FAKULTÄT
FÜR !NFORMATIK

Faculty of Informatics

# Evaluation of a novel approach for requirements engineering

## An empirical study on laser engraving machines

DIPLOMARBEIT

zur Erlangung des akademischen Grades

**Diplom-Ingenieur**

im Rahmen des Studiums

**Software Engineering and Internet Computing**

eingereicht von

**Gernot Rumpold**
Matrikelnummer 0728159

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Ass.-Prof. Mag. Michael FILZMOSER, PhD
Mitwirkung: Dipl. -Ing. Siegfried SHARMA

Wien, 27.01.2014

(Unterschrift Verfasser) (Unterschrift Betreuung)

FAKULTÄT
FÜR !NFORMATIK

Faculty of Informatics

# Evaluation of a novel approach for requirements engineering

## An empirical study on laser engraving machines

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

**Diplom-Ingenieur**

in

**Software Engineering and Internet Computing**

by

**Gernot Rumpold**
Registration Number 0728159

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Ass.-Prof. Mag. Michael FILZMOSER, PhD
Assistance: Dipl. -Ing. Siegfried SHARMA

Vienna, 27.01.2014

(Signature of Author) (Signature of Advisor)

# Erklärung zur Verfassung der Arbeit

Gernot Rumpold
Garbergasse 12, 1060 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum)

(Unterschrift Verfasser)

# Acknowledgements

I would like to express my appreciation to my advisors Ass.-Prof. Mag. Michael Filzmoser, PhD and DI Siegfried Sharma. Thank you for your guidance throughout the whole process and your feedback and insights. Special thanks go to Siegfried Sharma sacrificing his free time for supervision and feedback even when he had left Vienna University of Technology to study abroad.

My gratitude also goes to the research team of the Ge:MMaS project for allowing me to use their empirical data for my work. It was also a great expirience to participate in project meetings of a research project in the very final phase and it provided me valuable insights to a researchers way of working.

I also want to thank my employer rubicon IT GmbH, especially my colleagues and team leads for providing me the flexibility that was neccessary to finish my thesis and also for their feedback and the valuable discussions.

A final thanks goes to all the people that enriched my opinions and ideas and motivated me by various discussions and reviews of this thesis - writing this thesis would have beeen half the fun without these dicussions.

# Abstract

In the field of product design and development requirements engineering gained more and more attention over the last decade in various disciplines. During the last years the importance of direct input of users in the design process was recognized which lead to user centered design approaches. On the other hand product implementation and requirements management require structured approaches to handle requirement specifications over the whole product lifecycle.
The way a user interacts with a product is individually and differs for each person. Nevertheless it is a knwon practice to classify users in user-groups. By nowadays common approaches for requirements engineering, the individual needs of user-groups are not handeled adequately which leads to discrimination of subsets of users.

The matter of this work is to show, that differences in the demands that several user-groups make on a product exist. Based on this hypothesis it becomes neccessary to define a requirements engineering process that is capable of handling different requirements from several user-groups to avoid discrimination. Based on an empirical study which was conducted as part of the Ge:MMas project, it is shown that the assumption of different requirements among gender-groups exist. Furthermore a process to handle requirements of differnet groups of users is described and evaluated based on the data of the empirical study.

The results show, that existing approaches for requirements engineering differ among technical disciplines, ranging from quite unstructured approaches in the field of agile development to strictly structured approaches like functional analysis in the field of construction. The proposed approach, called *function-requirements-elevation*, combines elements from both methods. It applies principles that are stated by User Centered Design (UCD) by allowing user to directly state their demands and collecting them in a semi-structured way. The semi-structured requirements of multiple user-groups are transformed into a sound and structured requirements specification that contains the consolidated requirements of all user-groups.
The evaluation of the process based on data gathered during the empirical study of Ge:MMaS shows that the approach can be applied in practice. Furthermore it is shown that differences between the requirements of user-groups classified by gender exists. This substantiates the need of a requirements engineering process that is capable of handling this diverse requirements.

# Kurzfassung

Im Gebiet Produktdesign und Produktentwicklung hat der Bereich der Anforderungsanalyse in vielen Disziplinen in den letzten Jahrzehnten an Bedeutung gewonnen. In den letzten Jahren wurde auch die Wichtigkeit der Einbeziehung von Anwendern erkannt was zu anwenderzentrierten Designprozessen führte. Andererseits erfordern Produktimplementierung und Anforderungsmanagement strukturierte Ansätze um Anforderungsspezifikationen über den gesamten Lebenszyklus der Produkte zu handhaben.

Die Art und Weise in der ein Anwender mit einem Produkt interagiert ist individuell verschieden, nichtsdestotrotz können Benutzer in Benutzergruppen klassifiziert werden. Gängige Ansätze der Anforderungsanalyse sind nicht imstande die individuellen Bedürfnisse verschiedener Benutzergruppen adequat abzubilden, was zu Diskriminierungen von Anwendern führt.

Gegenstand dieser Arbeit ist es zu zeigen, dass Unterschiede zwischen den Ansprüchen verschiedener Benutzergruppen existieren. Ausgehend von dieser Hypothese wird es notwendig einen Prozess zur Anforderungsanalyse zu definieren der imstande ist unterschiedliche Bedürfnisse mehrerer Benutzergruppen zu handhaben. Basierend auf einer empirischen Studie die im Zuge des Forschungsprojektes Ge:MMaS durgeführt wurde wird gezeigt, dass die Annahme unterschiedlicher Anforderungen durch verschiedene Benutzergruppen zutrifft. Weiters wird ein Prozess zum Umgang mit diesen heterogenen Anforderungen vorgeschlagen und anhand der Daten aus der empirischen Studie evaluiert.

Die Ergebnisse zeigen, dass die existierenden Ansätze zur Anforderungsanalyse sich zwischen verschiedenen technischen Disziplinen unterscheiden. Sie reichen von weitgehend unstrukturierten Vorgehensweisen in der agilen Entwicklung bis hin zu streng strukturierten Methoden wie Funktionenanalyse im Bereich der Konstruktion. Der vorgeschlagene Ansatz, genannt *Funktionen-Anforderungs-Erhebung*, kombiniert Elemente beider Methoden. Es werden Prinzipien aus dem Bereich des anwenderzentrierten Designs (UCD) angewandt die Anwendern die Möglichkeit bieten ihre Anforderungen direkt in den Prozess einfliessen zu lassen. Diese werden in einer semi-strukturierten Darstellung gesammelt. Die semi-strukturierten Anforderungen verschiedener Benutzergruppen werden in eine vollständige und strukturierte Anforderungsspezifikation übergeführt, welche die konsolidierten Anforderungen aller Benutzergruppen enthält.

