaluation done by expert evaluators found 44 percent of the problems found in user tests (Desurvire, 1994). The same study also showed that the percentage of the problems found with heuristic evaluation was larger for minor problems (80 percent), but for major problems the evaluators' sensibility was worse (only 29 percent). This result proves that inspections cannot replace testing with users, although many problems can be identified using inspections only.

Brooks (1994) suggests inspections to be used before user testing to eliminate major problems. Then the modifications made based on the evaluation can be validated with users. In Brook's opinion, inspections are also a sufficient evaluation method when making decisions about factors that are less important to the users, for example, when choosing between two alternative designs or when choosing a product for in-house use. However, in a competitive product market, usability testing is essential to ensure success. (Brooks, 1994)

Looking at the UE methods presented in this chapter, it can be noted that a number of techniques are used long before any user interface design takes place. The most important activities are really those that take place *before* substantial resources have been committed to any particular design – they involve making the most far-reaching decisions about which functionalities and features to support (Karat & Dayton, 1995). Only later after these decisions come the questions of how to lay out the selected features on the screen. After all, the layout of particular buttons and text fields is of minor importance if the interaction and functional logic of the system have not been designed to support users' work.

5 VIEWPOINTS TO USABILITY TRAINING

Introducing user-centred design in a software development organization is more than teaching people new techniques and work methods. Foremost, it is a question of change management (Mayhew, 1999a). It is of critical importance to understand what motivates change and which factors can inhibit it. This chapter discusses organizational challenges in UCD integration and presents some previous experiences in UCD training. Learning in organizations is also briefly reflected.

5.1 Experiences in Usability Training

There are few reported experiences in training practical UCD skills to software developers. Moreover, research seems to have concentrated mainly on large software companies where usability specialists are often available as a separate resource. A few examples can be found of practical workshop-based training.

Latzina and Rummel (2003) describe a 2-day workshop organized at SAP where a simulation game with four development groups was used to encourage software engineers in co-operation with usability specialists. They state that a short training should focus on social and motivational aspects more that encyclopaedic knowledge. Moreover, in such a large organization cross-disciplinary interaction between different parties in the development process was regarded necessary (Latzina & Rummel, 2003).

Karat and Dayton (1995) report a 3-day consultative workshop in a company of 7000 employees, of which about 50 were usability engineers. In the workshop, four real development teams of up to seven members work on real projects of their own, starting from business process analysis and task analysis all the way to interface design. Each group always includes a real user. In this workshop the development teams get to try the usability engineering methodology in real-life projects with the people who they work with in real teams, which reduces their doubt whether the methods would really work in practice. Although a rather long training, it contributes to the projects in hand and justifies the time spent on it. Also, the learning environment is a good approximation of the software development environment with time pressure. The writers also conclude that the best way to learn the wide discipline of UCD is through interaction rather than technical guidelines. (Karat & Dayton, 1995)

It appears that user-centred design has been more discussed in the context of academic interest (see e.g. Seffah & Adreevskaia, 2003; Rozanski & Haake, 2003; Faulkner & Culwin, 2000) than in individual company level. Moreover, the notion that user-centred design is still quite unknown to small companies (Seffah & Metzker, 2004) is indeed relevant, at least judging by the lack of reports on the topic. Next, organizational learning and training are discussed from a theoretical perspective.

5.2 Learning in Organizations

When planning an organizational training programme, it is good to understand what learning at work actually means. Learning in general can be defined as a relatively permanent change in the learner's knowledge or behaviour (Sinkkonen et al., 2002). This means that the change will appear also after the very activity of learning (Sinkkonen et al., 2002). Organizational learning, or learning at work as discussed here,

means learning that takes place at work environment, either in a specific training event or in daily work. This definition is to distinguish learning at work from traditional school-based learning or learning through leisure activities and hobbies.

Learning easily associates with reading and written knowledge in the minds of many people. In fact, however, up to 70 percent of organizational learning takes place outside formal training events – through stories and gossips or watching other people work (Pfeffer & Sutton, 1999). Nonaka (1994) distinguishes between two types of knowledge: explicit and tacit knowledge. While explicit knowledge refers to codified knowledge that can be expressed in formal, systematic language, and that is easily transmittable (Nonaka, 1994), tacit knowledge is highly personal and difficult to formalize (Nonaka & Konno, 1998). Tacit knowledge is rooted in an individual's experiences and actions, and it also includes personal beliefs, values, and emotions. Tacit knowledge can be seen as consisting of two dimensions: technical and cognitive (Nonaka & Konno, 1998). The technical dimension includes the skills and crafts often referred as know-how; the cognitive dimension consists of the mental models and ideals that are deeply ingrained in human mind (Nonaka & Konno, 1998).

Usability engineering is an empirical science and the practices are learned primarily by doing. The knowledge in usability engineering can be reasonably classified as tacit knowledge, or know-how, that is difficult to verbalize and teach. Although general guidelines and design practices can be given in written form, really learning most techniques such as user and task analysis and usability testing require field experience.

Successful training would produce learning that can be transferred into action at work. Yet, there is a known dilemma that all learning does not affect the ways of doing things; this phenomena is called the "knowing–doing gap" (see e.g. Pfeffer & Sutton, 1999; Tate, 2004). This means that knowing something is important does not automatically mean doing it in practice. A few viewpoints have been presented for tackling with this dilemma.

New knowledge is more likely to be applied in practice if it is clearly relevant to the business and if it has an impact on the learner's job (Tate, 2004). One way to support learning is to use company-specific learning material. Although more expensive and probably more difficult to produce, examples from real work yield better results than generic training material (Tate, 2004). Building connections between earlier and new knowledge is also vital; training should always consider the preliminary knowledge of the audience (Sinkkonen et al., 2002).

The environment can also affect learning results. Instead of learning with strangers, learning with colleagues can strengthen understanding and support behaviour changes back at work (Tate, 2004). Moreover, important knowledge, including technical knowledge, is often transferred through human interaction; rather than sharing facts and statistics, training should concentrate on the underlying philosophy and *doing* rather than just talking (Pfeffer & Sutton, 1999). Pfeffer and Sutton (1999) also remark that the success of most organizational improvement projects depends on implementing what is already known, rather than adopting completely new ways of doing things. In most organizations there is plenty of previous knowledge that needs to be activated and turned into useful work practices.

5.3 Organizational Challenges in UCD Integration

Although there is a wide acceptance of usability principles in system development companies, the majority of development projects are done with little or no usability expertise in house (Maguire, 2000). Even though there was enough knowledge of UCD, integrating it into work practices and design processes can be difficult. This section discusses the challenges in UCD integration. Four major categories for problems in integrating user-centred design can be identified: lack of knowledge, organizational structures, culture and attitudes, and time and cost. These are examined next in detail.

5.3.1 Knowledge and Skills

As pointed out in section 2.1, UCD is a wide-ranging discipline that incorporates various types of knowledge and skills. The structure of UCD and the techniques are still relatively unknown to common developers and to small- and medium-sized companies (Seffah & Metzker, 2004). Mastering usability engineering and being able to apply it in software development projects can be an obstacle to companies with little experience in such activities. In a case study of a small software-intensive product development company of 60 employees, usability engineering was considered theoretical and as such difficult to understand by software engineers (Iivari & Abrahamsson, 2002).

5.3.2 Organizational Structures

Faulkner and Culwin (2000) state that separation of HCI specialists and software engineers leads to an unmanageable situation, where usability engineering is not properly carried out from the start of the process to the very end. Too often, usability is considered as a separate process at the end of the development (Faulkner & Culwin, 2000). Usability activities taking place too late do not have a real influence on the design (Maguire, 2000). Faulkner and Culwin (2000) suggest that software engineers should actually become usability engineers: having both understanding of the principles of software engineering and an appreciation of user needs, these usability engineers can best ensure designs that are both feasible and usable (Faulkner & Culwin, 2000).

Access to real users is the key principle of user-centred design. Yet many organizations have consciously isolated their developers from the customers and centralized the contacts to marketing, customer support, or other specialists (Poltrock & Grudin, 1994). This solution may work when designing systems with little or no human interface, but it can be an obstacle for promoting user involvement in the development process of interactive systems (Poltrock & Grudin, 1994). Opening a communication channel between the users and the developers by bringing the developers in the planning phase would promote user involvement.

Furthermore, in companies where customer support or sales and marketing are separated from the development, feedback may not always reach the developers (Dumas & Redish, 1994). The developers may not know that the program they wrote is difficult to use and thus won't change the way they do things. Moreover, information received from a representative seldom has the same effect as when received directly from a customer (Dumas & Redish, 1994). Also, the tone and actual contents of the feedback and comments may change when told on by intermediaries.

Project management and rewarding systems may also inhibit initiatives in usability engineering. Often the project work is organized in goals or milestones that need to be met, but these goals rarely include goals for usability (Gould, Boies, & Lewis, 1991). Rewarding from meeting the deadlines and making savings in the budget irregardless of usability also fights against UCD (Bloomer & Croft, 1997). Dumas and Redish (1994) point out that product development and maintenance may have different budgets, and this may lead to engineers cutting costs in the development. Being only interested in the bottom line of the development project may lead to ignoring problems in the product that will eventually increase costs later in maintenance and product support (Dumas & Redish, 1994).

5.3.3 Culture and Attitudes

Once in a while one can hear the statement that usability is a matter of opinion rather than an absolute property of any product. Instead of talking in abstract terms of "user-friendliness", usability must be made measurable (Nielsen, 1993). Stating specific goals for usability, such as that a user must be able to install a program in ten minutes (Dumas & Redish, 1994), will quit debates about abstractness and show clearly whether the product is usable or not.

Cultural differences between organizational units may affect their willingness to adopt UCD methodology in their work (Iivari & Abrahamsson, 2002). In many companies there are usability specialists that are responsible for usability considerations of the product. Still, many HCI specialists do not have a technical background and they may not be taken seriously by the software developers, either for lacking understanding of software engineering practices or because they do not speak the same language (e.g. Seffah & Metzker, 2004; Karat & Dayton, 1995).

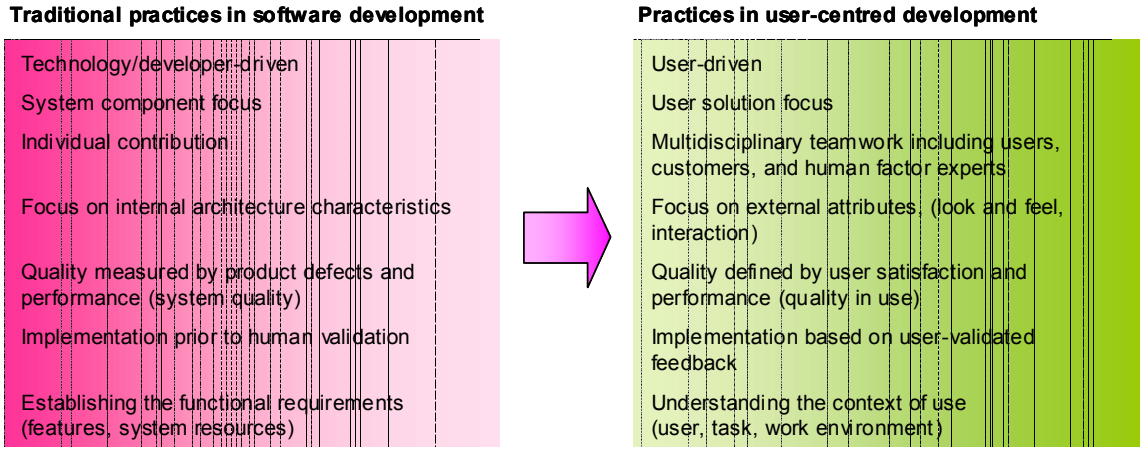

Figure 14. Transition from technology-driven to user-driven development. (Modified from Seffah & Metzker, 2004)

Conservative attitudes may restrain usability and user-centred design from being applied beyond a basic level (Maguire, 2000). Adopting user-centred design in an organization requires a shift in the attitudes of the developers (Seffah & Metzker, 2004). The focus must be changed from traditional software development practices to human-centred practices, where user is put ahead the developer, and where the quality comes from external attributes of interaction and look and feel, instead of the internal

architecture of the system. Figure 14 presents the two approaches. In gradual transition towards user-centred development, the organization needs knowledge about the best practices of UCD.

5.3.4 Cost and Time

Karat and Dayton (1995) state:

> Many companies are coming to recognize usability as an important component of software quality. But often their quest for that quality is not for the right way to do usability engineering, it is for a way that has positive influence on system usability without any offsetting costs of time, money, people, or other resources. They are looking for guidance about how to improve usability, but are unwilling to have the recommendations negatively affect cost and schedule of software product development. (Karat & Dayton, 1995)

The cost of usability is a widely argued question with not just one right answer. Full-scale usability engineering process can indeed seem expensive (see e.g. Mayhew, 2005). The figures must yet be analyzed with caution: the benefits gained from the usability engineering process can overrun the costs in few years (Mayhew, 2005).

Employing usability engineering in a development process does not need to increase development costs remarkably (Nielsen, 1993). When detailed usability testing reports are not needed, testing can be done with moderate resources as a standard part of work and still gain significant improvements.

Another argument against usability is that it adds cost to the beginning of the project and delivers benefits only later (Siegel, 2003). Counter-examples exist, however. There are cases where employing UCD methodologies in the project specification phase actually simplified the process by refocusing the product concept to better target the users' needs (Siegel, 2003). It is still relevant to argue that often usability engineering can extend time-to-market with the first release; yet, time-to-market of the first truly usable product may shorten (Bias, 2005).

5.3.5 How to Overcome the Obstacles?

Integrating user-centred design into a development organization's work inevitably requires some rearrangements in current practices. Although acknowledging the challenges in the way, they should not be seen as insuperable obstacles. Instead, the organization should focus on clarifying their goals and needs for the process improvement.

One of the critical factors in introducing and deploying UCD in a company is the articulation of core principles of UCD (Rosenbaum et al., 2002). A common excuse for not having integrated UCD into work processes is that there are no agreed-on elements that can be measured and thus managed (Rosenbaum et al., 2002). User-centred design must become a strategic choice and a systematic goal for development actions.

A mere intent to employ UCD is not enough without skilled usability engineers. Basic skills and knowledge can be distributed throughout the organization, but to be able to gather and analyze user data and perform usability testing, the organization will need

employees specialized in usability engineering. This knowledge can be acquired from outside the organization, or alternatively, in-house training can be used to train employees in UCD.

Usability engineering and software engineering may not always meet in practice, and the organization must be able to make some trade-offs. Time pressure and costs do matter to some extent, and iterations and testing cannot go on forever. At some point, iterating must end and the product needs to be released in use. Nielsen (1993) also reminds that system design may sometimes violate usability requirements because of other more important requirements. For example, security considerations might override usability in certain situations. When all usability requirements cannot be achieved, the dilemma should be solved by deciding which attributes are most important in the project in question.

6 USABILITY ENGINEERING PROCESS FOR CASE COMPANY

Usability is a topic that is known to inspire also criticism and debate. From merely a business point-of-view, investing in usability might not seem as important in tailored-software business as in businesses targeting consumer markets. After all, the users are known in advance, and the customer has already agreed to pay for the software. In consumer business a failure to meet user needs may lead to a total failure of the product, but in custom-made software services the customer is bound to the product before the development actually starts. Also, the convention of bidding projects can discourage usability: adding costs of usability testing or user and task analysis in the bid may raise the total price higher than that of those competitors that do not include usability engineering in the development process. One part of the dilemma is thus to convince the paying customer for the value of usability.

Yet, failing to consider usability in the development process will most certainly backfire later. A program with bad usability will cause update requests from the customer and require extra training and user documentation. It would be short-sighted to state that these supplementary updates after the delivery are bringing extra profit for the vendor company; exceeding the initial estimate for total costs of the project will guarantee an unhappy customer. Instead, good usability and well-completed project will encourage the customer to consider the software company in further projects as well.

Chapter 2 presented the principles of user-centred design, and various usability engineering methods were discussed in Chapter 4. UCD was characterized as a multi-disciplinary approach to software design that is iterative in nature. The core of UCD is early customer involvement and continuous improvement through design prototyping and testing. Usability engineering techniques cover user and task analysis, prototyping, usability inspections, and user testing, among others.

The previous chapter suggested that user-centred design is not very widely practised in small companies, and presented ways to enhance learning in organizations. Some challenges and considerations regarding UCD integration into work practices were pointed out: four key challenges identified were knowledge, organizational structures, attitudes, and cost and time.

Spellpoint Group, the case company of this study, is a small-size software development organization that currently employs 18 people. Their interest in usability has led to examining the possibilities of deploying user-centred design in their work. This chapter will present suggestions for Spellpoint in implementing user-centred design in their working methods.

6.1 No Time for Usability?

Looking at the usability engineering techniques presented in Chapter 4, a few remarks can be made. Two of the usability engineering methods, namely personas and use cases, have been discussed earlier at Spellpoint Group. Conceptually, new methods for the company are prototyping, heuristic evaluation, and user testing. As discussed in the analysis of the current state, there is no established usability engineering process for

software development, and paying attention to usability has mostly been in the hands of individual developers.

In the interviews conducted for the current state analysis, several members of the company personnel presented their doubt whether there really is time for usability engineering in projects. Delivery schedules are often tight, and customers who are billed on an hourly basis are not willing to pay for "extra" hours spent in usability. Usability engineering was regarded time-consuming and difficult, and selling it to customers was seen as a challenge.

What it comes to selling usability to customers, one important thing should be kept in mind: Usability is an essential part of the product quality and as such it should never be completely ignored. Nielsen (2003) reminds that usability engineering should not be charged as extra work, but instead the costs should be included in the total price of the software. Just like you do not deliver a product without functional testing, you should not deliver it without any thought of usability.

What is characteristic of Spellpoint Group is that many projects are done to a same customer. After several years of working together, the vendor company gets to know the business environment and working habits at the customer. The users are already known when an order for new software arrives. This pre-existing knowledge of the customer is already utilized in product development and it should be taken advantage of in the future, too.

Another important characteristic is that the employees work in a shared work space that makes it easy to communicate and collaborate with each other. This can be put to use in the field of usability engineering by asking co-workers for opinions, evaluations, and help when needed. As this is already done in many projects, it will not require big changes in working habits.

When there is little time for usability, the author suggests five rules of thumb: 1) Specify goals and context, 2) Prioritize, 3) Design on paper first, 4) Remember design heuristics, and 5) Ask feedback before coding. These rules are easy to integrate into the company's present working methods, and are likely to help in software development. Table 9 summarizes the rules, and each rule is discussed more in detail below.

Table 9. Five rules of thumb for better usability.

1. Specify goals and context
2. Prioritize
3. Design on paper first
4. Remember design heuristics
5. Ask feedback before coding

1. **Specify goals and context.** At the beginning of a software project, the intended users and the characteristics of their work environment are specified. Most important usages are identified for the future system. All these data are based on information gathered from the customer, either by observation or – most likely – by interviewing relevant contacts at the customer. The context of use is then made known to all employees working for the project. The descriptions can be included in

the initial specification document, if one is written. Putting them on paper will help to keep them in mind while working. In minor projects with very little work this can consist of a short listing with only a few bullets – just to help the developers understand what the project is all about. Understanding the context of use helps to understand the user requirements for the system and provides background information for prioritizing the requirements. (See section 4.1 for details on context of use.)

2. **Prioritize.** When there are few resources and little time available, it is important to know how to prioritize design efforts between various features. Selecting the most important functions and user interface views based on the understanding of the use context, design efforts can be focused on those functionalities that are most used and that affect most the user experience and the interaction with the system. When working under time pressure, making the daily use of the system as fluent as possible will enhance product quality more than focusing on less-often used features of the system. Attention should be paid to natural order of use and to linking between consecutive functions and tasks. The exact placement of buttons in millimetres is of minor importance.

3. **Design on paper first.** It is recommended to use low-fidelity tools for early designs until all major functionalities and the scope of the project have been agreed on with the customer. This helps in avoiding wasted work if the customer changes their minds after seeing the design prototype. An early prototype can be drawn on paper or produced with a visual tool for software prototyping. Care should be taken when a visual development tool is used, though: the code of the interface should always be rewritten for the final program if it does not conform to good coding habits. Using low-fidelity prototyping for early designs is a fast and easy way, and it will at best shorten the total development time by cutting extra work required to refine the design. (See section 4.3 for details on prototyping.)

4. **Remember design heuristics.** Design heuristics can be used as a checklist to assist in user interface design. The software developers should keep the principles in mind and try to obey them whenever possible. For better results, a colleague who has not participated in the interface design could be asked to evaluate any major user interface in the product when possible. Checking the designs against known usability principles will eliminate major problems from the interface and cut down further modification requests caused by bad usability. (See section 4.4.1 for details on design principles.)

5. **Ask feedback before coding.** Even if there is not a possibility to conduct formal usability tests with real users, early designs should nevertheless be shown to the customer. The most important use cases should be gone through with the customer first, and only then can the program code be written to implement the functionalities. In minor projects with little contribution to user interface design, it is also possible to use a work colleague who has not participated in the design as a test user. Testing will reveal major problems in the intuitiveness and performance logic of the system and help correcting problems before delivering the final program to the customer. (See section 4.4.2 for details on usability testing.)

In the author's opinion, these five rules can quite easily be integrated in any project without remarkably adding work hours or costs to the total project. The major changes in current working methods will be an increased focus on the planning phase and presentations of designs to the customer already when there is nothing functional yet to show. These improvements will however guarantee a better match between the customer's expectations and the final delivery, and eliminate the need to change the design when most of the program code has already been written.

In close collaboration and consecutive projects done with the same customer, especially when engaging in software development at customer premises, the client organization and the users become known. In such cases, the development team does not always need to perform user and task analysis as a separate procedure. When the customer is already known, it is easier to design software that fulfils the users' needs. The development team must still be cautious in these situations: although working with the same customer or client, the end users of the system may vary between projects. It is always essential to find out who are the real people that will use the product – they rarely are the business management or system architects that may be involved in writing the specifications and the system requirements.

So far, software development at Spellpoint Group has already been iterative in nature. After initial prototypes, both the technical functionalities and the user interface have often been redefined and refined with the customer. Iterations have gone on until the customer is happy with the final product. By systemizing the software development procedure, the number of these iterations can even be cut down without violating the main principle of UCD, iterative design. When first iterations are done with low-fidelity prototypes – on paper or by using software mock-ups – less iteration is needed with the functional version that is tested on the field (so called "beta test"). As discussed in the introductory section 1.2, the cost of making changes increases as more design details are bound.

The last two recommendations address UI design and testing. Here it should be remarked that nothing compares to the feedback and first-hand opinions of the end users. Whenever there appears a possibility to show the designs to end users, be it in customer meetings or site visits, it is strongly recommended. Yet, some testing is better than no testing at all, and as the company is only taking the first steps into usability engineering, formal usability evaluation methods have not been emphasized here. Showing the designs to another designer with some understanding of usability will also produce design ideas or suggestions for improvement that would not have been found if the design work was done by one individual only.

6.2 Usability Engineering Process

One of the objectives of this study was to construct a suitable usability engineering process for the case company to be applied in major projects if possible. Building on the suggestions presented in the previous section, a process model was constructed. Figure 15 shows a representation of the UCD process model with appropriate usability engineering techniques suggested for each phase. The model considers earlier usability training in the company and builds on that pre-existing knowledge. The model has been inspired by the ISO 13407 standard process model (see section 2.3.2) and Timo Jokela's process model (see section 2.3.3).

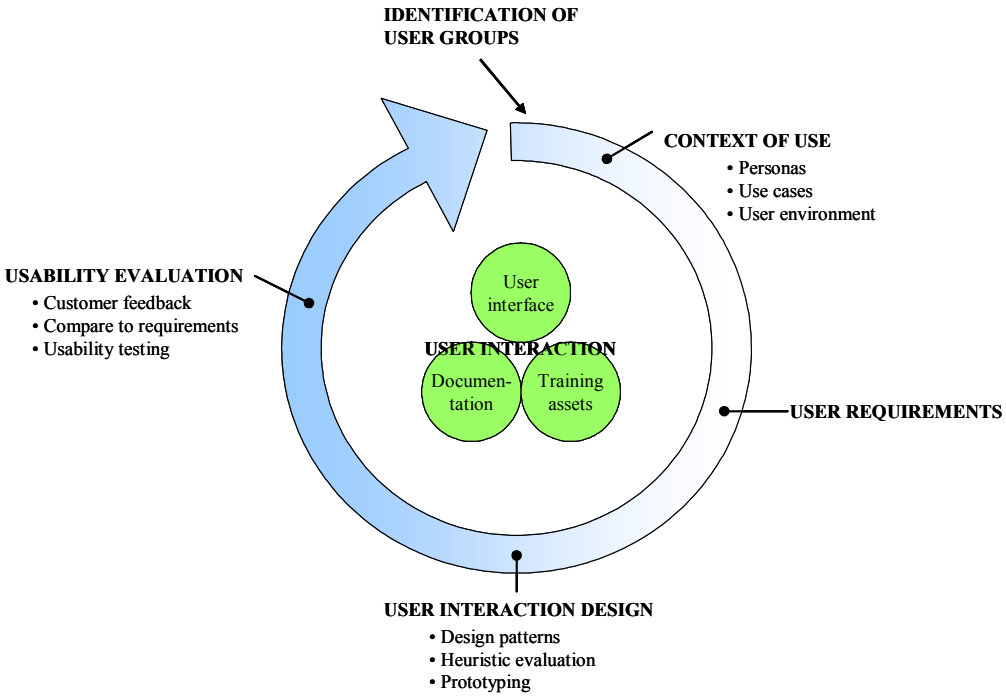

Figure 15. Suggested usability engineering process for Spellpoint.

A usability engineering process starts with identifying the future users for the system. For each user or group of users, the use context is defined. The necessary information is gathered from the customer to back up the description. The context of use can be described using personas and use cases as design tools – the techniques being already familiar to the company. Also, the user environment characteristics need to be considered. The context of use information is applied to form user requirements for the system. A written document is strongly suggested at this point, listing clearly the scope and requirements for the project; this will support the design specification and technical solutions that eventually need to be based on the requirements.

Based on the understanding of use context and user requirements, the user interaction with the system is designed. In addition to user interface, user documentation and possible training needs are considered here. The interface is designed to support the user tasks and to provide a natural way of performing them. Design patterns, as discussed in the company's previous usability training, and heuristics can be used as tools for user interface design. Informal or formal heuristic evaluation can be used to verify the design and look for major problems in it. Being an iterative process, the design is first implemented in low-fidelity prototypes and only on later iteration rounds a functional prototype can be used.

The usability of the design is then evaluated to decide whether it fulfils the requirements and needs specified for the system. The evaluation is done based on customer feedback, comparison with predefined requirements and – if possible – usability testing. The iteration is continued until the product meets its requirements and can be released to the customer.

Usability processes should always drive the development. Customer requirements and user needs must be acknowledged in the planning phase, and the work flow using the

system needs to be designed to support the intended use cases for the system. Technical implementation of the user interface functions will be done only after the design has been verified with evaluations in co-operation with the customer.

As the UCD process is still only gaining ground in the company, the formality of the usability engineering techniques does not need to be the primary concern in the design process at first. More important is the shift in the attitudes and priorities as demonstrated in the chart in Figure 14 (page 67). User needs and goals should always be the primary argument when deciding on certain functions and features, and actions should be taken towards including users in design process through prototyping, demonstrations, and feedback collection. In major development projects a developer with knowledge in usability should oversee the usability process whenever such expertise is available.

When the process starts to formalize and the company has gained some experience in the user-centred design approach, it is time to start thinking about formal methodology. At that point, usability testing can be a good start to proving the value of usability techniques, as it quickly provides visible results and has been reported having a strong impact on the development teams' attitudes (e.g. Mayhew, 1999a; Rosenbaum et al., 2000). Trying usability testing first in some larger project would provide the company with understanding of the benefits and help to direct the design process later towards real user involvement beyond paper concepts.

7 USABILITY TRAINING AT CASE COMPANY

Based on the literature review and the findings from the current state analysis conducted at the case company, a UCD training programme was organized for the personnel of Spellpoint Group. This chapter reports the implementation of the training. First, the objectives for the training are summarized, and the selected training methods are presented. After that, the practical arrangements are described with the results gained through effect measurement. The chapter ends with discussion of the results and the experiences from the training, and presents a statement of the general felicity of the training as a whole.

7.1 Training Schedule and Objectives

7.1.1 Schedule

The time resources for the training were set by the company based on the time allocated for this project. It was agreed that the training would be organized in two parts, and each session was reserved half a work day. Thus the total time for the training programme was 8 hours, including breaks. The training sessions were scheduled one week apart.

The training was organized at work time, and everyone in the organization was invited. Again, it was considered important to spread the knowledge to everybody, no matter what their job description was. Because of the scope of the project, however, the training concentrated mostly on practical software development.

7.1.2 Training Objectives

The current state analysis of working methods at Spellpoint Group revealed several problems that repeat in typical customer projects. The projects are carried out under tight schedules, and the implementation of the software is often started well before all requirements have been identified. A project may start with rather vague specifications that are altered and refined after the first functional prototype has been delivered to the customer. It was concluded that the customers are sometimes unable to tell what they actually want until they have something concrete to see and try out. The usability engineering model constructed for the company (see Chapter 6) addressed these problems and aimed at improving the software development process by eliminating typical pitfalls. Based on these suggestions, the focus for the company training was also selected. As there was only a limited time available for the training, the most critical points of software development were targeted. The main message of the training was summarized as following:

Effort in Specification and planning phase: Context of use, user requirements, and user interface should be carefully examined and designed whenever possible. Even in tight schedule these should not be ignored. Putting more effort in specification can reduce extra work in later phases of the development process.

Low-Fidelity Prototyping and Evaluation of Programs: Low-fidelity prototyping is rapid and cheap, and it can be used to validate early designs. Using usability methods for UI evaluation will eliminate worst usability problems in the program.

Early customer involvement: In addition to specification phase, the customer should participate in the design phase. Low-fidelity prototypes of early UI designs should be shown to the customer, instead of working demo versions of the program. Paying attention to the customer in early phases of the project will lead to a more managed and planned implementation, and reduce the need to alter the design later.

Personal Motivation of Software Engineers: While there is no established process model in use and when project management in small projects is mostly in the hands of the individual software developers, their personal motivation will play an important role in user-centred approach to software development.

Two more themes that the author wanted to emphasize in the training were used-centred design techniques and the wide concept of user–system interaction. This was partially influenced by the assessment for the company's usability maturity (see section 3.6.4) that pointed those two topics as the least considered in the company. Addressing these topics was important for the employees to get a better understanding of usability and usability engineering.

It is important to understand that software engineers will not become UCD specialists in only two half-days' of training. As summarized in section 2.1, to become a UCD specialist requires a large amount of both theoretical knowledge and practical skills. Moreover, every software developer cannot know and does not even need to know everything about user-centred design. Thus the mission of the training was to spread general information, affect the attitudes, and give the employees some hands-on experience in usability methods. A personal insight was regarded more important than a load of theoretical information. An enjoyable and good-humoured event would create a more lasting effect than a single academic lecture, and this was supported by the facts identified in the literature review, too (see section 5.2 for organizational learning).

7.2 Work Methods and Contents of the Training

The training was decided to consist of an introductory lecture and several interactive exercises and discussions. The next sections describe the selected study methods for the training.

7.2.1 Introductory Lecture

Because some basic knowledge needed to be taught to the employees, a traditional lecture was decided to be held at the first training session. From the lightweight usability methods discussed in Chapter 4, heuristic evaluation, paper prototyping, and thinking aloud (usability testing) were selected as new methods to be introduced. Also, a quick presentation about user-centred software process as a whole was to be introduced, following Jokela's (2002) process model. This would better connect the methods to the various phases of a project and give an understanding of how the methods can be used together to create an iterative improvement cycle. Moreover, this allowed reviewing the "old and familiar" methods introduced in the company's previous user interface training, i.e. user profiles and use cases, and combining them with the new methods. This was important for not to confuse the employees with new information without providing enough linking to the teachings from before.