Die Evaluierung des Prozesses basierend auf Daten der empirischen Studie aus Ge:MMaS zeigt, dass der vorgeschlagene Prozess in der Praxis anwendbar ist. Weiters wird gezeigt, dass Unterschiede zwischen den Anfoderungen von Benutzergruppen die nach Geschlecht klassifiziert sind existieren. Dies untermauert den Bedarf an einem Prozess der die adequate Handhabung dieser unterschiedlichen Anforderungen ermöglicht.

# Contents

# CHAPTER 1

# Introduction

## 1.1 Motivation

The efficiency of today's machines as well as software systems is mainly affected by the design of their human-machine-interfaces (HMIs). The quality of the HMI defines the way the user interacts with the system as well as the set of of all available features that is used by the machine operator. Due to the increasing complexity of today's systems and the amount of features available, the complexity of user interfaces (UIs) is increased as well. This leads to an amendable usage of those systems as some of the available features are either not known or simply not used by the systems consumers.

The specific subset of available features, that is employed individually differs. One of the many factors that influence the set of features that a user explores is the user's gender. The term gender in this case refers to the social gender, which is defined by the social background and personality of a person, and is not limited to their sex [54]. The differences between those gender groups and their requirements lead to significant inequities in the working ergonomics of individuals and therefore their productivity.

To allow a more efficient utilization of today's machines as well as software systems, it is necessary to design systems in a way, that provide a maximum of usability for all the various groups of users. As a precondition for system design with respect to the various gender groups among the systems consumers, the influence of the users gender to their requirements needs to be analyzed.

Over a products life-cycle the users expectations to the product change continuously. This leads to the fact, that requirements need to be managed over the whole product life-cycle and the product needs to be adapted over time to match the mutated requirements. In case of the gap between the current systems functionality and the current user needs becomes too big it can be a valuable course of action to redefine the requirements list completely instead of gradually adapting to new situations by small changes to the system. This situation is likely to occur for products that have a long life-cycle or as a consequence of dramatic changes of the environment in which the product operates. One of those dramatic changes to products environments has emerged over the last years when gender aware product design gained popularity.

This facts raise the need for a process of requirements re-engineering which allows for creating a new set of requirements to match the current situation of all affected user groups, although taking into account the current state of the product and the information that is already available from preceding requirements management activities.

## 1.2 Problem Statement

Based on an empirical study conducted as part of the Ge:MMaS [1] research project on users of laser engraving machines by Trotec GmbH, an Austrian manufacturer, the gender specific requirements are assumed to be a crucial factor for improving the efficiency of the production process.

As a part of this project the relevant user groups of the systems have been identified as *female users*, *male users* and *managers*. Based on this distinction the individual requirements for each group have been determined in a qualitative manner in terms of workshops and interviews.

Building on this qualitative data a process for re-engineering the human-machine-interface (HMI) is to be developed, which takes is capable of handling different requirements from multiple user-groups although providing a sound and structured representation that serves a basis for product implementation. The resulting list of requirements should serve as a basis for developing a guideline to define the importance of various product components and the HMI of the laser engraving machines to enable the design of a new product line.

## 1.3 Method

For the definition of a novel requirements engineering approach a literature analysis is performed first. Current methodologies of requirements engineering in various disciplines are examined, where a focus is put on user centered approaches for requirements gathering as well as structured representation of requirements catalogs.

Based in this literature evaluation novel process for requirements engineering is described in enough detail to allow for a practical implementation of the proposed approach. To prove the validity and practical relevance of this approach, the qualitative data gathered in user workshops during the Ge:MMaS project is used. The described process is executed on this data which leads to a structured catalog of requirements that covers the needs of all the defined user groups and serves as a basis for subsequent product implementation or improvement.

[1] Project-title: Genderspezifische Anforderungen für Entwicklung neuer Maschinen unter Berücksichtigung der Mensch Maschine Schnittstelle
Supported by Österreichische Forschungsförderungsgesellschaft (FFG)
Projectnumber: 826182

## 1.4 Structure

The reminder of the work is structured in two main parts, where the first one is about theoretical analysis and the second one covers the empirical study and practical evaluation of the developed approach. Considering the theoretical analysis a first chapter examines User Centered Design by defining its most important principles and motivation to apply these techniques. Handling of user-groups and integration of user centered approaches to larger processes is discussed as well. The following chapter covers the discipline of Requirements Engineering and states its importance for product development. Key factors like team constellation are examined and nowadays most common techniques are analyzed. The third chapter of literature research covers function analysis which, in contrast to user centered design, is a very structured approach for handling requirements. Based on its history and the definition of terminology current fields of application and methods for practical application are described in detail.

Based on the three mentioned chapters of related work a novel approach of requirements engineering is developed which combines user centered techniques with the structuring approaches from function analysis. The whole approach takes into account the findings about current requirements engineering techniques as described in the previous chapters.

The second part aims at proving the practical applicability of the developed approach for requirements engineering. To do so, each of the defined steps is executed on data gathered in the context of the Ge:MMaS project. The findings an experiences of each phase when applied to real world data are discussed and if necessary, the process will be adapted partly. Finally the results evolving from this empirical study are used to provide a basis for product implementation and to compare the requirements of the various groups. This will one the one hand show, that considering the requirements of various user-groups separately is reasonable and on the other hand that the proposed approach is capable of handling this different requirements.

In a final discussion the findings of the empirical study as well as the practical applicability of the proposed approach are evaluated.

# Part I

# Theory

CHAPTER 2

# User Centered Design

Each product that is created to be operated by a human being requires some kind of Human Machine Interface (HMI). Obviously most kind of products are designed to interact with humans in some way, although the type of operator ranges from specially trained personnel to inexperienced end users. Due to this fact, the design of an adequate HMI is crucial for most machines. There are several methodologies for designing HMIs where some of them are mainly based on technical requirements and others that are focusing on the needs of the products users. One of those user oriented approaches, that gained wide acceptance over the last years is the principle of User Centered Design (UCD).

User Centered Design is a broad term which is used to describe design approaches that enforce participation of the products target group in the process of HMI development. In UCD users are seen as development partners and not only as customers of the product to be designed. This is the base of creating products that are easy to use and support the users intuitive workflow and handling of the product, which leads to increased efficiency and user satisfaction.