One goal for the training was to give practical hints and tips to help the software engineers in their current work. To support that goal, several real examples of good and bad design solutions were included in the presentation of Nielsen's (1993; see section 4.4.1) ten design heuristics. Examples were selected mostly from Internet services, and some bad error messages were captured in screen shots. Real life examples helped to link the heuristics in practice and showed how to apply them in analysis.

7.2.2 Hands-On Training

Evidence from the interviews held for the current state analysis showed that those employees who had participated in the first user interface training at Spellpoint and done the design exercise then, could better recall the techniques. Hands-on training was also brought up in the wishes of the interviewees. It was a general opinion that learning by doing is probably the best method to acquire practical skills and thus the training was focused on practical doing rather than pure lecturing. Both workshops were to include exercises where the methods presented in the opening lecture could be tried in practise.

The first workshop would give the participants a chance to rehearse heuristic evaluation. First, a user interface would be evaluated without any instructions. After a lecture about heuristic evaluation, the UI was to be evaluated based on Nielsen's heuristics. After first analyzing an interface alone, the findings of each evaluator would be combined and discussed together. This would most likely also prove that more evaluators find more usability problems than one alone.

The interface to be evaluated was a Finnish library database service (HelMet catalogue[19]) where the library items from Helsinki capital area can be searched, browsed and reserved. It was selected as the evaluation target because the service was easy to approach and understand, because it did not require any specific domain expertise, and because the interface was considered imperfect to the extent that examples of poor user interface design were to be found easily. Picking up too good a design would not have been a fruitful example for novice evaluators as problems would have been more difficult to find. The goal of the exercise was to teach the meaning and usage of the heuristics, and the poorer design the better for this purpose. The service was also regarded very similar to many web-based applications that Spellpoint has expertise in, and was a good example because the employees might participate in designing a similar system in real world.

The second workshop would be a one big exercise on its own; no lectures were planned for that training session. The four-and-half-hour workshop would introduce a user-centred software design process in miniature. The participants would work in four groups, each having their own assignment to work on. The workshop would cover the techniques of user profiling, use cases, paper prototyping, and usability testing with a visiting test user from another group. The main goal of the exercise would be to practise user-centred techniques and gain personal experience and again, hopefully, realization of the benefits of usability engineering in a software development project.

---

[19] HelMet catalogue. Espoo, Helsinki, Kauniainen and Vantaa city libraries. Available at: http://www.helmet.fi/

The author was well aware that the time available for the second workshop was rather limited. A concern was raised whether the groups would have time to plan the prototype in such a short time to the extent that it could be tested. It was also critically discussed how the usability testing will work when there are no usability specialists involved to manage the tests. However, the workshop was decided to be held as planned, as it was argued that the personal insight gained in the experiment would probably weigh more than doing everything by the book. Moreover, the workshop would provide the participants with a whole picture of one design iteration cycle, and essential parts of the iteration would have been left outside if testing was not done.

7.2.3 Presentations

It was hoped that the training would raise discussion and questions that could be further examined together in the workshops. At the end of the second workshop, each group would present their design solution and main findings from the user test, and tell what they had learned during the exercise. Sharing the experiences with other groups would reveal similarities and differences between their findings and provide potentially fruitful dialogue between the employees. As there was not that much time available, the more personal experience the training could provide the more effective it would probably be. The assessment of the effectiveness of the training would take place after the training, and it is discussed further in the next section.

7.3 Effect Measurement Methods

To measure the effect of the training, a set of questions were listed that would help to assess the effect. The questions were:

- Do participants feel that they learned something in the workshops?
- Do they think they can apply their new knowledge at work?
- How does learning show up in practice: Will the participants utilize user profiles or use cases in the planning phase? Will they use design heuristics or other principles?

In addition, general feedback of the training as a whole was requested.

The main methods selected for assessing the effectiveness of the training were a feedback questionnaire and a design exercise. Also the UI evaluations done twice in the first workshop would be compared to see whether the results differ. General feedback and participants' own estimates of the effectiveness could be gathered with the questionnaire. The design exercise would be done the same way as it was done the last time, allowing for comparisons between the two designs.

The questionnaire consisted of four pages with both numerical assessment questions with 1 to 5 scale, and more importantly, open-ended questions for explaining the numerical answers and providing qualitative data about the participants' feelings after the training. Questions were presented about the overall success of the training, and the employees were asked what they found useful or useless, and whether they feel that the training enhanced their abilities to consider usability aspects in their work. The questionnaire is shown in Appendix G. Open-ended questions were preferred in this questionnaire because the closed-ended questions in the current state analysis did not

produce quite enough material for analysis. As a consequence, the questionnaire got relatively long. This was not considered a big issue, however, because filling in the questionnaire was done during working hours.

Employees' own assessments and estimates about the effectiveness of the training were important and they were given a high priority, because in this project it was not possible to observe whether the training would really have an impact on the usability of software produced in the future; it would have required more time between the training and the assessment to show actual improvements.

To get some data about the effectiveness of the training unrelated to the employees' opinions, the design exercise tried in the current state analysis stage was repeated. It would provide a clear indication of improvement if the designs were better when redone. The two designs from each employee were to be compared together, and all designs would be judged based on the novelty of the designs and the reasoning behind the design. It was considered too time-consuming to perform another interview round to gather information about the designs and to collect feedback – thus the design exercise was complemented with a short questionnaire with five open-ended questions asking the employee to point out his reasoning and design principles in the exercise. This was done in the hopes of gaining additional information about the design process and the reasons for implementing certain features. It also allowed the designer to explain potential new features or specific parts of the design.

7.4 Training Arrangements

The workshops were organized in May 2005. In this section the arrangements and the course of the workshops are discussed. The schedule for the workshops was planned in advance and is shown in Appendix E.

7.4.1 First Workshop

The first training was organized in May 11th beginning at 8 am. There were 13 people present in the workshop – both management and employees – two of which were new employees who had not participated in the current state analysis stage. They had, nevertheless, been given the same introductory lecture about interaction design by the company as all other employees, and were thus theoretically at the same level with the others. Each participant was given a folder with the presentation slides and evaluation material for the exercises.

The session started with the library database evaluation exercise. The user interface was first thoroughly presented and each participant was asked to list problems in the interface on an evaluation sheet. Screenshots of the interface had been printed in colour for everyone and were included in the given material. This exercise was done in 15 minutes and the evaluations were then collected for later analysis.

The lecture was held in two parts with a small break in the middle. The overall duration of the lecture was 100 minutes. The lecture started with short introduction to usability as a concept, for refreshing the memory, and a couple of reasons for paying attention to usability were pointed out from the software business perspective. The major part of the first half was then used for introducing Nielsen's (1993) ten design heuristics. The

heuristics raised some discussion with the audience, and time was taken to answer to participants' questions.

The second half of the lecture started with a discussion about prototyping. It was stressed out that early designs will always somewhat change in testing. Paper prototypes were suggested as a good way to present ideas to customers, and everyone was reminded that usability evaluation can be done to very early designs, too. Heuristic evaluation was presented as an evaluation method at this point, and usability testing was introduced as an evaluation method that involves real users. The author shared some of her own experiences from usability testing and reminded of the valuable information that can be acquired from real users. At the end of the lecture, the user-centred design process was shortly presented as discussed in section 2.3.3, with suitable UCD methods suggested for each phase. Because of the tight schedule, this part of the lecture was covered relatively shortly; the process model was also considered rather theoretical and was not given too high a priority in the presentation. The latter half of the lecture did not evoke as much discussion as the first half; this may be due to contents that were more theoretical and new to many listeners.

After the lecture, the evaluation exercise was done again, this time namely heuristic evaluation, using the design heuristics presented in the lecture. A new evaluation form was provided for this exercise, this time with a table with columns for the description of the problem, violated heuristics, and the severity assessment. Up to 40 minutes was reserved for heuristic evaluation to guarantee that each employee would have enough time to go through the whole interface without a feeling of time pressure. There was no need to introduce the system again and the evaluators could start going through the user interface right from the start. The author served as an assistant to the evaluator group that was encouraged to ask if there were any unclear parts in the interface. The evaluators presented several questions about the functionality of the service and the functionalities were tried out together to find out the answers. The author performed the test tasks with a laptop and a projector so that everybody could see what happens. Rather technical functionalities were examined also, for example, filling in invalid search terms and trying to generate an error situation. Some ambiguous links and functionalities that were not fully included in the screenshots were also tried. The evaluators seemed to have a good time and the 40 minutes time was generally considered sufficient for the evaluation.

After the individual evaluations the results were discussed together. The initial purpose was to gather the findings on a flap board with severity ratings agreed on together. However, the discussion soon got a little disorganized because the evaluators had a lot of comments about the interface and the problems were found to be very severe. It also seemed that the participants found it difficult to state how severe the problems really were. The time ran out quite soon and the discussion had to be ended before all problems were gone through together. The evaluation forms were again collected for later analysis.

At the end of the training a homework exercise was distributed. The homework was to perform a heuristic evaluation for a user interface design of one software product currently under work at the company. Four screenshots from the system were provided in printing together with empty evaluation forms and the list of the ten heuristics for instruction. The employees were asked to submit their homework in the next workshop.

The purpose of the homework was two-fold: first, the employees could rehearse the heuristic evaluation again. More importantly, the results of the evaluation could be utilized later in the development of the system.

7.4.2 Second Workshop

The second workshop was organized one week after the first one, and the same 13 participants were present at this time, too. This workshop's agenda was to rehearse both familiar and new usability engineering methods in a miniature software development process simulation where one iteration cycle was implemented. Each group had two hours to plan, design, and test a small software product. They first identified the users for the system and wrote profiles for them. Then they collected the use cases for the system and designed the user interface based on them. The paper prototype of the design was then tested with a test user from another group. The test tasks for the user test were chosen from the use cases identified and the groups were asked to prepare a few interview questions to be presented to the test user after the test. Finally, the findings and experiences were shared with the other groups in 15-minute presentations at the end of the workshop.

The group division had been prepared in advance, dividing the participants into four groups: one group of four members and three groups of three members. The groups were planned so that the administrative personnel and software developers would be as evenly distributed in the groups as possible, and that there would be both experienced and novice employees in all groups. To support group work in the workshop, two small cabinets were reserved in addition to a larger conference room that could be divided in two; this way each group would get an individual working space without extra disturbance.

At the beginning of the workshop, the group division was presented and a short instruction was given by the author about the agenda and schedule of the workshop. It was stressed out that the purpose of the workshop was to get an understanding of the process as a whole. As the schedule was very tight, the groups were asked to control their time.

Each group received their instructions and took their rooms. The instructions consisted of a written description of the system to be designed, the schedule, and instructions for user testing. The rotation of test users was also written down in the instruction so that each group could see who will be their test user. The group was allowed to choose one test user among them to go test another group's design; the rest would divide the roles of a test experimenter and observers.

Instructions for the design were different for each group. This way the test users would be unfamiliar with the system, and the presentations would be more interesting. Although imaginary, the software products in the instructions were selected to reflect typical software tools that the software developers would work on in their real work as well. At that moment there were no suitable-sized projects at Spellpoint that would have benefited from a design exercise, and thus the exercise instructions had to be made up. Appendix F presents the group instructions.

Each group was equipped with pencils and colouring pens, an A3-sized notebook, transparencies and drawing ink, scissors, a ruler, and an eraser. For the presentation,

they were asked to prepare transparencies of the user profiles, use cases, user interface designs and main findings from the usability test; one slide of each.

The author did not participate in the groups' work. She followed each groups' working and spent a few minutes at the time in each room listening, observing, and documenting the work both in writing and with a digital still camera. The groups were encouraged to ask for help if they needed, and at few times the author gave the groups some practical hints. Especially in user testing preparations the author reminded the groups about necessary preparations before the test user arrives. Most of the time, however, the author tried to stay as invisible as possible and to let the groups work without interruptions. Below in Figure 16, two photographs illustrate the usability tests performed by the groups.

Figure 16. Usability testing.

In the final presentations the groups presented their designs and reported the main findings from the tests. Each group had found several problems in the interface, many of them concerning the interaction with the system: terminology, naming of buttons and the layout of components. They had also noted that the tests would have required a lot more planning and preparations. One of the groups admitted that they should have divided the work more to keep up with the schedule; instead, they had worked as a team and done most of the design together. The groups also noted that they had been forced to test the system with known defects because they did not have time to redraw the prototype. Nevertheless, the groups agreed that user testing proved relevant problems in the design and testing was collectively considered useful.

After the workshop the feedback questionnaires (see Appendix G) and design exercises (see Appendix H) were given. The discussion continued during lunch that was served after the workshop; the employees even used the terminology adopted from the heuristics in discussing certain features of user interface designs.

Comments were informally gathered right after the workshop from a few participants. They had liked the second workshop much and found it a totally different way to study than the first, theory-oriented workshop. In their opinion, time passed very quickly this time and they did not feel tired at any time. The design exercise was considered motivating, creative, and inspiring.

Observations and Discussion

Following the groups working was interesting and useful. The groups seemed to have a good time designing the system, and they got a good start with the instructions. User profiles and use cases were methods that had been taught earlier, and each group was able to form a set of profiles and use cases without any help from the author. It seemed that these methods had been adopted very well. One group had an interesting discussion about use cases and how they should be formulated; having this conversation the group members had taught each other and at the same time shared their opinions about the methods. Having these kinds of discussions can open one's eyes to different opinions and interpretations, and offer a wider perspective to the topic.

A problem common to all groups was that they aimed too high in the design project. Although they were reminded of the time constraints, they tried to design large systems with multiple features, instead of concentrating on only the most important functionalities as instructed. Probably this should have been even more emphasized in the written instructions, too.

The groups functioned together somewhat differently. Two of the groups were probably more initiative and active than the other two; yet all four groups managed to plan, design, and test the system on schedule. The social skills and personality differences affected the group dynamics, and the differences showed up most in the test situation.

Considering the time constraints and that none of the participants had previous experience in such a test situation, the tests went quite nicely. Naturally the inexperience of the experimenters made the tests rather unmanaged; an experienced experimenter could have been able to keep the test on track also when essential features were missing in the prototype. The role division was also somewhat unstable in the tests; the observers could not always keep quiet and let the experimenter lead the test. Instead, the functionalities were discussed together, and the test situation in one group resembled an informal walkthrough rather than a real usability test. The author did not consider this solution that bad after all, because the designs were very early and there were several clear defects in the prototypes. The test users were not much encouraged to think aloud by the experimenters either, and parts of the tests were conducted in quiet when no obvious problems were encountered. This would also have required more experience from the experimenter to know how to have the test user think aloud more.

Some of the problems can be explained by the limited schedule: the groups did not have enough time to draw all views of the interfaces, and the group members did not always have a common understanding of specific features in the system while testing it. All functionalities had not been even planned, and this caused problems when the test user wanted to know more about them. Also there was close to no time to plan the test in advance, and thus the instructions for the test users were rather ambiguous. Considering the time available for the groups, they did a fairly good job. The need for good test preparations and the strong recommendation for a pilot test were yet proved true in this experiment.

7.4.3 Feedback Session

A short feedback session was held after the workshops during a personnel meeting in June 2005. The session served as a close-up for the project and gave the participants a

chance to give some final comments and to present questions. The main results from the feedback questionnaire and design exercises were presented, and some questions gathered from the feedbacks were answered. The author used a few minutes for going through the suggestions how to consider usability in every-day work (see section 6.1). A short discussion was had about the feedback results, too. Finally, everyone was thanked for their participation in the training project. As a symbolic gesture everyone received a diploma for their participation in the training.

7.5 Training Results and Implications

To be able to assess the effectiveness of the training, its measurement was carefully planned ahead. The measurement was done based on the heuristic evaluation exercise from the first workshop, the feedback results gathered with a questionnaire, and the design exercise done again after the training. In this section the results from the effect measurement are presented.

7.5.1 Heuristic Evaluation Exercises

The evaluation exercise was done twice in the first workshop. First the participants individually evaluated the library catalogue service with their current understanding and the exercise was repeated after the training by using the heuristic evaluation method. The evaluation from each evaluation rounds were collected for later analysis.