## 2.1 Definition and Origination

The evolvement of UCD in 1980s can be seen as the fourth wave of rational need driven design [39]. Therefore UCD was developed by criticism of previous design approaches. According to Keinonen, there are 3 preceding important schools of design techniques.

**Bauhaus Design** The design methodology at this time was mainly focusing on *type-forms*, which were meant to be ideal solutions for parts of products or challenges that would fit the needs of all users ideally. Those kind of template solutions were defined based on rational needs like mass manufacturing and wide applicability. Due to this, ’soft-facts’ like design and variation of style became unnecessary. To achieve this goal, Bauhaus established a form of language including the ’correct’ shapes [84]. Summing up it can be said, that Bauhaus was more focusing on good and pure design over the user’s reality and individual preferences. Nevertheless, this movement put the relation between a product's purpose and form into focus at that time [39].

**The science of the artificial** During 1960s research focused on techniques on non-coincidental design from the methodological perspective of design, which was referred to as *The science of the artificial* [70]. The field of design became accepted as a field of science in this decade, applying approaches of science, technology and rationalism.

**Design for the real world** The 3rd wave of rational driven design criticized the previous lack of social responsibility of design. The main message was on focusing *real world problems* over sophisticated and inflexible methodologies as described in [84]. At this time barrier-free design and taking into account all types of users when designing a product became popular. Furthermore environmental sustainability and focusing on the user's need over artificial created wants came to the fore.

The next consequence of this eras of design was User Centered Design in 1980s. In contrast to previous waves, UCD was born from industry's needs and focused on transforming complex techniques that originate from professional fields for real world applicability and laypersons. The movement of UCD applied research results from the fields of cognitive science and psychology to model the human-machine interaction in quasi-scientific settings. [39]

The term *User Centered Design* originated in the 1980s in the research laboratory of the cognitive psychologist Donald Norman at the university of California [3] [60]. Norman identified three conceptual models of a product, which are the *design model*, the *user's model* and the *system image*. The design model is the abstraction that the designer has in mind, while the user's model is what users develop to explain how they would like the product to be like. In the best case those models are equivalent. Both models are mapped to the system image, which describes how the concrete product behaves in terms of its physical appearance, responses, its operation as well as manuals and instructions [58].

In conservative product design, the designer develops a design model based on his understanding of the products usage and maps this model to a concrete product implementation. This design model is mainly driven by technical constraints and results that need to be reached using the product, at the same time it lacks mapping of users workflows and consideration of the environment in which the product is used.
Based on this implementation, the user has to find a way to map the system image that was implemented by the designer based on this design model to his user's model. In case the design model and the user's model are very similar this mapping is easy for the user, but every mapping done by the designer that is not intuitive from the users perspective causes usability issues.

Since the 1980s the idea of UCD got widely adopted in various domains of product design and development. Several specific techniques evolved from the basic principles and different understandings of User Centered Design where developed. Today, User Centered Design is used in every field of product design, although it is often referred to by different terms like *Usability engineering*, *User Interaction Design*, *Human Machine Interface Design* and others. Since its origination a lot of research has been done in the field of UCD and the basic methodology was further operationalised and optimized by others in different domains of development [82]. Nevertheless, surveys among companies from across several industrial domains revealed that several of the ideas that evolved from research are not effective to be applied in practice [83]. As a modernized definition of the term UCD to be used in the questionnaire of a survey, User Centered Design was described as follows:

> „UCD is herein considered, in a broad sense, the practice of the following principles, the active involvement of users for a clear understanding of user and task requirements, iterative design and evaluation, and a multi-disciplinary approach.“ [83]

This definition is a good description of UCD as it was seen at the very beginning of the 21st century. During the last decade UCD has undergone further evolvement, especially in the field of ICT. It can be seen as the base of the large field of *Interface and Interaction Design* and caused the development of several techniques for *Usability Testing*. Furthermore numerous methodologies for product design with comprehensive participation of users evolved, which are united in the field of *Participatory Design*.

Nowadays, User Centered Design is defined by several standards e.g. [36]. This standard refers to UCD using the more general term *Human Centered Design* and gives the following definition:

> Approach for the design and development of systems, which aims to make interactive systems more usable, by focusing on the use of the system and applying knowledge and techniques from the fields of ergonomics and fitness for use. [1] [36]

## 2.2 Principles of User Centered Design

User Centered Design became widely used after the publication of [60], which was published by Norman and the psychologist Stephen Draper in 1986. Norman continued research in the field of User Centered Design and published [58] in 1988. The book was reissued in 2002 and is still one of the central publications in the field of User Interaction Design and Usability. It had in the past and still has remarkable impact to the practical application of UCD as it states methods and techniques how the basics of the scientific method of User Centered Design can be applied in real world projects. The book [59] offers four basic principles regarding user-friendly design:

**Make it easy to determine what actions are possible at any moment**

A designer has to make sure that the user is able to figure out what to do at any time. Recognizing possible actions should be possible without further instructions, labels or training. Each further explanation that is done should cause the user to react like „Of course“ or „Yes, I see.“. If it becomes hard for the user to remember those explanations and actions the design has failed.

[1] Translated from german: Herangehensweise bei der Gestaltung und Entwicklung von Systemen, die darauf abzielt, interaktive Systeme gebrauchstauglicher zu machen, indem sie sich auf die Verwendung des Systems konzentriert und Kenntnisse und Techniken aus den Bereichen der Arbeitswissenschaft/Ergonomie und der Gebrauchstauglichkeit anwendet

**Make things visible**

Making things visible includes the conceptual model of the system as well as alternative actions and the results of actions.

The overall concept of the product or at least the part of it that is essential for the user in a specific situation should be intuitive for the user. This means e.g. that the users knows which point of his workflow he is currently at and to be able to imagine what the next steps will be like without seeing them.

**Make it easy to evaluate the current state of the system**

The user should have the possibility to know about the systems state at any time and to forecast what will happen next. If the user is surprised by the systems behavior in a negative way, this goal is not met.

**Follow natural mappings**

Natural mappings should be followed between intentions and the required actions; between actions and the resulting effect; and between the information that is visible and the interpretation of the system state

The designer should implement the system in way, that allows the user to find a mapping from his personal abstract model of the system to the real system image easily. This means workflows and mappings of states of the system should be done in a way that is related to the environment which the product is used in. This mapping should be intuitive from the users perspective to allow him to understand the product based on his present knowledge of their working domain.

Adhering to those principles basically puts the user in the center of product design. Instead of designing a product based on his own opinions (based on the design model), the designer focuses on facilitating the task for the user, which means to understand the users perspective (the user's model). The main goal is to ensure that the user can utilize the product as intended with a minimum of learning effort. Although, it is not enough to tell designers to build products in a way that allows intuitive usage and to see things from the user's perspective. For that reason, Norman suggested the following 7 guidelines for designers [59]:

**Use both knowledge in the world an knowledge in the head**

By building conceptual models, write manuals that are easily understood and that are written before the design is implemented.

**Simplify the structure of tasks**

Make sure not to overload the short-term memory or the long-term memory of the user. On average the user is able to remember five things at a time. Make sure the task is consistent and provide mental aids for easy retrieval of information from long-term memory. Make sure the user has control over the task.

**Bridge the gulfs of Execution and Evaluation**

The user should be able to figure out the use of an object by seeing the right buttons or devices for executing an operation.

**Get the mappings right**

One way to make things understandable is to use graphics.

**Exploit the power of constraints**

This includes natural as well as artificial constraints in order to give the user the feeling that there is one thing to do.

**Design for error**

Plan for any possible error that can be made; this way the user will be allowed the option of recovery from any possible error made.

**When all else fails, standardize**

Create and international standard if something cannot be designed without arbitrary mappings.

Those principles where stated in similar form later on by the computer scientist Ben Shneiderman as eight golden rules and they were used by the engineer Jacob Nielsen to produce heuristics for usability engineering [3].

Although this principles are the base of the large field of User Centered Design, the need to be adapted tor nowadays needs as the whole User Centered Design changed over the last decades. Today's important cornerstones are stated in several standards are are widely similar to those principles defined by Norman in 1980s, although they enrich these by better applicability to today's development processes and refer to specialized sub-disciplines that have not been established when Norman stated his basic principles. A modern definition of key principles for Human Centered Design has been developed by ISO. Thus, independent from the operative process, the distribution of responsibilities and relevant roles, a human centered design should follow the following conventions: [36]

**Design is based on broad understanding of users, tasks and working environments**

The design of products, systems and services should consider the people that will use them as well as further stakeholders that will be affected directly or indirectly. This makes it necessary to identify all the relevant stakeholders, otherwise this is one of the main reasons for a products failure. The fitness of use depends on the context in which it is used, which needs to be taken into account when identifying the stakeholders.

**Users are involved during design and development**

Involvement of users during system design is a valuable source of information about the later usage of the system. Participation of users can be achieved directly via letting them participate in during the design phase or indirectly by gathering information from them. In any case, the involved persons should cover the full spectrum of later users. An active participation of users in a products design raises the later acceptance of the product and engagement of users.

**Continuous refinement of design solutions based on user-centric evaluation**

Feedback of users is a essential source of information in Human Centered Design and the improvements that are performed based on this feedback drastically reduce the risk of designing a system that does not fir the users needs or organizational requirements. This kind of assessment allows to test interim design solutions in real world scenarios and leads to step wise improvements. In the case of final approval user feedback can ensure that the requirements have been met.

**The process is iterative**

An adequate solution for an interactive system can usually not be reached without iterations. Therefore iterations should be used to eliminate uncertainties step by step during system design. The complexity of human-machine interfaces causes the problem, that it is impossible to define a sound and detailed description of the system at the beginning of the design phase. Many requirements of various stakeholders arise during the development process when it becomes easier for users to articulate their needs in respect to possible design variations.

**During design, the whole User Experience is considered**

User Experience arises out of presentation, functionality, interactive behavior and supporting resources of an interactive system. Furthermore previous experiences, approaches, habits and personality of the user influence the User Experience. Therefore it is not sufficient to design a product in a way that it is most easy to use, but personal goals of the user have to be considered as well as job satisfaction and deletion of monotony. To align design towards User Experience several aspects need to be considered, e.g. organizational impacts, training, long-term use, packaging and others.

**Interdisciplinary skills and perspectives are represented in the design team**

Design-teams do not have to be very large, although they need to be staffed interdisciplinary. Important areas of competence therefore are e.g. ergonomics, accessibility, user-research, domain knowledge and business analysis. Depending on the specific situation and project, there may be further useful or necessary experts to be present in the team. In general, projects profit from the additional creativity and the collective base of knowledge that arises from interdisciplinary teams. Furthermore it deepens the understanding of specialists for the limitations and technical demands of different domains.

Summing up it can be said that the basic values of User Centered Design as it was initially described by Norman in 1980s are still valid and reach upon today's definition and understanding of this specific field of product design. Nevertheless several adaptions have been that are necessary to integrate the practices of User Centered Design into nowadays process models and development environments. Furthermore some practices that emerged as important for successful design have lead to independent fields of proficiency and research, e.g. Usability Testing and practices for continuous evaluation and user involvement like participatory design and iterative development processes.

### Usability Testing

Usability Testing summarizes several techniques and methodologies to gain information about how users utilize a product, their satisfaction while using the product, ergonomic aspects of usage behaviors as well as accessibility of products. Not only direct feedback from users is conducted in form of questionnaires and interviews, but also analysis of their behavior while using the product can provide insights in needs and possible improvements that the users are not aware of knowingly. Research on Usability Testing provided various ways of generating insights about the ways products are used and revealing undetected user needs and room for improvement by focusing on user needs, relying on empirical measurement and employing iterative design [3]. Historically, usability tests have been carried out in laboratories under the control and inspection of experts in user-interface design and testing. The methods used often required extensive technical equipment and complicated setups, where in many cases the designers observed the testers unnoticed [3]. This lead to the fact, that usability tests where hardly implemented in practical projects and people relied on generalized findings and guidelines that have been developed as a result of those scientific studies.

Nowadays, Usability Testing plays a central role in User Centered Design [3], as it allows designers to improve the user interfaces of a product iteratively, assuming that there are techniques that allow for testing product design with users in an easy and low-cost manner. Several techniques to do so have been developed over the last years, not only because of the wide adoption of UCD. Experience and investigation on customer satisfaction showed, that once the stakeholders have been identified and their needs have been evaluated, designers can develop alternative design solutions that users can evaluate during the process of product design and implementation [3]. The field of approaches ranges from simple paper prototypes [72] over so called *wizard of oz* approaches to complete process models like rapid prototyping [13] and usability tests in professional environments. To ensure continuous user satisfaction and product improvement, evaluation has to continue even after the product is released, e.g. in terms of interviews and focus group discussions [40] [3].