The total amount of problems found did not remarkably differ between the two evaluation rounds. Altogether 69 different problems were identified in the first evaluation, whereas 79 problems were listed in the second round. In the first round, the number of problems found per evaluator ranged from one to 23 problems with the median number being 8 and the average number 9. In the second round the range of problems found varied from four to 18 problems and the median and average numbers were 10 and 11 respectively. The result indicates a moderate increase in the number of found problems, although the maximum number of problems found per evaluator actually decreased. This can be due to several reasons: either the evaluators did not bother to write down all the problems again, or they may have examined the interface from a different perspective now when they had the heuristics to compare the user interface against. Moreover, the author had to split some of the individual problem descriptions in more than one problem to be able to combine the results from several evaluators. At the first round the problem descriptions were very lengthy and ambiguous, and did not state the problem very clearly, which may have increased the number of problems found per evaluator.

The findings from the first and second evaluation rounds were somewhat different. Most problems from the first round concerned the information contents of the page and the presentation and order of the interface elements – namely the naturalness of the dialogue in the user interface. Also the terminology received several comments. A few problems in layout consistency were discovered and suggestions were made to provide more shortcuts. In the second round where the design heuristics were used, the findings showed some changes in the focus. Naturally the same findings from the previous round were reported, but additional remarks had been made about navigation and exits, providing help and prevention of errors. Consistency problems were also clearly more emphasized in the evaluations. This shows that with the help of heuristics, the

evaluators were able to identify more problems of different types. Naturally, there was more time for the latter evaluation which partly contributed to finding more problems.

What was also different between the two evaluations was the number of evaluators who found specific problems. In the first round the largest amount of evaluators for one problem was five, whereas in the second round the largest number was seven. Moreover, while in the first round there were two problems that were found by five evaluators, in the second round there were ten problems that were found by five or more evaluators. This result clearly shows that the training enhanced the evaluators' abilities to identify problems in the interface in general.

The participants indicated that the heuristics helped in finding the problems. However, it seems that the evaluators did not quite understand how to apply the heuristics. When the evaluators were asked to specify the violated heuristic(s) for each problem, clearly inapplicable heuristics were also proposed. The evaluators had typically listed many heuristics, both appropriate and unsuitable heuristics. Especially the heuristics 2 (Speak the user's language) and 3 (Minimize the user's memory load) were extensively offered in many cases. As an interesting example, an ambiguous colouring of the navigation buttons illustrated in Figure 17 received evaluations for all the heuristics 1, 2, 3, 4, 5, and 6 (see Table 6 on page 56 for definitions). In this example, the heuristics 2 (Speak the user's language) and 5 (Provide feedback) seem quite inapplicable.

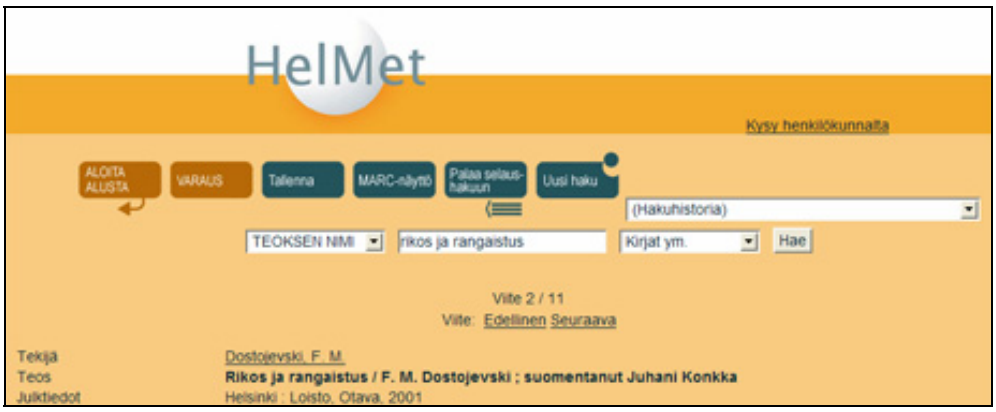

Figure 17. HelMet catalogue service (http://www.helmet.fi/).

The evaluators were also quite unsure of the severity of the problems. On the severity scale from zero to four, two (minor problem) and three (major problem) were the most given severity evaluations. As an indication of missing a common reference scale among the evaluators, the same problem would receive evaluations of all severities, i.e. one, two, three, and four. The inability to decide how big a problem actually is may affect the team's ability to prioritize between their fixing.

Another evaluation was performed as homework for a user interface design of an in-house project waiting for implementation. This evaluation yielded 40 usability problems, most of them cosmetic or minor problems. The number of problems found by a single evaluator ranged from one to twelve problems with a median of five problems. One major problem was pointed out by nine evaluators, showing that this problem should clearly be given a thought before implementing the design. All evaluations were

considered relevant for the further development of the design, and thus the homework exercise actually produced exploitable results for the benefit of the company.

In both heuristic evaluations, the heuristics for feedback, error messages, and error prevention received very few comments. This was clearly due to the fact that the evaluation was done with printed screenshots of the user interface. Lacking real use experience and contact with the look-and-feel, those types of usability problems are difficult to detect.

7.5.2 Feedback

Feedback from the training was gathered with a questionnaire (see Appendix G). All 13 participants returned their questionnaire anonymously. Generally, the feedback from the training was very positive, and the respondents had clearly taken time to answer the questions. Reflection and self-analysis showed in the responses. The participants also gave many constructive ideas for further training. Knowing the difficulties in interpreting numerical assessments in questionnaires, the written answers were given bigger weight than the exact numbers, even though average grades and other statistical information was gathered from the closed-ended questions, too.

The training met the participants' expectations well, and three out of four respondents suggested that it had even somewhat exceeded their expectations. Hands-on training was reviewed as very educational and rewarding, and the respondents told that they had had fun in the training. "Learning by doing" repeated in nine responses, and no one expressed hoping that more time had spent in lectures. The whole personnel working together created good feelings of togetherness, and the presentations and discussions had been an important supplement for many participants.

The training was reviewed as quite useful. Yet, in these comments the respondent's job description made a big difference: those who reported that they regularly participate in user interface design found the training more useful (on average 4.0 on scale of 1='useless' to 5='very useful') than those who did not see it as a big part of their work (on average 2.7). The division between these two groups did not quite follow the job titles – in addition to administrative personnel there were also software developers who did not see the training relating much in their current work.

When asked what the most important offering of the training was, many different answers were received. According to four respondents, the training offered a good understanding of usability and the design process as a whole. As single techniques, the ten design heuristics and other practical tips shared were regarded important and useful for work, and they were seen to provide a concrete way to evaluate the usability of an interface beyond common sense.

The training had refreshed the participants' memories of the teachings from the user interface lecture held earlier. Nine out of 11 respondents thought that this new training linked to and complemented the earlier training well or very well – two participants were unable to answer the question. None of the respondents had the impression that the training would have conflicted with earlier teachings, nor did it overlap too much with them. Usability testing, design heuristics, and some new details and tips were considered as new topics compared to the earlier training, and generally the respondents saw that they supported it well. Also the old methods of user profiling and use cases

were well rehearsed in the second workshop through hands-on training. According to two respondents, the training also provided everybody with a common terminology and made it easier to talk about usability with others.

Many respondents, nine in total, had nothing to complain about the training and they were not able to name any specific topic that would not have been useful. One respondent stated that some of the methods presented, specifically usability testing, seemed unrealistic in current projects where there is little time for planning. Another one questioned the usefulness of paper prototyping, as mock-up software prototypes can be produced with current tools without functioning code and they are easy and fast to change when needed. One respondent thought that the usability engineering process chart was ambiguous and hoped that more time had been used to explain the differences between the phases. One felt being already familiar with usability engineering basics but admitted yet that repetition is a key to learning.

Usability Methods

The respondents were asked to think which usability methods discussed in the training they think they will use in the future. The responses of those who did not see themselves as participating in the design process were given less weight in the analysis here. User profiling as a tool to support system design divided opinions. Nine respondents believed they will use it in their work some way, but many of them estimated that they won't have time to put much effort in it. The basic idea of thinking about who will use the product was considered important, though. In continuous collaboration with the same customer, profiling was not seen that important because the users are already known by default.

Use cases of all methods were generally most appreciated by the respondents – ten supporting responses together – and three of the respondents reported already using them in their work. A bit disappointingly, user environment analysis was not ranked that high: eight respondents thought they might use it, yet three of them added the disclaimer "when needed"; the need was not seen to come up very often. Two respondents thought that this information is usually included in the initial specifications. Probably the fact that much work is done to the same customers reflects in these answers, too – when the environment is already known, there is no clear need to perform a thorough study of the environment. Nevertheless, as one respondent wrote, this is something that is often unconsciously used.

Paper prototyping received both praises and resistance. It was considered useful for sketching the first designs in design meetings, and three respondents said doing so even now. Yet, in three other responses semi-functional software prototypes were seen as potentially more useful tools for presenting designs for customers.

The two evaluation methods, heuristic evaluation and usability testing, were seen as rather formal and heavy-weight, but their value was clearly appreciated. Eight respondents saw the design heuristics as a useful tool for further work and estimated that they will use them when designing user interfaces. Formal heuristic evaluation received a little less support – the guidelines were better seen as a suitable checklist. Three respondents hoped that usability testing could be used more; altogether five believed they could use it somewhere. Four respondents doubted if usability testing is a

realistic method to be used in the company, and one respondent stated that like software testing, also usability testing should be done by separate people specialized in it.

Total Success and Development Ideas

The effect of the training as a whole was considered important. The training brought the topic back in discussion and reminded of things that had gone forgotten. Six respondents estimated that they will consider usability more in the future and put effort in planning in the beginning of the project if time allows. Two respondents said they will do more planning on paper before moving on to writing code. One respondent wrote that even if no extra time is reserved for planning, the guidelines learned in the training will stay in mind and influence the designs – the same message could be read between the lines in many other responses, too. Even those who did not see themselves as being part of every-day software projects had found useful information in the training, and reported being more aware of the importance of users and usability in all work. Only two respondents estimated that the training did not have much influence in their working habits; they were both already familiar with most topics discussed and they thought that their attitude towards usability had already changed towards respecting of end users. One respondent also noted that in order to put more effort in user interface design, the current work processes should be changed respectively.

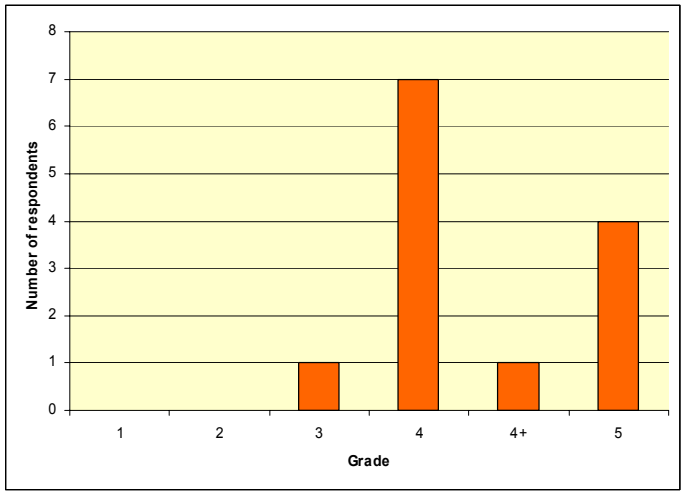

Figure 18. Overall grades given for the training on one-to-five point scale.

An overall grade for the training received seven reviews of 4 (good) and four grades of 5 (very good). One respondent gave the training a grade of 4+, and another gave the grade 3 because the contents were quite familiar to him. The average grade was thus 4.25. Four respondents found the lack of time the biggest issue in the training, and one of them said that with a little more time used he would have given the grade 5. Figure 18 illustrates the grade distribution.

A couple of respondents noted that big changes don't happen in one day. In order to take fully advantage of the things learned here, more practice is still needed. Nine respondents altogether reported being interested or very interested (four or five on one-to-five-point scale) in further training. In the opinion of the respondents, topics that still need more discussion were the integration of usability techniques into the software development process and the overall process of UCD (four respondents together) among

others. Two respondents hoped that the training would have dealt also with technical implementation of user interfaces, and four hoped for more practical examples of good and bad interface designs.

7.5.3 Design Exercise

The design exercise was received from all 13 participants. They were compared with the designs from the current state analysis phase. Two persons who participated in the training now had not done the exercise earlier and thus their designs could not be compared to earlier designs. Moreover, two persons who did the exercise in the current state analysis phase did not participate in the training and did not return the latter exercise either. One respondent did not return a new design drawing at all, but stated instead that he wouldn't change anything in the previous design.

Altogether five respondents wrote user profiles and use cases in the exercise, one of which had not done the exercise before. As a completely new topic, the user groups were listed in two assignments before the user profiles, and the profiles were written based on the identified user groups. Those who wrote the profiles on the first time did so this time, too, and one person who had not considered users at all in the first round did so now. The number of identified user profiles ranged from two to five; in the first round the range was from one to four. Common to all profiles was that they were more detailed and somewhat longer. Whereas the profiles written before the training mostly consisted of the user name, age, and profession, now the users had been given also personal characteristics such as hobbies and family members, and also their experience in using computers was stated in two profiles. One respondent had written the use cases using Unified Modeling Language (UML) notation, and the rest had listed them separately for each profile. One respondent mentioned a target user group by name ("travellers") but without characterizing it more. Seven respondents did not write nor report thinking of user profiles or use cases.

The respondents were asked how they approached the exercise. Almost everyone clearly reported that they aimed at an easy and simple solution. Versatility and special features had also been a goal in three designs. Two respondents clearly said that they had kept the heuristics in mind while designing the user interface. One of them listed several benefits for his design solution:

> *R4: "Ruudulla vain olennainen tieto, etenee luontevasti. Sama tieto aina samassa paikassa. Nopea käyttää –> yleisimpien käytettyjen valuuttojen pikapainikkeet (käyttäjäkohtaiset)! Tarjoaa oikoteitä, tarjoaa apua ja opasteita."* [20]

Judging by the terminology and the way the respondents described their designs, the training had had an effect in other designs, too:

> *R3: "Pyrin tekemään nopean ja yksinkertaisen dialogin omaavan työkalun."* [21]

---

[20] "Only relevant information on the screen, progresses naturally. Same information always in the same place. Fast to use -> hot keys to most commonly used currencies (user-specific)! Offers shortcuts, provides help."

[21] "I tried to make a tool with a fast and simple dialogue."

R1: "Näkymiä on vähän ja käyttäjän pitäisi pystyä tekemään helposti muutoksia ilman että palaa aina alkuun." [22]

Four respondents told that they had designed a user interface that they thought they would like to use themselves.

Generally the user interface designs resembled those done before the training. Only one respondent had fully changed the design; others had only added or removed some minor functions or features. The respondents reported that it was rather difficult to break out from the initial idea and that is why they did not alter their design very much. Another reason presented was that the initial design was still good and did not need any specific changes.

Some common improvements in usability could be identified in the designs. A descriptive instruction text or Help link had been added to four designs. Also, in one design an abstract note "ohje" ("instructions") had been replaced with a concrete text example. Selections for user language had been added to two designs but at the same time removed from one. A possibility to select the country from a map had been added in three designs, and search by country name in two designs. (In the first round a map was used in one design and country name in six designs.) Two designers had added an automatic update for the fields as the user types in numbers or text. Two designs showed the navigation path at the upper part of the page when proceeding on from the main view. Four designs now included a logout or exit function for leaving the service. Additional features included adding a chart for the exchange rate history (in two designs) and possibility to order and pay foreign currencies in the same service (one design). Appendix I shows a numerical comparison by certain features between the two design rounds.

The respondents were asked if they could identify potential problems in their design. One respondent had done a heuristic evaluation for his design and found four problems of severity 1 or 2. One stated that he cannot tell personally what the problems are, but that usability testing would probably reveal some. Two respondents remarked that the user interface might be a bit complicated for a novice user even though they had tried to keep it simple. One respondent said that he found it difficult to provide a short but informative instruction text for the user. Two respondents pointed out that their design did not offer a possibility to save earlier conversions. Many respondents also pointed out possible challenges in technical implementation, such as automatic updating of text fields or database maintenance.