Overall Usability Testing is a self-contained field of scientific research to identify usability patterns and develop new techniques of usability testing. Nevertheless it is an essential part of User Centered Design and is nowadays widely applied in practice as it offers the possibility of ensuring to meet the users requirements.

## 2.3 Motivation

The goal of each product designer is to create a product that makes the user feel good while using it. Products that satisfy users are more sustainable in the market and therefore increase better revenues for the product vendor. In several areas like retail market, customers are willing to to pay a higher amount for well designed products and systems [36]. Beside that economic reasoning, products that are convenient to use, increase the efficiency and productivity of users in several ways. Obviously, products that fit into the user's intuitive workflow require less learning efforts and help to speed up or ease the tasks that have to be fulfilled by the user. Furthermore, products that cause positive moods for their users can have motivational effects on them. This

means, if a user is satisfied by a product, because it is easy to operate and help him in fulfilling his task, he may draw intrinsic motivation from that, which positively affects the user's overall productivity .
For satisfying a user it is crucial for products to be usable in an intuitive way that fits established workflows. Products need to integrate into the environmental established workflow that the operator is used to or, if it introduces new workflows, integrate into the workflow as the user would imagine it intuitively. If a product causes the user to change his way of fulfilling tasks or his operational procedure due to constraints caused by the product, it will not satisfy the user and fail to generate a positive working atmosphere. Therefore, the concrete benefit of User Centered Design is that it allows for creating products that integrate smoothly into everyday life an working routines. This means that humans get supported by those products without the need of adapting to it - it simply eases daily life without any necessary investment for getting this easement.

From an economic point of view the benefits of applying User Centered Design are not the obvious as they are hard to measure. There are several soft facts as it is important to understand the needs for potential new and innovative products. Doing so is one of the central function of marketing research as such an understanding is clearly an essential input to the development process of new products [78]. Beside that, Kohne et. al. stated several further economical driven benefits of User Centered Design as well as ways to measure them in a quantitative manner, which are divided in producers benefits and customer's benefits in [42].
The motivation for users and their organization to participate in User Centered Design and Development of a product therefore is to *improve operational efficiency* of the product. Although participation in product development is likely to cause additional expenses on the short hand, this effort pays back later on. A reduced number of errors and overall operation time increases the operational efficiency of the user's organization. The cost of operation time is a crucial factor as it includes the number of people working with the product and their working hours, as well as unit price. Therefore this part of benefits that increases operational efficiency can be measured as quantitative data [42].
Benefits for the development organization are driven by numerous factors, where a substantial of the is *reduced development cost*. This reduction is hard to measure directly, because comparing the cost of suing UCD and not using it is usually impossible on the same project. To make this benefits visible, developers may be asked to estimate the reduction of cost by using UCD in terms of development time required to implement a product of the same quality without according to principles of UCD. This estimations and real project costs can be compared quantitatively afterward.
A further often mentioned advantage is the improved quality of the product as usability is improved by executing UCD. Usability metrics are effectiveness, efficiency and satisfaction, which can be shown in user test experiments in terms of operation time and number of errors. This is especially interesting to measure, if a previously implemented product has been redesign using UCD.
Finally UCD leads to *increased sales volume and profits*, which is the most attractive property of UCD for managers. The investment in UCD as well as the amount of orders or sales volume can be shown as quantitative data for an organization. Although, this data can be misleading as

there are several external factors that can increase sales volume, trends show that applying UCD techniques increases sales volume and profits [42].

In a real world study UCD methods like observation were used to find problems of the end-user's perspective. UCD professionals proposed a new concept about the customer's facilities and system based on those insights, which lead to receiving an order due to its potential to increase the customer's business value. After implementation the product built using UCD techniques replaced a competitors system at the customer's site and order amount was written as quantitative beside quantitative benefits in terms of customer's comments in which they recognize the concept made by UCD activities. On the quantitative side, the development cost could be reduced by the cost that 10 people worked in 2-3 months, which was furthermore strongly felt by the developers. Additionally, for one application the time required for one user to send and organize e-mails within the designed application was reduced by about 8 seconds. For the customer, this makes increased operational efficiency of 800 hours a year. Details on the concrete project situation as well as numbers are stated in [42].

### Consequences

If customers are able to utilize products without additional support, this decreases the cost for customer support and consultancy as well as it widens the field of potential users to new groups of society that would not be able to use the product otherwise. In most countries employers as well as suppliers are legally obligated to protect users from threats to their health and safety. Human centered practices can reduce those risks and ensure the barrier free access to the product without discrimination of specific groups of people [36]. According to standards, applying human centered practices has the following positive consequences:

- Increased productivity of users and economic efficiency of organizations
- Reduced cost for training due to easier understanding of the product
- Increased fitness for purpose for people with a larger band of capabilities and therefore improved accessibility
- Improvement of user experience
- Reduction of discomfort and stress
- Providing competitive advantage, e.g. by sharpening the brand image
- Contribution to sustainability goals

Summing up, several positive consequences of the application are stated in standards, where some of them are even legally required or positively affect social and ecological desirable. Most of those consequences result from the overall life-cycle-cost of a product, system or service including conception, design, implementation, operation, utilization and finally abandonment [36].

### Impact

User Centered Design causes impacts for the developing organization as well as the customer's, which are widely agreed to be mainly positive, although they can be analyzed and stated distinct of each other to illustrate benefits for specific stakeholders [42]. The main negative impact to be mentioned, is an increased development effort for customers for the exchange of less required support and increased efficiency. On the other hand research findings yield that UCD methods generally are considered to improve a product's usefulness and usability, although organizations differ remarkably in the degree of adoption of UCD methods [83]. The positive impacts of applying UCD to a product development process are often not seen or underestimated. Case studies showed, that especially developers that have no experience with UCD methodologies are likely to underestimate the diversity of users among their customers. Therefore they do not see the advantages at the first hand, that a systematic process to identify different user groups, select representative users and identify representative user needs can provide [44]. Despite the fact, that the degree of appliance differs, the adoption of UCD methodologies in practice quite mature, especially in specific fields of products that mainly are related to science and technological complexity. In case studies, users were found to be the actual developers of 82% of all considered commercialized scientific instruments as well as 63% of all semiconductor and electronic subassembly manufacturing equipment innovations studied [78] [79]. Since those years, the overall adoption of UCD practices increased steadily and the positive implications of those techniques are widely agreed on. More recent case studies showed, that the degree of the application of UCD in product development still varies widely. Despite this fact, 72% of the participants agreed that UCD methods had made a significant impact on product development by indicating 5 or higher on an 7 point scale. The large majority of 80% stated, that UCD methods improved usefulness and usability of products they developed, where a quarter of them chose 7 on a 7 point scale [83].

It is widely agreed that the involvement of users in the design of a product causes them to know from an early stage what to expect from a product. Furthermore it causes positive mood at the user's side as they feel that their ideas and suggestions have been taken into account during the design and development process [3]. This leads to a positive attitude among the product overall and to positive working experience and intrinsic motivation in special.

### Measurability

As User Centered Design is an abstract set of methodologies and practices to be applied at several phases of a product life-cycle, the benefits of applying those principles are hard to measure. It is widely agreed that there are several positive affects that are caused by accordance to UCD, mainly focusing on user experience, accessibility and product positioning. Nevertheless there are only few implications that can be directly traced back to the implementation of UCD in a quantitative manner. The lack of measurability of UCD effectiveness and common evaluation criteria has be shown be research as well. Research on this topic further revealed, that cost-benefit analysis and tradeoffs are the key consideration in the decision if UCD practices should and will be applied in projects and organizations. Furthermore there is a discrepancy between commonly cited measures and the ones that are actually applied in practice [83].

The widely agreed positive implications of UCD as well as the fact the quantitative measurements are hard to define leads to the combination of quantitative and qualitative as both areas need to be taken into account to estimate the impact of UCD techniques to real world projects. Previous research already clarified the cost benefits of UCD in general, such as reduced need for resources and support cost as wells as increased customer satisfaction and productivity [42]. Although this research was based on case studies and projects in scientific scenarios that range over a broad field of industries and project types. To decide if the employment of UCD techniques is reasonable specific projects in certain fields of industry raises the need for measures that allow for estimation of the impacts of doing so in concrete projects. Furthermore organizations require measures that enable them to determine if application of UCD in previous projects lead to improvements to decide if they continue to implement projects in a user centered manner.
A pragmatic approach for measuring the impacts of UCD in respect to specific projects and organizations was proposed by Kohno et. al. [42]. The described approach takes into account qualitative analysis as well as clearly quantitative techniques of measurement for economical relevant key data. Several of the measures use qualitative methods to gather data which is further used to generate quantitative results. Key measures defined by this approach are changes in sales volume and increased operational efficiency in terms of working hours, as well as reduced cost of project implementation and quality of the created product based on estimations of developers and stakeholders after the project is finished. Qualitative measures like interviews on customer satisfaction, the mood caused by working with the product as well as accessibility are considered as well.
As for today there is still no clear qualitative method for measuring the economic benefits of applying UCD, nevertheless there is effort spent on making the positive effects of UCD clearly visible to establish the practices in industry projects. UCD is clearly a field that can not be measured in a purely quantitative manner as most of the goals of the included techniques aim at increased customer satisfaction and usability, which are not easy to measure by themselves. Therefore it is essential to value to qualitative implications of UCD, although quantitative measures are required to consolidate the position of UCD in industrial environments.

## 2.4 Process Integration

As User Centered Design does not contain an own dedicated process model, the appliance of UCD needs to be embedded in surrounding processes. Even if a organization or team does not want to apply any traditional process model, UCD is not capable of ensuring or guiding a successful project handling. Therefore a basic process model is necessary to implement UCD practices efficiently and draw the most revenue of doing so. If there is no a priori process model to be adhered to, agile methodologies will serve best as several of their key principles are a natural fit to key techniques of UCD. This is because standards state the execution of iterations a an essential element of implementing UCD entirely and iterations are also one of the key principles of agile techniques [36].
As UCD is mainly a collection of techniques and advises how to utilize them, it is possible to embed UCD practices in any process model. For each project and process it can be decided at which stage of the process specific techniques will be implemented and furthermore it is not

necessary to make use of all the practices proposed by UCD as also a partly implementation will lead to an improvement of the final product and it's user experience. Despite those facts, there are some phases that are common amongst most process models, e.g. definition of requirements or final testing, which are focused by UCD approach. If UCD does not influence those stages of the overall process it is unlikely that it is possible to recognize all its advantages.

In general UCD has to be applied over the whole product life-cycle, this means also and especially at a layer above the process model of a products development project. Therefore also standards describing UCD are mainly targeting people that are capable of influencing the process of whole product life-cycle management [36]. To ease the practical implementation and adaption of such processes and life-cycle management, the appendix of ISO 2942-210 defines a guideline for doing so in real world projects.

Summing up it can be said that UCD influences the processes during the design and implementation of products on a project level, as well as the processes used for life-cycle and product management and even the mindset and principles that influence a whole organizations culture. Nevertheless, the grade of application and to which extent existing practices and processes are influenced is up to the developing organization for each individual scenario.

UCD by itself does not prefer specific process models and courses of action over others. It is interesting to observe that UCD and the agile movement developed the principle of iterative approaches widely independent of each other, although some of the proposed techniques arose at the same time. Furthermore t is common to both approaches, that they may be applied at project level as well as to a whole organization. This backs up the leading thoughts of agile techniques as well as of UCD practices.

## 2.5 User Groups

It is obvious the the customer base of products, especially in retail market can get inestimably large. Nevertheless UCD requires the comprehension of users in the design and development process. This raises the need of choosing specific users or groups of those from a large customer base that are representative for the whole customer base. Selecting a representative subset of users from a large customer base is often not easy, as the distinction between those groups is usually neither obvious nor without any doubt. Therefore special techniques for establishing appropriate categorization of users become necessary.

### Definition of User Groups

Involving users in the development of a product usually is possible for just a few individuals which should give a representative sample of the whole customer base. As those have to be selected from a potentially huge customer base, a sampling strategy is needed [44]. Nevertheless the is no general strategy to obtain a representative subset of users, as the selection of those is highly dependent on the type of product, the size of the overall user-base as well as the size of the sample to be selected. A reasonable selection of the representing user groups is essential as not all users can contribute the same value in new product development [29], therefore the ones who can provide the deepest insights and most efficient suggestions are to be selected.

Beside the distinction of those groups, the sample size can be hard to define as well. The size of the user groups to be involved in development and design depends on the product as well as the customer base. Main factors are the overall number of users and the variability among those as well as the number of user roles that should be represented. Nevertheless, research showed, that as few as six users may server as representative sample and can provide useful information to the designers and developers [45]

.