The time used to the design exercise varied from 10 to 84 minutes with a median in 30 minutes and the average being 38 minutes. The instructions had suggested 30 to 60 minutes to be used for the exercise. Notably, writing user profiles and use cases increased the total time used for the exercise.

One month earlier, during a monthly meeting where the current state analysis results were discussed, the author had referred to the design exercise and stated that she had

[22] "There are only few views and the user should be able to make changes easily without having to go back to the start."

hoped everyone would think about the users and what the typical uses for the tool are. It seems that this message had not been enough emphasized, or the participants had forgotten it. On the other hand, user profiles and use cases were not specifically asked for in the exercise instructions; it was intentionally left for the designers to decide how to approach the assignment. Probably the system was so simple and the main use cases so evident that everyone had not seen a specific need to stop to think about the context of use. Still, this result implies that unless explicitly asked for, usability engineering methods are not necessarily formally used, even though they are taught in lessons.

Quite interestingly, no one reported having discussed the exercise with work colleagues between the two design rounds. Initially it was expected that possible discussions might affect the design solutions on the next round. The feedback session at the company meeting after the current state analysis was mentioned by one participant only, although everyone was present there – this indicates that the feedback received there might not have been consciously thought about when making the second design exercise.

7.6 Discussion of the Arrangements

Judging by the feedback, the training succeeded well in bringing the topic back in the minds of the participants and raised positive interest in further discussions. The participants found the training very useful and were able to identify and name several topics that would help their work in current jobs. Especially the examples of good design solutions included in the presentation of the design heuristics were considered practical and easy to use as a reference. The training also managed to strengthen the understanding among the participants that usability engineering is a process that influences all phases of software development. Motivation to include usability engineering in work was generally very strong, and the feedback indicates that with adequate resources the techniques could be used also in real work.

During this experiment there was little time for the participants to rehearse the new skills in practice. The effect that the training has on real work will show up later on and the final effect can be analyzed only after some time. Learning takes time and immediate improvements are rare, because learning requires repetition and rehearse. Nevertheless, small improvements were identified in the design exercise: providing more instructions and offering search tools show that the designers tried to support and facilitate the users in their tasks. Also, the drawings were more detailed and somewhat less ambiguous than in the first designs: this proves that the skills to sketch designs on paper may also have improved. More important than the actual drawings were however the reasons for selecting certain features, and the descriptions of the designers showed that the design heuristics had been considered when making the designs. Written user profiles and use cases – when done – showed that the context of use was thought about, and that the users' needs were really thought over. Still, many designers did not report paying attention to the actual context of use or at least their doing so could not be observed.

Considering the short time span there was available for the training project and given the feedback and results from the design exercise, the training was successful and fulfilled the personnel's expectations. Motivationally the training had a significant effect on the participants, and the practical skills of the employees were enhanced by letting them try the techniques in practice – some of them for the first time.

Yet, one drawback must be mentioned. Probably the biggest deficiency in the training was that no real users were involved in the workshop where the design process was simulated. User profiles and use cases were the result of the developers' imagination (although formed based on the written instruction that somewhat reminded a lightweight user and task analysis), and the test users were selected from among work colleagues. One group even reported that during the usability test, their test user tried to act more stupid than he really was which certainly affected the test results negatively. In practice, it would have been virtually impossible to include real users in this schedule: testing the designs with real users at that early stage would have required more time to plan and to prepare the test, which was not possible under the available time frame. As a consequence, however, real user-centeredness did not make its way to the workshop, because real-life users were not available to comment on the designs and contrast them with their own work habits. In the opinion of the author, the most eye-opening experience in user testing is to see how the user and the system really interact, and to become part of the user's work by observing, listening, and learning. The appreciation of the user as a domain expert was missing in this experiment, because the system and the domain were as new to the test users as they were to the designers. Also, the test user being a colleague may have removed some of the tension that is always present in test situations – whether positive or negative. Hence, this two-day training may not have fully convinced the software engineers of the value gained from usability engineering; however, the practical skills, common knowledge, and understanding of the process were certainly enhanced. Yet, none of the participants really complained about the test arrangements per se in the feedback, except for the tight schedule in general.

The training was offered for the whole personnel of the company. All participants were present in both workshops which ensured that everyone had the same experience of the training as a whole. The administrative personnel being present did not remarkably reduce group efficiency in the workshops; instead, it can be argued that including them helped in establishing a common terminology and a conceptual understanding among the whole personnel, which is important in this early phase of UCD integration. The feedback questionnaire results showed that the training did indeed contain a few topics that those people did not find quite useful for their own work; yet, the feedback in general was positive and also the people who did not see user interface design being a big part of their jobs had found the training interesting and felt it is good to know about these things.

Unfortunately, three employees were not able to participate in the training, including both web graphics designers of the company. This may have some effect on how usability is taken into account in graphical designs of the interfaces. There is potentially a need to spread the teachings to these employees, too.

8 CONCLUSIONS AND DISCUSSION

This thesis has reported a case study of introducing user-centred design in a small-size Finnish software development company, Spellpoint Group. During the study, a two-day workshop of user-centred design was organized for the company's personnel. The training contents were selected based on a current state analysis conducted in the company prior to the training. A review of related work discussed user-centred design, usability engineering techniques, and challenges in UCD training and integration.

The main objective of this work was to establish a common user-centred method infrastructure at Spellpoint Group and to enhance software developers' UCD abilities in various phases of software development. In the beginning of this study (see section 1.3), the practical objectives were to find out what kind of training the company needs, and what the contents of the training should be. The answer to these questions was searched by conducting a current state analysis of the company's working methods and current knowledge in user-centred design. The current state analysis showed a strong interest in usability within the company. The personnel saw usability as an important quality attribute of software and thought it is necessary to consider end users' goals in all software development. However, the employees felt they did not have enough knowledge to build usability in a product, and their conceptual understanding of usability was quite weak. Although there are some developers with prior knowledge in usability, there aren't any designated usability engineers in the house and usability engineering is not generally practised during software development projects.

After the current state analysis, the study objectives were refined (see section 3.8). The new study objectives aimed at resolving how usability engineering methods could be integrated into current working practices. Again, the question of an appropriate usability training programme was posed. Furthermore, it was asked whether user-centred design can be taught by organizing a short intensive training.

Based on the review of other related work, an approach to usability engineering was suggested for the company in Chapter 6, focusing on lightweight techniques and easy employability. Five rules of thumb were introduced to facilitate usability in software development: early customer focus, feature prioritization, and low-fidelity prototyping were emphasized. Also, a more systematic process was suggested to be used in larger projects, should the resources allow it. However, at this early stage, formal methodology and processes were not overly emphasized. The suggestions and findings from the literature were used as a basis for a training programme that was organized to target the most critical aspects of user-centred design and to give the personnel a chance to try UE techniques in practice.

The training participants' learning was measured with a design exercise, heuristic evaluation exercise, and their own feedback. The results showed that the training had enforced a positive attitude towards usability engineering, provided the employees with new tips for user interface design, and reminded of the importance of user recognition. Especially the ten design heuristics (Nielsen, 1993) presented in the training were well learned and generally considered useful. User interface designs showed improvements especially in the aspects of user help, navigation, and shortcuts. Heuristics helped the participants to identify more problems in a user interface and provided them with knowledge in doing informed choices between user interface solutions. The training

was appreciated by the participants and it received many praising comments. It also achieved in creating a strong motivation in further training in UCD, especially in UCD integration in development process and technical implementation of user interfaces. Judging by the results from learning measurements, it can be stated that the training did a good job in introducing usability engineering to the company.

8.1 Reflections

Focus

Initially, the focus of the training was in every-day tools and tips that could be used to facilitate software developers' work. After the current state analysis the focus slightly shifted towards more process-oriented training where a lightweight UCD process was introduced and some essential UCD techniques were tried in practice. Practical tips were included mainly in the presentation of the ten heuristics; here some examples of good UI design were given and poor ones shown as warnings. Also the heuristic evaluation done in the workshop seemingly proved the problems that lead to poor interaction. Letting the developers learn from bad examples is one way to nurture better solutions.

This shift from a purely practical "hints and tips" workshop to a somewhat more process-oriented training was due to the observations and comments gathered during the current state analysis. The personnel clearly indicated that there was no established software process model in the company, and that the lack of control of the development process leads to problems of many kinds. Also many interviewees wished to have a method or tool to help the design and to identify possible problems in the interface; having used to certain solutions can cause a developer repeating them project after project even if the solutions are poor. Moreover, the earlier usability training held in the company had already successfully addressed practical tips and design patterns for user interface design.

Thinking back the training sessions, the time constraints did affect their total success a bit. Especially usability testing as a complex topic suffered from the lack of time as it forced the presentation to stay at a general level. In the workshop the groups tried usability testing on their own, which may not have given them a fully correct image of the method as when done by experts. Allowing the software developers to observe a real usability test would possibly have offered a deeper understanding of the activity. Unfortunately, that was not feasible because of the lacking infrastructure and missing a system suitable for testing. Yet, the choice to employ hands-on training was definitely the right approach for making a lasting impact on the employees.

Although initially considered, no written self-study material was produced for the software developers. At the beginning of the project there was an idea of writing a style guide or design instructions with practical tips and tools to support software developers' work. After the training this was not done, however, as the scope of the project was considered to be quite large already. The training material and presentation slides were handed out to all employees, and this material could potentially be used also later for training new employees.

Study Methods and Implementation

The author was able to interview everybody in the company for the current state analysis, which let all the employees participate in the project. It also served as a motivation for the upcoming workshops. Naturally, the project was carried out with limited resources, which affected the methods that were available for both data collection and training workshops.

The selected study methods for the current state analysis – interview, questionnaire, design exercise, work list inspection, and observation – produced a large amount of data, which was more than enough to form an understanding of the company's work structures and the personnel's knowledge in UCD. Consequently, the extensive amount of data required a fairly large effort in selecting, sorting, and analyzing the pieces of information.

The training's effect on the participants' learning was measured with a feedback questionnaire and two different practical exercises. The empirical data from the exercises was valuable and gave wider perspective to the analysis. Although quite challenging and not fully unambiguous, the design exercise and heuristic evaluation results can be seen to reflect the effect on the participants' learning well.

The company's customers were not contacted during the study. Measuring the training's effect on customer work and software designs would have provided material from real work, but the time was considered too short for real improvements to show up. Had there been a suitable project on-going by the time of this project, the measurement could have included the customer effect as well. In the conditions of this project, however, this kind of empirical data was not available.

Before this study, the author had worked for the company as a software developer for six months. Within that time, she participated in designing user interfaces in several projects and shared her knowledge to other employees through informal discussions, too. This has definitely affected the other employees' perceptions about usability recognition in the company, and was also brought up in the interview during the current state analysis. What is difficult to estimate is how the results were affected by the fact that the author was an in-house, not an outside observer. The interviews and training workshops were conducted with good team spirit. It is likely that an outside researcher would have needed more time to get to know people and for establishing trustful relationships with other employees. The way the study was conducted now, there was already mutual trust and existing communication channels with the other employees, which not only made the practical arrangements easier, but also may have helped the author to dig deeper into the organization.

The author did, however, acknowledge the risk of data distortion because of her being an insider and having personal experiences and perceptions of the work practices in the company as well. At all phases, the results were carefully examined without any assumptions. Where the author's own observations or feelings were used as a source, it was also clearly stated in order to avoid misunderstandings and data distortion.

Was It Worth It?

This project took half a year and included fourteen individual interview sessions with the company's personnel, each taking from one and a half hours to three hours, two training workshops of four hours for the whole company personnel, and two presentations in monthly meetings also for the whole personnel, 80 minutes together. In addition, each employee spent work time for the "home assignment" from the workshops, from half an hour to two hours altogether. For each employee that participated in the training project from the beginning to the end – eleven together – the total time used was approximately 12 to 14 hours. Additionally, the information gathering and training involved the author's work from January to May with an average contribution of 20 hours a week.

One of the exercises done in the training, heuristic evaluation to the user interface for an in-house software product, served as productive work for the company, and the results from the evaluations provided the development team with concrete improvement ideas for the product. The rest of the training material was imaginary or concentrated on evaluating other products than the company's own software. The experiences from those exercises were good: they served well the purposes of the training that were to demonstrate usability problems with existing products and to rehearse usability engineering techniques in a software development project. Yet, the outcome of those exercises is not further utilizable for the company.

Based on the total evaluation of the project, it did a good job and fulfilled more than well the expectations that were set up to it before the training. To take full advantage of the investment of time and the contribution of this training, the skills learned and rehearsed should be further maintained and put to use in future software projects. Unless the training contents are integrated into working practices, the effect of the training will fade and the practices learned will be forgotten.

8.2 Company Future

Spellpoint Group being a young company with only five years of history, it is natural that formal working procedures are only starting to establish. Willingness for process improvement and agreeing on common working methods is a sign of such interest. The company has only recently grown in size and is still growing – during this training project alone three new employees were hired. The company still has little in-house software development, which makes the lack of structures understandable – customers' work procedures and schedules have dominated most projects in the company's history. Moreover, the various cultures inside the company, caused by the physical separation and different backgrounds of the employees, is one factor that will raise future challenges.

Spreading a right attitude towards usability and user recognition in the whole organization is a large step towards user-centred design, especially for a small company. As long as it is the software developers who are mainly responsible for the software development outcomes and as such the user interfaces produced, teaching the basics of user-centred design is a leap in the right direction to ensure that users will be taken into account during the design process. In the case of Spellpoint Group, this is how the work is currently done: in small teams or even more often – alone. Offering support,

information, and a selection of suitable user-centred design principles to be used where applicable will most likely have an effect on the quality of the software.

Establishing a steady user-centred design process in a company is a complicated and a long-term project. It will affect everybody in the organization, starting from the top management's commitment to the topic. If user-centred design is a goal, each project manager at an individual project level is responsible for ensuring that sufficient resources are given to UCD, and that user-centred design techniques are applied in all phases of software development. At the first steps, this might be possible only in selected projects where more resources can be used, and later when experiences have been gained, the process could be extended to minor projects as well.

When the organization grows, however, it will become practically unfeasible to train every software developer to a novice UCD practitioner. If the company wants to establish UCD as a common work practice and philosophy, the software development teams need to be complemented with a dedicated UCD specialist. This team member would bring along the essential knowledge and practical skills in usability engineering techniques needed in the development process. Also, regarding the status that usability testing alone has gained in the field, it is difficult to imagine a company saying that they do usability without employing any formal evaluation methods.

In the section 2.4 three dimensions of UCD were presented: 1) the UCD infrastructure, 2) the performance of UCD, and 3) usability in business strategy (Jokela, 2001). This project has concentrated on enhancing the practical skills of the personnel, and thus focused on establishing an adequate UCD infrastructure in the company. The other two dimensions were deliberately left outside the scope of this project. In the author's opinion there is enough knowledge now at Spellpoint to start establishing a UCD process and a common culture towards usability. Next steps for the company will be to concentrate on the two other dimensions – that is, the UCD performance and the business strategy aspects – to be fully able to exploit UCD and guarantee a steady and effective implementation of usability in software products. As Jokela (2001) points out, it is not enough to talk about usability in a positive tone; instead, in order to become an organization-wide practice, usability needs to become an objective for each manager, and effort must be made to integrate UE outputs into the development process.

It remains to see how seriously usability engineering will be taken in future projects at Spellpoint Group. Will UCD find its way to every-day work, or stay as a nice theory rather than a practice? This will heavily depend on the management and resources of upcoming projects. Formal usability engineering could be taken as a principle in in-house software development projects, but in customer projects the suggestions for a more informal process (as presented in section 6.1) will probably be more suitable. If usability engineering is successfully built into the development practices, the costs of the development will most likely pay off in reduced corrections to ready software, better match with customer expectations, and quality in user interface design. Yet, to achieve any major improvements in usability of the software produced, some changes in current working methods are essential. If the usability aspect remains to be ignored at the early phases of a project, an individual software developer has few means for increasing usability late in the implementation phase. As discussed earlier in this thesis, actions at the end of a project can mainly concentrate in improving the look-and-feel of a user

interface, but the most fundamental problems in user–computer interaction and the structure of the system as a whole can rarely be changed.

The company also needs to address the question of training employees. To keep the skill level up in the company, new employees need to be trained to UCD as well. Old employees' skills must be regularly utilized to prevent them from forgetting what they already know. Further training for software developers and web designers is likely to be needed. In the opinion of the author, however, the training organized during this study is likely to be enough for those employees not directly involved in daily software development practices.

8.3 Generalizing the Results

This study has highlighted the characteristics of software business from the perspective of a small software company. The challenges of time, money, and customer requirements are common to all companies in the tailored-software business. Small companies employing mostly software developers are likely to struggle with the same questions identified here, namely the lack of knowledge about UCD practices, project management resources, and time available for a full-size usability engineering process.

The suggestions for appropriate usability engineering techniques and the training programme organized were both built on the existing knowledge in the case organization of this study. Positive attitude towards usability and understanding of the need to consider users in the development process were already established, and the training did not need to focus that much on the very basics of human-centred approach and HCI. Had there been no introduction to usability in the company before this project, the emphasis of the training would have been somewhat different. Now, the contents were selected to continue from the base set by the earlier training, and to deepen the understanding of the process of usability engineering as a whole. In the author's opinion, the contribution and results of this study can nevertheless be utilized in other companies, too, if enough care is taken to ensure sufficient preliminary knowledge in the company in question.

8.4 Further Research

During this study a few questions were left unanswered and new interesting fields of study were revealed. This section presents some topics for further research.

Management Role in UCD

This work did not really address the role of management in the implementation of a UCD process. It is clear, however, that actions taken towards improvements must always be supported and led by managers. An interesting question is exactly how much responsibility an individual software developer can be expected to carry in implementing usability engineering practices? Should all usability engineering be led and supervised by managers and experienced usability specialists, or can individual teams successfully initiate in UCD?

Usability Engineers vs. Software Engineers

In this case study software developers were trained to take users more into account in a software development process. What was not very much discussed was whether software developers alone can proficiently use UCD methodology, or if usability specialists should always oversee the usability development process. This has already been addressed in discussions about software engineering curricula of universities and academies (e.g. Seffah & Adreevskaia, 2003; Rozanski & Haake, 2003; Faulkner & Culwin, 2000), and the debate will probably continue. Can a software engineer become a usability engineer or will these two always be disconnected?

Marketing Usability

There is a need to further discuss how usability should be marketed to customers. Building user-centred design into a company's business strategy requires convincing the paying customers of the benefits of usability. The reasons for usability were discussed in this work, too, but empirical research would be needed about how to explain the need for usability to customers as well.

REFERENCES

Alanne, M. (2002). *Käytettävyyden kehittämisprosessin uudistaminen isolle ohjelmistotalolle. Pro gradu -tutkielma, Helsingin kauppakorkeakoulu.*

Ames, A. L. (2001). Users First! An Introduction to Usability and User-Centered Design and Development for Technical Information and Products. In *Proceedings of IEEE International Professional Communication Conference* (pp. 135–140). Institute of Electrical and Electronics Engineering, Inc.

Bevan, N. (1999). Quality in Use: Meeting User Needs for Quality. *Journal of Systems and Software, 49,* 89–96. ISSN: 0164-1212.

Beyer, H., & Holtzblatt, K. (1998). *Contextual Design: Defining Customer-Centered Systems.* San Francisco, CA: Morgan Kaufmann Publishers. ISBN: 1-55860-411-1.

Bias, R. (2005). Cost-Justifying Usability: The View from the Other Side of the Table. In R. G. Bias, & D. J. Mayhew (Eds.), *Cost-Justifying Usability. An Update for the Internet Age* (2nd ed.) (pp. 613–621). San Diego, CA: Morgan Kaufmann Publishers. ISBN: 0-12-095811-2.

Bias, R. G., & Mayhew, D. J. (Eds.) (2005). *Cost-Justifying Usability. An Update for the Internet Age* (2nd ed.). San Diego, CA: Morgan Kaufmann Publishers. ISBN: 0-12-095811-2.

Bloomer, S., & Croft, S. (1997). Pitching Usability to Your Organization. *Interactions, 4,* 18–26. ISSN: 1072-5520.

Brooks, P. (1994). Adding Value to Usability Testing. In J. Nielsen, & R. L. Mack (Eds.), *Usability Inspection Methods* (pp. 255–272). U.S.A: John Wiley & Sons, Inc. ISBN: 0-471-01877-5.

Butler, K. A. (1996). Usability Engineering Turns 10. *Interactions, 3,* 58–75. ISSN: 1072-5520.

Cooper, A. (1999). *Nörttien valtakunta. Miksi korkeateknologiatuotteet saavat meidät sekaisin ja kuinka palauttaa järki* (R. Parkkonen, Trans.). Jyväskylä: Suomen ATK-kustannus Oy. ISBN: 951-762-989-3.

Cooper, A. (2003). *About Face 2.0.: The Essentials of Interaction Design*. Indianapolis, IN: Wiley Publishing Inc. ISBN: 0-7645-26413.

Dayton, T., Barr, B., Burke, P. A., Cohill, A. M., Day, M. C., Dray, S., et al. (1993). Skills Needed by User-Centered Design Practitioners in Real Software Development Environments: Report on the CHI '92 Workshop. In *ACM SIGCHI Bulletin, 25,* 16–31. ISSN: 0736-6906.

Desurvire, H. W. (1994). Faster, Cheaper!! Are Usability Inspection Methods As Effective as Empirical Testing? In J. Nielsen, & R. L. Mack (Eds.), *Usability Inspection Methods* (pp. 173–202). U.S.A: John Wiley & Sons, Inc. ISBN: 0-471-01877-5.

Dumas, J. S., & Redish, J. C. (1994). *A Practical Guide to Usability Testing* (2nd ed.). Norwood, NJ: Ablex Publishing Corporation. ISBN: 0-89391-991-8.

Earthy, J. (1998). Usability Maturity Model: Human-Centredness Scale. INUSE Deliverable D5.1.4(s). Retrieved March 1, 2005 from http://www.lboro.ac.uk/eusc/guides/d514S_1c.doc.

European Usability Support Centres (EUSC) (2000). *Usability Assurance Guides.* Retrieved March 1, 2005 from http://www.lboro.ac.uk/eusc/r_usability_assurance.html.

Faulkner, X. (2000). *Usability Engineering.* Chippenham, Wiltshire: Palgrave. ISBN: 0-333-77321-7.

Faulkner, X., & Culwin, F. (2000). Enter the Usability Engineer: Integrating HCI and Software Engineering. In *Proceedings of the 5th annual SIGCSE/SIGCUE ITiCSE conference on Innovation and technology in computer science education* (pp. 61–64). New York, NY: ACM Press. ISBN: 1-58113-207-7.

Gould, J. D., Boies, S. J., & Lewis, C. (1991). Making Usable, Useful, Productivity-Enhancing Computer Applications. *Communications of the ACM, 34,* 74–85. ISSN: 0001-0782.

Gould, J. D., & Lewis, C. (1983). Designing for Usability – Key Principles and What Designers Think. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 50–53). New York, NY: ACM Press. ISBN: 0-89791-121-0.

Gould, J. D., & Lewis, C. (1985). Designing for Usability – Key Principles and What Designers Think. *Communications of the ACM, 28,* 300–311. ISSN: 0001-0782.

Hackos, J. T, & Redish, J. C. (1998). *User and Task Analysis for Interface Design.* U.S.A: John Wiley & Sons, Inc. ISBN: 0-471-17831-4.

Hakiel, S. (1999). Sufficient and Necessary Conditions for Routine Deployment of Used-Centred Design. In *IEE colloquium on Making User-Centred Design Work in Software Development (Ref. No. 1999/010) (*pp. 1/1–1/4*).* Institute of Electrical and Electronics Engineering, Inc.

Hansen, M. (1991). Ten Steps to Usability Testing. In *Proceedings of the 9th Annual Conference on Systems Documentation* (pp. 135–139). New York, NY: ACM Press. ISBN: 0-89791-452-X.

Iivari, N., & Abrahamsson, P. (2002). The Interaction Between Organizational Subcultures and User-Centered Design – A Case Study of an Implementation Effort. In *Proceedings of the 35th Annual Hawaii International conference on System Sciences, 2002* (pp. 3260–3268). Institute of Electrical and Electronics Engineering, Inc.

Jacobson, I., Booch, G., & Rumbaugh, J. (1999). The Unified Software Development Process. Addison-Wesley Longman. ISBN: 0-201-57169-2.

Jokela, T. (2001). *Assessment of User-Centred Design Processes as a Basis for Improvement Action. An Experimental Study in Industrial Settings. Department of Information Processing Science, University of Oulu.* Oulu: Oulu University Press. ISBN: 951-42-6551-3.

Jokela, T. (2002). Making User-Centred Design Common Sense: Striving for an Unambiguous and Communicative UCD Process Model. In *Proceedings of the second Nordic conference on Human-computer interaction* (pp. 19–26). New York, NY: ACM Press. ISBN: 1-58113-616-1.

Karat, J., & Dayton, T. (1995). Practical Education for Improving Software Usability. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 162–169). New York, NY: ACM Press/Addison-Wesley Publishing Co. ISBN: 0-201-84705-1.

Kuutti, K., Jokela, T., Nieminen, M., & Jokela, P. (1998). Assessing Human-Centred Design Processes in Product Development by Using the INUSE Maturity Model. In *Proceedings of the 7th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design and Evaluation of Man-Machine Systems*. Kyoto: IFAC.

Latzina, M., & Rummel, B. (2003). Soft(ware) Skills in Context: Corporate Usability Training Aiming at Cross-Disciplinary Collaboration. In *Proceedings of the 16th CSEE&T 2003 Conference on Software Engineering Education and Training* (pp. 52–57). Institute of Electrical and Electronics Engineering, Inc. ISSN: 1093-0175.

Maguire, M. (2000). Increasing the Influence of Usability Practices Within the Design Process. In *Extended Abstracts of the CHI '00 Conference on Human Factors in Computing Systems* (p. 305). New York, NY: ACM Press. ISBN: 1-58113-248-4.

Maguire, M. (2001). Context of Use Within Usability Activities. *International Journal of Human-Computer Studies, 55,* 453–483.

Mao, J-Y., Vredenburg, K., Smith, P., & Carey, T. (2001). User-Centered Design Methods in Practice: A Survey of the State of the Art. In *Proceedings of the 2001 conference of the Centre for Advanced Studies on Collaborative research* (pp. 12–24). IBM Press.

Marcus, A. (2005). User Interface Return on Investment: Examples and Statistics. In R. G. Bias, & D. J. Mayhew (Eds.), *Cost-Justifying Usability. An Update for the Internet Age.* (2nd ed.) (17–40). San Diego, CA: Morgan Kaufmann Publishers. ISBN: 0-12-095811-2.

Mayhew, D. J. (1999a). Strategic Development of the Usability Engineering Function. *Interactions, 6,* 27–34. ISSN: 1072-5520.

Mayhew, D. J. (1999b). *The Usability Engineering Lifecycle. A Practitioner's Handbook for User Interface Design.* San Francisco, CA: Morgan Kaufmann Publishers, Inc. ISBN: 1-55860-561-4.

Molich, R., & Nielsen, J. (1990). Improving a Human-Computer Dialogue. C*ommunications of the ACM*, *33,* 338–348. ISSN: 0001-0782.

Nielsen, J. (1993). *Usability Engineering*. London: Academic Press. ISBN: 0-12-518405-0.

Nielsen, J. (1994). *Heuristic Evaluation.* In J. Nielsen, & R. L. Mack. (Eds.), *Usability Inspection Methods* (pp. 25–62). U.S.A.: John Wiley & Sons, Inc. ISBN: 0-471-01877-5.

Nielsen, J. (2001) *First Rule of Usability? Don't Listen to Users. Alertbox, August 5, 2001*. Retrieved February 28, 2005 from http://www.useit.com/alertbox/20010805.html.

Nielsen, J. (2003). *Convincing Clients to Pay for Usability. Alertbox, May 19, 2003.* Retrieved April 10, 2005 from http://www.useit.com/alertbox/20030519.html.

Nielsen, J. (n.d.). *Ten Usability Heuristics.* Retrieved May 2, 2005 from http://www.useit.com/papers/heuristic/heuristic_list.html.

Nielsen, J., & Molich, R. (1990). Heuristic evaluation of user interfaces. In *Proceedings of the ACM CHI'90 conference on Human factors in computing system*s (pp. 249–256). New York, NY: ACM Press. ISBN: 0-201-50932-6.

Nonaka, I. (1994). A Dynamic Theory of Organizational Knowledge Creation. *Organizational Science, 5,* 14–37. ISSN: 1047-7039.

Nonaka, I., & Konno, N. (1998). The Concept of "Ba": Building a Foundation for Knowledge Creation. California *Management Review, 40*, 40–54. ISSN: 0008-1256.

Pfeffer, J., & Sutton, R. J. (1999). Knowing "What" to Do Is Not Enough: Turning Knowledge into Action. *California Management Review, 42,* 83–108.

Poltrock, S. E., & Grudin, J. (1994). Organizational Obstacles to Interface Design and Development: Two Participant Studies. *ACM Transactions on Computer-Human Interaction (TOCHI), 1,* 52–80. ISSN: 1073-0516.

Riihiaho, S. (2000). *Experiences with usability evaluation methods. Licenciate's thesis. Helsinki University of Technology.*

Rohn, J. A. (2005). Cost-Justifying Usability in Vendor Companies. In R. G. Bias, & D. J. Mayhew (Eds.), *Cost-Justifying Usability. An Update for the Internet Age* (2nd ed.) (pp. 185–213). San Diego, CA: Morgan Kaufmann Publishers. ISBN: 0-12-095811-2.

Rosenbaum, S., Chauncey, E. W., Jokela, T., Rohn, J. A., Smith, T. B., & Vredenburd, K. (2002). Usability in Practice: User Experience Lifecycle – Evolution and Revolution. In *Extended Abstracts of the CHI '02 conference on Human factors in computing systems* (pp. 898–903). New York, NY: ACM Press. ISBN: 1-58113-454-1.

Rosenbaum, S., Rohn, J. A., & Humberg, J. (2000). A Toolkit for Strategic Usability: Results from Workshops, Panels, and Surveys. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 337–344). New York, NY: ACM Press. ISBN: 1-58113-216-6.

Rozanski, P., & Haake, A. (2003). The Many Facets of HCI. In *Proceedings of the 4th conference on Information technology curriculum* (pp. 180–185). New York, NY: ACM Press. ISBN: 1-58113-770-2.

Royce, W. W. (1970). Managing the Development of Large Software Systems: Concepts and Techniques. In *WESCON Technical Papers*. Reprinted in *Proceedings of the Ninth International Conference on Software Engineering,* 1987, pp. 328–338.

Schach, S. R. (2002). Object-Oriented and Classical Software Engineering (5th ed.). McGraw Hill. ISBN: 0-07-239559-1.

Seffah, A., & Adreevskaia, A. (2003). Empowering Software Engineers in Human-Centered Design. In *Proceedings of the 25th International Conference on Software Engineering* (pp. 653–658). Washington, DC: IEEE Computer Society. ISBN: 0-7695-1877-X.

Seffah, A., & Metzker, E. (2004). The Obstacles and Myths of Usability and Software Engineering. *Communications of the ACM, 47,* 71–76. ISSN: 0001-0782.

Siegel, D. A. (2003). The Business Case for User-Centered Design: Increasing Your Power of Persuasion. *Interactions, 10,* 30–36. ISSN: 1072-5520.

Sinkkonen, I., Kuoppala, H., Parkkinen, J., & Vastamäki, R. (2002). *Käytettävyyden psykologia* (2nd ed.). Helsinki: IT Press. ISBN: 951-826-574-7.

Snyder, C. (2003). *Paper prototyping: The Fast and Easy Way to Design and Refine User Interfaces.* U.S.A: Morgan Kaufmann Publishers. ISBN: 1-55860-870-2.