In general there are 3 basic types of users for every kind of product, which are *primary users*, *secondary users* and *tertiary users*. Primary users are those who directly interact with the product and whose workflows and daily routines are influenced by the product. The group of secondary users covers those who use the product occasionally or in an indirect manner. Tertiary users do not use the product themselves, although they are influenced by the impact the product causes. More detailed classification of the three basic types of user groups have been originally formulated by Eason in 1987 [20].

Beside those three basic user groups, there are two constitutional approaches on how to distinct among users and which kind of criteria to use. The first one classifies users by the way they use the product respectively which goals they want to achieve. This method is likely to focus on primary users mainly and split them into sub groups which share special views in respect to the product. The second approach is by splitting users into groups by individual characteristics, that are not apparently related to the product, e.g. body height, gender or age. The first approach is classical for industry projects to ensure product acceptance over the whole target group, which is predefined or already known. The second approach is useful to ensure accessibility and avoid discrimination by the product among all types of possible users. Furthermore the second approach is very suitable for scientific research on how different groups of people utilize specific tools or which usage patterns are possibly related to which individual characteristics oh humans.

The complexity in defining user-groups is significantly driven by the type of product and therefore the structure of the target group. The identification of user groups is especially demanding if there is a large number of users or if the user base is very heterogeneous [44]. A special difficulty arises for new products without an established customer base. For this type of products, the types of customers and representative individuals have to be estimated which is especially difficult as the developers have a market in mind but this assumption is not necessarily identical to the real market that is reached when the product is finished [25]. Nevertheless, this fact allows for designing products selectively to specific market segments. If the segment that should be attracted is well defined, choosing user groups among this segment and applying UCD will lead to better acceptance of the product in the desired section.

An further observation regarding the estimations of developers is, that they usually underestimate the diversity of users. Most developers neglect the importance of various groups beside the primary target group. Therefore they tend to omit groups as infrequent and indirect users (secondary and tertiary) as well as other special groups. This fact can be counteracted by special user identification sessions, which have shown to help developers to orient towards different groups by brainstorming various user characteristics [44].

### Common approaches

Explicit definition of user groups is often avoided as this process is complex and costly, even though the required effort is lower and benefits are increasing in later stages of a project. If there are no explicit user groups, developers usually design the product in a way that seem to be most useful for their idea of a typical user. This means they are explicitly not developing as it would fit their personal needs and bias but for their personal model of an abstract user. Although a developers imagination of a typical user is often vague or even contradictory to the real world users [44]. A compromise that is often taken ion practice is using archetypes as they have been described by Cooper [16]. Using this technique, the elastic notion of user is replaced by archetypes of typical users which are called *personas* and have defined names, needs and biases. When doing so a persona is the abstract representation of a user group summing up their characteristics and goals in one exemplary abstract person. The weighting of factors like the size of various groups and similar metrics are lost in this case. The creation of such personas can be done based on interviews or in cooperation with real users to ensure a close to reality representation of the users by the persona. Although establishing personas makes it necessary for the developers to look into the user base and describe characteristics in detail, they are not adequate to real user-groups and can not be directly compared in later process phases. Furthermore persona development emphasizes building detailed descriptions of typical users, but it does not concentrate on identifying representative user, which is a part of setting up user groups [71]. Important differences hereby are, that there is no possibility for dialogues with personas as well as usability tests on personas are not possible. In industry projects it is often the case, that personas are developed within the development team and maybe few representatives of an applicant just to establish a common understanding of a user among the development team. Although this technique also relies on personas, it can not be denoted a User Centric Design approach, as the personas rely on assumptions which do not necessarily hold true for real users. Nevertheless personas can be a good approximation of user groups if they are set up correctly and with influence of real users and they can server the needs of projects where establishing real user groups is to costly or otherwise impossible.

In literature on Human-Computer Interaction there is usually the advice to categorize users by means of certain common characteristics. An exemplary list of such characteristics was set up by Scheiderman [69] where he states that each design should be based on understanding of the intended users. Characteristics mentioned in this list are including population profiles reflecting age, physical abilities, education, gender, as well as cultural background. Furthermore there are training, motivation, goals and personalities as well as characteristics influenced by user communities, different countries, rural and urban location, economic profiles and disabilities. Although this list is quite long, there are more detailed lists of user characteristics by various authors containing lots of other measures [44]. It is not very likely that user groups are defined according to all of those characteristics, although those lists serve as an origin to develop a list of categories for a specific product and situation.

One should be aware that user groups can not be defined at the very beginning of a development project that hold true over the whole product life-cycle. Case-studies showed that establishing adequate user groups is an iterative process [44]. This means that at an early point in time user groups are established according to categories that seem reasonable at the current situation.

While working with those user groups in terms of discussion, usability tests and other techniques it will be revealed that several individuals agree on their opinions and therefore are likely to represent a natural user group. On the other hand it is likely to happen that single individuals do not agree with the majority of their group repeatedly. In such situations it is valid to assign an individual user to a different user group at any time during the product life-cycle. If the situation occurs that there are commonalities among users, even of different groups, that have not been treated as a significant characteristic yet it is reasonable to introduce a new user group summing up those individuals and categorizing them by the newly discovered commonality even if this causes the annulment of a previous user group. Another reason to adapt user groups and move single users to another group can be individual changes affecting this persons view to the product which are outside the scope of the project, e.g. getting a new job.

### Lead User approach

When identifying user groups, one characteristic used may be the user's competency in using a product. This can result in one experienced or expert group. Focusing on this group is referred to as lead user approach developed by von Hippel [44] [80]. The biggest difference to the approach of user groups is, that the lead user approach is not defining multiple user groups but focusing on the one group of users that is likely to have the most valuable impact on product design and development. For identifying lead users there exist several methodologies where all rely on the same principle, which is to measure a users competence in terms of using a product as well as knowing the market which the product is placed in [29]. The relevance of lead user techniques has been analyzed in case studies comparing separate samples of lead and non-users. As a result the lead user concept was judged to be superior alternatives available at that time [78].