Spellpoint (n.d.). *Vision, Mission and Values*. Retrieved April 19, 2005 from http://www.spellpoint.fi/visionmissionandvalues.

Tate, W. (2004). Applying Learning in Practice. *Industrial and Commercial Training, 36,* 57–60. ISSN: 0019-7858.

UsabilityNet. (2003). *ISO 13407. Human Centred Design Processes for Interactive Systems.* Retrieved January 12, 2005 from http://www.usabilitynet.org/tools/13407stds.htm

Venturi, G., & Troost, J. (2004). Survey on the UCD Integration in the Industry. In *Proceedings of the third Nordic conference on Human-computer interaction* (pp. 449–452). New York, NY: ACM Press. ISBN: 1-58113-857-1.

Vredenburg, K., Mao, J-Y, Smith, P. W., & Carey, T. (2002). A Survey of User-Centered Design Practice. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves* (pp. 471–478). New York, NY: ACM Press. ISBN: 1-58113-453-3.

Wilson, C. E., Rosenbaum, S. (2005). Categories on Return on Investment and Their Practical Implications. In R. G. Bias, & D. J. Mayhew (Eds.), *Cost-Justifying Usability. An Update for the Internet Age* (2nd ed.) (pp. 215–263). San Diego, CA: Morgan Kaufmann Publishers. ISBN: 0-12-095811-2.

Wixon, D. R., Ramey, J., Holtzblatt, K., Beyer, H., Hackos, J, Rosenbaum, S., et al. (2002). Usability in Practice: Field Methods Evolution and Revolution. In *Extended Abstracts of the CHI '02 conference on Human factors in computing systems* (pp. 880–884). New York, NY: ACM Press. ISBN: 1-58113-454-1.

APPENDICES

APPENDIX A – TARGETS FOR DATA COLLECTION

Conceptual UCD knowledge:
To what extent the employees are familiar with basic concepts of usability and UCD. This data was gathered from interviews and questionnaire, and it was qualitative in nature.

Characteristics of working environment:
What is the physical and social working environment at the company? How does it affect the working methods and practices? This data was acquired from observation and interviews, and it was qualitative in nature.

Characteristics of software development process:
How is software development done at the company? What are the phases, lengths, and durations of typical projects? How many employees are involved, what are their roles? This data was gathered mainly from interviews and artefact analysis, and it was both qualitative and quantitative in nature.

UCD methods in software development:
What UCD methods, if any, are used in software development? This data was acquired from interviews, artefact analysis (work lists), and questionnaire, and was mainly qualitative in nature.

Practical UCD abilities:
How employees' UCD abilities show up in practical assignments? What UCD methods are used in design phase? This data was acquired from design exercise, interviews and questionnaire, and it was qualitative in nature.

Attitudes towards usability:
What is employees' attitude towards usability and usability engineering? What is the role of usability in employees' minds? This data was acquired from interviews and questionnaire, and it was both qualitative and quantitative in nature.

APPENDIX B – CURRENT STATE INTERVIEW STRUCTURE

1. Mikä on nykyinen työnkuvasi Spellpointissa?
    1) Osallistutko ohjelmistokehitykseen sen jossakin vaiheessa?
    2) Mitkä ovat omat tehtäväsi tai roolisi ohjelmistokehityksessä?

2. Onko käytettävyys sinulle tuttu termi? Mitä mielestäsi tarkoittaa hyvä käytettävyys? Mitä asioita siihen liittyy?
    1) Mikä tekee tuotteesta hyvän käyttää?
    2) Minkä tuotteiden tai asioiden kohdalla käytettävyydestä voidaan mielestäsi puhua?

3. Kerro, mikä on mielestäsi käyttöliittymä.
    1) Mistä se koostuu, mitä komponentteja, miten käytetään?
    2) Missä kaikessa on käyttöliittymä? Anna esimerkkejä.

4. Oletko suunnitellut käyttöliittymiä? Millaisia? Millaisia ovat ohjelmat, joissa ei ole ollut käyttöliittymää?

5. Millaisia ongelmatilanteita tai haasteita sinulle on tullut työssäsi vastaan käytettävyyteen tai käyttöliittymäsuunnitteluun liittyen? WWW:ssä? Desktop-ohjelmistoissa?

6. Oletko oppinut joitakin hyviä menetelmiä tai suunnittelusääntöjä käyttöliittymäsuunnittelusta? Mistä, millaisia?

7. Muistatko jonkin tilanteen, jossa korjasit ohjelman käytettävyyttä parempaan suuntaan? Millainen tilanne oli, miten huomasit, miten korjasit?

8. Mitkä ovat mielestäsi tärkeitä asioita ohjelmistojen käytettävyydessä?
    1) Mitkä asiat suunnittelussa pitäisi/on tärkeää huomioida?
    2) Miten ne voidaan huomioida suunnittelussa? Teetkö näin?

9. Milloin käytettävyys on mielestäsi otettava huomioon ohjelmistokehityksessä? Millaisten ohjelmistojen yhteydessä erityisesti?

10. Kerro, miten työ etenee työpaikalla, kun uutta ohjelmistoa aletaan suunnitella. (Ajattele vaikka jotakin viimeaikaista projektia)
    1) Mitkä ovat työvaiheet?
    2) Ketkä osallistuvat projektiin?
    3) Millainen määrittely ohjelmistosta tehdään ennen koodauksen aloittamista?
    4) Mitä taustatietoja ohjelmaa varten hankitaan? Saatko riittävästi?
    5) Miten usein ja kuka on yhteydessä tilaajaan?
    6) Miten paljon asiakas määrittelee, millainen ohjelmasta tulee?

7) Näytetäänkö ohjelmaa asiakkaalle suunnitteluvaiheessa? Entä kesken toteutuksen?
8) Saadaanko asiakkaalta kommentteja ennen lopullista toimitusta? Kuinka paljon ohjelmaa muutetaan ensimmäisen version jälkeen asiakkaan kommenttien perusteella?
9) Miten ohjelmaa testataan?
10) Mitä kirjallista materiaalia ohjelmistosta syntyy projektin aikana?

11. Onko tämä mielestäsi hyvä tapa tehdä ohjelmia?
    1) Olisiko jokin tapa parempi? Miksi näin ei tehdä?

12. Tiedätkö mikä ja millainen on vaatimusmäärittely?
    1) Mitä hyötyä niistä on?
    2) Oletko käyttänyt joskus?
    3) Käytetäänkö Spellpointissa? Kuka tehnyt?

13. Oletko itse tai joku muu projektissa tehnyt demoja ennen lopullista toteutusta? Millaisia? Millä työkaluilla?
    1) Missä vaiheessa ohjelman tekoa demoja on tehty?
    2) Kenelle demoja näytetään?

14. Tiedätkö joitakin menetelmiä, joilla käytettävyyttä voidaan testata tai arvioida?
    1) Osaatko kertoa niistä tarkemmin?
    2) Oletko itse osallistunut käyttäjätesteihin/tehnyt arviointeja tms?

15. Onko työpaikalla puhuttu käytettävyydestä?
    1) Milloin, kuinka usein? Kuka on puhunut?
    2) Oletko osallistunut käyttöliittymäkoulutukseen? Mitä muistat, mikä on jäänyt mieleen?
    3) Millainen vaikutus koulutuksella oli työhön tai työntekotapoihisi? Osaatko antaa esimerkkejä?

16. Onko annettu ohjeita miten käytettävyyttä voitaisiin huomioida suunnittelutyössä?
    1) Haluaisitko enemmän?
    2) Mistä asioista?

17. Millainen asenne mielestäsi työpaikalla vallitsee käyttöliittymäsuunnittelun ja käytettävyyden suhteen?
    1) Panostetaanko siihen?
    2) Näkyykö se jotenkin käytännössä?
    3) Mikä on oma näkemyksesi käytettävyydestä?

APPENDIX C – CURRENT STATE QUESTIONNAIRE

Taustakysely Spellpointin työntekijöille

Tämän kyselyn tarkoituksena on kartoittaa Spellpointin osaamistasoa ohjelmistojen käyttäjäkeskeisen suunnittelun osalta. Kyselyllä ei ole tarkoitus mitata yksittäisten työntekijöiden osaamista, eikä tutkimuksen tuloksissa ja johtopäätöksissä raportoida yksittäisten työntekijöiden vastauksia.

Kysymyksissä käytetty asteikko on seuraava
1 = hyvin vähän
5 = hyvin paljon
EOS = En osaa sanoa

Arvioi omaa osaamistasi:
Kuinka hyvin osaat huomioida käyttäjät ja heidän tarpeensa ohjelmistokehityksessä?

huonosti, en juuri ollenkaan — erittäin hyvin

1 2 3 4 5 EOS

Arvioi Spellpointissa olevaa osaamista kokonaisuutena:
Kuinka hyvin yrityksessä osataan huomioida käyttäjät ja heidän tarpeensa ohjelmistokehityksessä?

huonosti, ei juuri ollenkaan — erittäin hyvin

1 2 3 4 5 EOS

Kuinka tärkeä osa ohjelmistoa sen käytettävyys mielestäsi on?

ei tärkeä — hyvin tärkeä

1 2 3 4 5 EOS

Kuinka kiinnostunut olet saamaan lisää tietoa käyttäjäkeskeisestä ohjelmistokehityksestä?

en ole kiinnostunut — hyvin kiinnostunut

1 2 3 4 5 EOS

Kuinka kiinnostunut olet oppimaan uusia menetelmiä ja työtapoja käytettävyyden kehittämiseksi?

en ole kiinnostunut — hyvin kiinnostunut

1 2 3 4 5 EOS

Mainitse asioita, jotka mielestäsi tekevät ohjelmistosta hyvän käyttää.

_____

_____

Mikä yllä mainitsemistasi tekijöistä on mielestäsi tärkein? *Voit valita useamman jos et osaa nimetä yhtä.*

_____

Mitkä seuraavista käyttäjäkeskeiseen suunnitteluun liittyvistä käsitteistä ovat sinulle tuttuja?
Rasti sopiva vaihtoehto tai vaihtoehdot.

Käyttäjäryhmä

- ☐ En tunne.
- ☐ Olen kuullut, mutta en tiedä miten käytetään.
- ☐ Tunnen ja osaan käyttää.
  - ☐ Olen käyttänyt työssäni Spellpointissa. *Voit halutessasi tarkentaa alle.*

    _____
  - ☐ Olen käyttänyt jossain muualla (työssä, opiskelussa). *Voit halutessasi tarkentaa alle.*

    _____

Käyttäjäprofiili

- ☐ En tunne.
- ☐ Olen kuullut, mutta en tiedä miten käytetään.
- ☐ Tunnen ja osaan käyttää.
  - ☐ Olen käyttänyt työssäni Spellpointissa. *Voit halutessasi tarkentaa alle.*

    _____
  - ☐ Olen käyttänyt jossain muualla (työssä, opiskelussa). *Voit halutessasi tarkentaa alle.*

    _____

Käyttötapaus

- ☐ En tunne.
- ☐ Olen kuullut, mutta en tiedä miten käytetään.
- ☐ Tunnen ja osaan käyttää.
  - ☐ Olen käyttänyt työssäni Spellpointissa. *Voit halutessasi tarkentaa alle.*

    _____
  - ☐ Olen käyttänyt jossain muualla (työssä, opiskelussa). *Voit halutessasi tarkentaa alle.*

    _____

Käyttöskenaario

- ☐ En tunne.
- ☐ Olen kuullut, mutta en tiedä miten käytetään.
- ☐ Tunnen ja osaan käyttää.
  - ☐ Olen käyttänyt työssäni Spellpointissa. *Voit halutessasi tarkentaa alle.*

    _____
  - ☐ Olen käyttänyt jossain muualla (työssä, opiskelussa). *Voit halutessasi tarkentaa alle.*

    _____

Ympäristökuvaus

- ☐ En tunne.
- ☐ Olen kuullut, mutta en tiedä miten käytetään.
- ☐ Tunnen ja osaan käyttää.
  - ☐ Olen käyttänyt työssäni Spellpointissa. *Voit halutessasi tarkentaa alle.*

    _____
  - ☐ Olen käyttänyt jossain muualla (työssä, opiskelussa). *Voit halutessasi tarkentaa alle.*

    _____

Kiitos avustasi!

APPENDIX D – INUSE HUMAN-CENTEREDNESS MATURITY MODEL: ATTRIBUTES AND MANAGEMENT ACTIVITIES

Attributes of INUSE maturity levels; four lowest levels (Earthy, 1998).

Level and Title	Attributes	Description
X: Unrecognised	(No indicators)	–
A: Recognised	Problem recognition	The extent to which members of the organisation understand that there is a problem with the quality in use of the systems produced.
	Performed processes	The extent to which processes are performed that provide input that could be used to make the system human-centred.
B: Considered	Quality in use awareness	The extent to which the staff carrying out a process are aware of quality in use as an attribute of the system.
	User focus	The extent to which staff performing processes relating to the user-facing elements of the system take account of the fact that a human being will need to use it.
C: Implemented	User involvement	The extent to which information is elicited from representative users using appropriate techniques throughout the lifecycle.
	Human factors technology	The extent to which human factors methods and techniques are used in or by human-centred processes.
	Human factors skills	The extent to which human factors skills are used in human-centred processes.

Management practices at maturity levels; four lowest levels (Earthy, 1998).

Level and Title	Attributes	Management practices
X: Unrecognised	(No indicators)	(None)
A: Recognised	A.1 Problem recognition	A.1.1 *Problem recognition.* Management and staff are aware that there is a need to improve aspects of the systems under development concerned with their use.
	A.2 Performed processes	A.2.1 *Information collection.* Information is collected which could be used to take account of user requirements.
		A.2.2 *Performance of relevant practices.* Practices are performed which could be used to include information about user requirements in the system or service.
B: Considered	B.1 Quality in use awareness	B.1.1*Quality in use training.* Staff are made aware that quality in use is a particular attribute of a system which can be improved.
		B.1.2 *Human-centred methods training.* Staff are made aware that quality in use is achieved through the use of a series of human-centred processes during the development and support/use of a system.
		B.1.3 *Human-system interaction training.* Staff are made aware that human-centeredness covers the total system, not just the user interface or the physical ergonomics.

Level and Title	Attributes	Management practices
	B.2 User focus	B.2.1 *User consideration training.* Staff are made aware that the needs of the end users of the system should be considered when developing or supporting the system.
		B.2.2 *Context of use training.* Staff are made aware that end users' skills, background and motivation may differ from developers or system support staff.
C: Implemented	C.1 User involvement	C.1.1 *Active involvement of users.* The development process ensures understanding of user needs through user involvement in all development phases.
		C.1.2 *Elicitation of user experience.* The design solution is shown to stakeholders and they are allowed to perform tasks (or simulated tasks).
		C.1.3 *End users define quality-in-use.* Systems are tested using measures of quality in use derived from end users.
		C.1.4 *Continuous evaluation.* Early and continual testing is an essential element of the development methodology. The process is based on the necessity for feedback from users.
	C.2 Human factors technology	C.2.1 *Provide appropriate human-centred methods.* Select and support methods for the elicitation of user input at all stages in the lifecycle.
		C.2.2 *Provide suitable facilities and tools.* Suitable facilities and tools are provided for quality in use activities.
		C.2.3 *Maintain quality in use techniques.* Ensure that methods and techniques are reviewed for suitability and that state-of-the-art user interface technologies are used as appropriate in developing new systems.
	C.3 Human factors skills	C.3.1 *Decide on required skills.* Identify required competencies and plan how to make these available in order to facilitate multi-disciplinary design solutions.
		C.3.2 *Develop appropriate skills.* Development of appropriate skills in human-centred staff either by training or by job experience.
		C.3.3 *Deploy appropriate staff.* Skilled staff are involved and effective in all stages of development as and when required.

APPENDIX E – USABILITY TRAINING AGENDA

First workshop

8:00–8:10 Coffee, day's agenda
8:10–8:30 Informal user interface evaluation exercise
8:30–8:40 Lecture starts
- What is usability? Why usability?

8:40–9:25 Ten design heuristics
9:25–9:35 Break
9:35–10:00 Usability evaluation
  - prototypes
  - heuristic evaluation
  - usability testing

10:00–10:25 User-centred design process; phases and techniques
10:25–10:30 Break
10:30–11:50 Exercise: heuristic evaluation
  10:30–10:35 Instructions
  10:35–11:15 Evaluation
  11:15–11:50 Going through the results
11:50–12:10 Homework and next time's schdule

Second workshop

8:00–8:10 Coffee
8:10–8:25 Day's agenda and instructions
–> Division to groups
8:30–10:00 User profiles and use cases + UI design (transprencies)
- Constructing paper prototype of the most important views
- Selecting test tasks(a couple of most important use cases + right ways to perform)
- Interview questions (A few relevant)

10:00–10:30 Usability testing
- A test user comes from another group

10:30–10:45 Listing of test results and presentation preparations (transparencies)
–> Gathering
Presentations 4 x (15 min + 10 min discussion)
10:50–11:05 Group 1
11:15–11:30 Group 2
11:40–11:55 Group 3
12:05–12:20 Group 4
12:20–12:30 End discussion

APPENDIX F – GROUP INSTRUCTIONS IN THE SECOND WORKSHOP

1. Sihteerin osoitekirja

Vaatealan tukkuyritys Tukkuri Oy:n toimistosihteerille suunnitellaan uutta sähköistä osoitekirjaa vanhan kuluneen paperisen kirjan tilalle. Sihteerin tehtävänä on hoitaa toimitusjohtajan puhelinasiat ja huolehtia jokapäiväisen arjen käytännön järjestelyistä. Sihteerin vanha osoitekirja on täynnä eri yritysten ja muiden tahojen yhteystietoja: muun muassa tavarantoimittajia, asiakkaita, yhteistyökumppaneita, sekä kokoustiloja, ravintoloita ja hotelleja. Sihteeri on kerännyt tietoja kirjaan vuosien varrella. Toisinaan yhteystiedot on saatu käyntikorteilla tai esitteinä, jotka on sellaisenaan sujautettu kirjan väliin. Nykyisin osoitekirja pullistelee kansiensa välissä ja sivujen kääntely on hankalaa.

Uudelta osoitekirjalta sihteeri toivoo helppokäyttöisyyttä ja nopeutta. Usein tiedot on löydettävä kirjasta nopeasti toimitusjohtajan toisinaan seisoessa vieressä odottamassa. Myös uusien tietojen lisäämisen tulisi tapahtua vaivatta.

2. Lounasravintolan kassaohjelma

Ravintola Lounastauko on vaihtamassa vanhan kassakoneensa uuteen tietokoneella toimivaan kassaohjelmaan. Yritys tarjoaa lounaspalveluja ostoskeskuksen yläkerrassa läheisten yritysten työntekijöille, jotka käyvät lounaalla ravintolassa. Ravintolassa työskentelee päivittäin kaksi kokkia sekä kolme kassatyöntekijää, joiden tehtäviin kuuluu lisäksi pöytien siivous sekä yleisestä siisteydestä huolehtiminen. Ravintolan lounastarjontaan kuuluu kaksi lämmintä ruokaa (liha ja kasvis), keitto, salaattipöytä sekä erikoishintainen grillilounas. Lisäksi kassalla on myynnissä pieni valikoima makeisia ja virvoitusjuomia. Ravintola ottaa vastaan maksusuorituksia käteisellä, lounasseteleillä sekä pankki- ja luottokorteilla.

Uudelta kassajärjestelmältä ravintola toivoo helppokäyttöisyyttä ja nopeutta. Kiireisimpään lounasaikaan ravintolassa on usein pitkä jono nälkäisiä asiakkaita, joilla on lyhyt ruokatunti, eivätkä he jaksa odottaa pitkään kassajonossa. Henkilöstön vaihtuvuus on suurta, joten järjestelmän toivotaan olevan myös nopeasti omaksuttavissa.

3. Autokorjaamon ajanvarausjärjestelmä

Maken korjaamo Oy tahtoo siirtää paperisen ajanvarausjärjestelmänsä tietokoneaikaan. Korjaamolla työskentelee kolme automekaanikkoa sekä huoltoesimies, jonka tehtävänä on toimiva työnvalvojana ja asiakaspalvelupäällikkönä. Lisäksi yritys harkitsee palkkaavansa pian neljännen mekaanikon. Korjaamolla on neljä hallipaikkaa, joten kaikki mekaanikot voivat työskennellä samanaikaisesti omilla työpisteillään. Nykyään ajanvaraukset tehdään käsin vihkoon, johon merkitään korjattavan auton rekisterinumero, asiakkaan yhteystiedot sekä varattu huoltoajankohta. Varauksia otetaan vastaan puhelimitse ja paikan päällä, ja ne kirjaa vihkoon henkilö, joka varauksen sattuu ottamaan vastaan: joko huoltoesimies tai joku mekaanikoista.

Uudelta järjestelmältä yritys toivoo helppokäyttöisyyttä ja selkeyttä. Nykyinen varauskirja täyttyy nopeasti ja siihen on hankala tehdä muutoksia, esimerkiksi jos varattu huoltoaika yllättäen peruuntuu. Toisinaan epäselvien merkintöjen vuoksi on sattunut päällekkäisiä varauksia, joiden selvittäminen on ikävää ja aikaa vievää. Uuden järjestelmän toivottaisiin helpottavan erityisesti varausaikojen sopimista, mutta toimivan myös työjärjestyslistana.

4. Hotellin loppulaskutusjärjestelmä

Hotelli Majoitus haluaa modernin laskutusjärjestelmän nopeuttamaan poislähtevien asiakkaiden uloskirjautumista hotellista. Hotellissa on 30 huonetta ja lisäksi asiakkaiden käytössä on eri

maksusta kokoushuone, ravintola, baari, huonepalvelu ja minibaari sekä kuntosali, tenniskenttä ja uima-allas. Nykyisin hotellin vastaanottovirkailija saa paperilla tiedot käytetyistä palveluista erikseen jokaisesta palvelusta ja laskee laskun loppusumman käsin vastaanotossa.

Uudelta järjestelmältä yritys toivoo helppokäyttöisyyttä ja selkeyttä. Uudessa järjestelmässä asiakkaan käyttämät palvelut kirjattaisiin keskitettyyn järjestelmään, josta erittely voidaan myöhemmin tulostaa vastaanotossa. Laskun loppusumma ja erittely tulisi voida muodostaa nopeasti asiakkaan odottaessa tiskillä. Kassatoimintoja järjestelmään ei kuitenkaan tarvitse sisällyttää, sillä hotelli on vastikään hankkinut uuden kassaohjelman.

APPENDIX G – FEEDBACK QUESTIONNAIRE

Palautekysely

Tämän kyselyn tarkoituksena on kerätä palautetta toukokuussa 2005 järjestetystä kaksipäiväisestä käytettävyyskoulutuksesta sekä arvioida Spellpointin osaamistasoa ohjelmistojen käyttäjäkeskeisen suunnittelun osalta koulutuksen jälkeen. Kyselyllä ei ole tarkoitus mitata yksittäisten työntekijöiden osaamista, eikä tutkimuksen tuloksissa ja johtopäätöksissä raportoida yksittäisten työntekijöiden vastauksia.

Kysymyksissä käytetty yleinen asteikko on seuraava:
1 = hyvin vähän
5 = hyvin paljon
EOS = En osaa sanoa

---

1. Osallistuitko Spellpointin käytettävyyskoulutukseen?

☐ Kyllä, molempiin työpajoihin.
☐ Osallistuin vain ensimmäiseen työpajaan 11.5.
☐ Osallistuin vain toiseen työpajaan 18.5.

2. Kuinka koulutus vastasi odotuksiasi? Ympyröi sopiva vaihtoehto.
(Valitse 3 jos koulutus vastasi odotuksiasi.)

Koulutus oli pettymys					Koulutus ylitti odotukset
1	2	3	4	5	EOS

Perustele vastaustasi. *(Jatka tarvittaessa kääntöpuolelle.)*

_____

_____

3. Kuinka hyödyllinen koulutus mielestäsi oli omaa työtäsi ajatellen? Ympyröi sopiva vaihtoehto.

Täysin hyödytön					Erittäin hyödyllinen
1	2	3	4	5	EOS

Perustele vastaustasi. *(Jatka tarvittaessa kääntöpuolelle.)*

_____

_____

4. Mikä koulutuksen sisällöstä oli sinulle uutta asiaa? *(Jatka tarvittaessa kääntöpuolelle.)*

_____

_____

_____

5. Mikä koulutuksen sisällöstä oli sinulle ennestään tuttua? Mistä? *(Jatka tarvittaessa kääntöpuolelle.)*

_____

_____

_____

6. Mikä oli koulutuksen tärkein anti sinulle? *(Jatka tarvittaessa kääntöpuolelle.)*

7. Jäikö asioita, joista olisit halunnut lisää tietoa tai keskustelua? *(Jatka tarvittaessa kääntöpuolelle.)*

8. Oliko koulutuksessa jotakin, mikä ei kiinnostanut sinua tai tuntui hyödyttömältä? Miksi?
(Jatka tarvittaessa kääntöpuolelle)

9. Kuinka koulutuksessa käsiteltävät asiat nivoutuivat mielestäsi aiemmin Spellpointin käyttöliittymäkoulutuksessa esitettyihin asioihin? Ympyröi sopiva vaihtoehto.
(Valitse 3 jos koulutus oli irrallinen, mutta ei varsinaisesti ristiriidassa aiemman kanssa. Valitse EOS, jos et ole osallistunut aiemmin käyttöliittymäkoulutukseen tai et muista mitä siellä käsiteltiin.)

Huonosti, ristiriidassa aiemmin opitun kanssa				Hyvin, tuki ja täydensi aiemmin opittua	
1	2	3	4	5	EOS

Perustele vastaustasi. *(Jatka tarvittaessa kääntöpuolelle.)*

10. Kuinka hyvin työpajatyöskentely sopi mielestäsi käytettävyyskoulutukseen (verrattuna tavalliseen luento-opetukseen)? Ympyröi sopiva vaihtoehto.

Huonosti				Erittäin hyvin	
1	2	3	4	5	EOS

Perustele vastaustasi. *(Jatka tarvittaessa kääntöpuolelle.)*

11. Arvioi koulutuksen vaikutusta osaamiseesi: Kuinka koulutus vaikutti kykyysi ottaa käyttäjät ja käytettävyys huomioon työssäsi? Ympyröi sopiva vaihtoehto.
(Valitse 3 jos koulutuksella ei ollut vaikutusta.)

Osaan selvästi huonommin				Osaan selvästi paremmin	
1	2	3	4	5	EOS

Perustele vastaustasi. *(Jatka tarvittaessa kääntöpuolelle.)*
(Jos valitsit muun kuin 3 tai EOS: millä osa-alueilla, miten erityisesti?)

12. Uskotko koulutuksen vaikuttavan työtapoihisi? Teetkö jotkin asiat jatkossa toisin?
Perustele myös mahdollinen ei-vastaus. *(Jatka tarvittaessa kääntöpuolelle.)*

13. Olisitko halukas osallistumaan myöhemmin lisäkoulutukseen käytettävyydestä ja käyttäjä-keskeisestä ohjelmistokehityksestä? Ympyröi sopiva vaihtoehto.

En ole kiinnostunut				Olen hyvin kiinnostunut	
1	2	3	4	5	EOS

Jos valitsit muun kuin 1 tai EOS:
Mistä aiheesta erityisesti haluaisit lisää tietoa? *(Jatka tarvittaessa kääntöpuolelle.)*

14. Anna yleisarvosana koulutukselle kokonaisuutena.

Epäonnistunut, hyödytön				Erittäin onnistunut, kiinnostava	
1	2	3	4	5	EOS

Perustele vastaustasi. *(Jatka tarvittaessa kääntöpuolelle.)*

15. Mitä seuraavista koulutuksessa käsitellyistä menetelmistä uskot käyttäväsi jatkossa työssäsi? Rasti mielipidettäsi vastaava vaihtoehto. *(ks. menetelmäkuvaukset seuraavalta sivulta)* Miksi / miksi et? *(Jatka tarvittaessa kääntöpuolelle.)*

1. Käyttäjäprofiilit / persoonat

☐ Uskon käyttäväni
☐ En usko käyttäväni

_____
_____
_____

2. Käyttötapaukset

☐ Uskon käyttäväni
☐ En usko käyttäväni

_____
_____
_____

3. Käyttöympäristön kuvaus

☐ Uskon käyttäväni
☐ En usko käyttäväni

_____
_____
_____

4. Paperiprototyyppi

☐ Uskon käyttäväni
☐ En usko käyttäväni

_____
_____
_____

5. Heuristinen arviointi

☐ Uskon käyttäväni
☐ En usko käyttäväni

_____
_____
_____

6. Käytettävyystestaus käyttäjän kanssa

☐ Uskon käyttäväni
☐ En usko käyttäväni

_____
_____
_____

Kiitos vastauksestasi!

APPENDIX H – DESIGN EXERCISE AND QUESTIONNAIRE

Suunnittelutehtävä

Tehtävänäsi on suunnitella Internetiin suomalaisen pankin kotisivuille **valuuttamuunnin**, jolla voi muuntaa rahamääriä valuutasta toiseen. Mieti, millainen muunnin olisi, mitä sillä voi tehdä ja miten sitä käytetään. Älä keskity tekniseen toteutukseen. Suunnittelutyön lopputuloksena toivotaan piirros valuuttamuuntimen käyttöliittymästä. Ohjeistus on tarkoituksella väljä; voit tehdä muuntimesta sellaisen kuin itse katsot sopivaksi. Suunnitelman laajuus ei ole olennainen arviointikriteeri.

Tee suunnitelmasi oheisille paperiarkeille ja palauta kaikki suunnittelutyön aikana syntynyt materiaali. Jos paperiarkit loppuvat kesken, voit jatkaa irtoarkeille. Laitathan nimesi myös mahdollisiin irtopapereihin.

Sopiva aika suunnitteluun on noin 30–60 min. Vastaathan suunnittelun jälkeen vielä suunnitelmaa koskeviin muutamaan kysymykseen.

Suunnittelutehtävä

1. Kuinka kauan käytit aikaa suunnitteluun? _____ minuuttia

2. Perustele lyhyesti suunnitelmaasi. Mikä on keskeistä? Mihin suunnitelmassa pitäisi kiinnittää huomiota? *(Jatka tarvittaessa kääntöpuolelle.)*

3. Kerro suunnittelusi etenemisestä: Miten lähestyit tehtävänantoa? Miten päädyit esittämääsi ratkaisuun? *(Jatka tarvittaessa kääntöpuolelle.)*

4. Mitkä ovat mielestäsi suunnitelmasi vahvuudet? *(Jatka tarvittaessa kääntöpuolelle.)*

5. Onko suunnitelmassasi mielestäsi joitakin ongelmia? *(Jatka tarvittaessa kääntöpuolelle.)*

6. Oletko keskustellut valuuttamuunnin-tehtävästä työtovereidesi kanssa maaliskuussa tehdyn nykytilan kartoituksen jälkeen?

   ☐ Kyllä
   ☐ En

   Jos vastasit kyllä, miten arvioit keskustelunne vaikuttaneen suunnitteluratkaisuusi?

Kiitos vastauksestasi!

APPENDIX I – DESIGN EXERCISE COMPARISONS

The table below summarizes the features found in the user interface designs done before and after the usability training. The first column describes the feature, and the second two columns state the number of designs where the specified feature was found when considering all designs returned from the first and second rounds. The last two columns on the right only include those eleven employees that made the design exercise twice that is, in the current state analysis and after the usability training.

Design feature	**Before training (N=13)**	**After training (N=13)**	**Before training (N=11)**	**After training (N=11)**
Named target user group(s)	0	3	0	3
Written user profiles	3	5	3	4
Written use cases	3	5	3	4
User instruction text in design	4	6	4	5
Additional Help link	4	2	3	2
User language selection	2	3	2	3
Currency search by country name	6	8	5	7
Currency search with a map	1	4	1	4
Navigation path visible	0	2	0	2
Logout or exit function	1	4	1	3
Shortcuts to currencies	4	4	4	4
Automatic update as user types	7	9	6	7
Two-way conversion	4	7	4	5