The lead user approach assumes that there are users, so called lead users, who can have a highly positive impact to the development process and should therefore be involved into this. This assumption is based on the idea that users who have experience with a need are better able to give information regarding it than others. Lead users usually are confronted with with requirements and needs that become present on the rest of the market months or even years later. Therefore, if the market segment is one where need-related trends exist, those people experiencing needs before others do are leading those trends [78]. A further attribute of lead users is that they profit significantly if a solution for those needs is found as fast as possible [29] [81]. Due to this lead users are motivated to put effort to understand the problem and find a solution for it, which is not a common attribute for ordinary users. Often it is the case that lead users identified need years before the market and already found personal solutions to overcome those problems, where there personal solution are a valuable basis for product improvement [44]. One field where the phenomenons assumed by the lead user approach appear especially often are most segments of the high technology market. As this market is changing very rapidly is it hard for supplier to stay on track of current market trends as well as it is hard for ordinary users to be at the front of the trend. But only those users, that manage to stay at the front of the trend or even participate in the definition of this trend have the real-world experience which manufacturers must analyze to know what the market will demand tomorrow [78].

Working with lead users and integrating them in the process of product development is attractive for developers as well as manufacturing organizations due to several reasons. The main reason is, that lead users are very familiar with the product and they can forecast market trends and they are willing to utilize their knowledge for product improvement. They are able to provide highly creative ideas, suggestions for improvement and innovations that can incorporate the product development process. Lead users are capable of triggering product innovations and they are willing to participate in this innovation process for the benefit that their needs get solved, even though they are not yet present in the market. An important attribute of lead users that are attractive for developers is the ability to communicate clear and effectively with organizations and development teams. In this communication lead users usually are proactive as they want to share their suggestions for improvements and influence the future design of a product [29].
The benefits of lead users become clear by the results of several case studies, where one of the identified that one lead user contributed as much information and insights to the development process as five ordinary users. One problem that has been observed in case studies is, that the requirements of lead users are so advanced that they are not relevant for ordinary users at all [44]. For that reason in many cases it is the best choice to combine lead users and ordinary users, where lead user support the team with their needs and innovative ideas which are checked for realistic needs by ordinary users. In the case of discrepancies it has to be decided by the development team if certain suggestions are implemented or not, as the implementation may cause further costs but on the other hand lead users may get demotivated and disappointed if their participation is not honored.

A classical methodology for applying lead user approach to a product development project can be described by the following steps [78]:

**Specify Lead User Indicators** To specify those indicators they should be divided in *market or technological trend and related measures*, which contain the users competency regarding the product and the market. Users should be experienced in using this or a similar product as well as they should have insights to possible alternative solutions. The second group of indicators are *measures of potential benefits*, as lead users are motivated to participate in product design as it can help to ease their everyday work. Therefore the potential benefits of the users should be analyzed, as a user that is front of the trend and knows the market but has not to expect any improvements for his situation is unlikely to be motivated to serve as a lead user.

**Identify Lead User Group** Based on those indicators a group of lead user can be selected from all the known users of the product. The size of the group to be selected is depending on the size of the development team as well as the complexity of the product. As lead users tend to be very proactive and innovative the group of lead users should not become too large to offer the possibility for every suggestion to be heard and evaluated. If there is a clear group of users that already have an established relationship to the developing organization and are known to be in front of the trend, this group of users may be used as lead user group directly without defining any indicators.

**Generate Concept with Lead Users** When the group of lead users is selected, a concept of the product should be developed by a cooperation of the lead users and the development team. Techniques to do so are available in sufficient quantities in the field of UCD and can be applied here directly.

**Test Lead User Concept** Once a prototype of the product that has been design with the influence of lead users is implemented it should be tested with ordinary users as well as lead users to avoid building a product for a specialized target group. Lead user groups are likely to design highly innovative products but often they fail today's market and the needs of ordinary users.

The lead user approach has significantly influenced the way product innovations are achieved today. The classical model was based on *producer models*m which means that suppliers and organization identify potential innovations and products for their customers. This has been reasonable on the first sight, as suppliers and manufacturers have the capabilities to implement innovation processes and develop new products, which are afterward used by a large customer base. Although there was no place for the innovation potential of product users which means that the potential of lead user lied idle [3]. The lead user approach allows to utilize this potential for innovation while still using the manufacturers capabilities and therefore can lead to innovative and new products that fit customers needs. Empirical studies show that up to nearly 40% of users engage in developing products in specific types of industrial products as well as consumer products [4] [21]. In terms of creation and design of innovative new solutions and whole products, the lead user method turned out to be almost twice as fast as traditional approaches to reveal promising product concepts while it is less costly as well [29] [31].

## 2.6 Discussion

User Centered Design is a quite general and philosophical approach on how products should be designed and where the input for a product development process should originate from. By putting the later customers into focus by letting them provide the required data for the whole development process and asking them for participation in the design phase, UCD leads to a removal of technology of the process of product design. This suggests the analogy to drivers for product design in economics, which are *technology-push* vs. *market-pull*, by treating classical, technology dominated development approaches as technology-push approaches while the methods of UCD cause a shift towards customer oriented market-pull approaches by developing exactly the kind of product the customer requests. The basic principle in UCD is to focus on the purpose of a product from a customers perspective, which is to fulfill a function that the customer needs. To reach this goal it is necessary to involve the user in the design process instead of building theoretical models of users and deriving abstract product concepts from those theoretical user models.

Product development through collaboration of designers, developers and users is a permanent learning process for all the involved parties. Therefore an iterative approach becomes reasonable and even necessary to allow each participant to identify the viewpoint of other parties and to understand their arguments and motivations. Doing so enables the team to lift the whole discussion to a higher level in the following iteration. One main advantage of UCD is that it allows a deeper understanding of the psychological, organizational, social and ergonomic factors that affect the use of the designed product [3].

The benefits of UCD for the developing organization, customers and designers are obvious and also economical benefits of UCD approaches can be founded. Although, from a management perspective and classical product development scenarios a more technology driven and well structured and defined process would be preferable. UCD hardly states defined sub-goals to be reached and its only requirement to the design process is to be iterative at least in parts. UCD by itself helps to reveal the users needs and give the user the possibility of participation, nevertheless UCD is not capable of ensuring the derivation of a structured and sufficiently technical representation of a product from this bunch of input, to allow for an efficient development and implementation process.

CHAPTER 3

# Requirements Engineering

A crucial phase at the beginning of any kind of project that aims to create a product for a customer is the phase of defining the requirements of those customers. Whenever the developer of a product is not its later user, the developer is challenged to uncover, understand and specify the user requirements (Davis et al.,1997) [43]. In this case, mental models represent how the user thinks about the product he wants to be implemented as well as a mental model of developers. Before requirements have been clarified, these two model can be radically different although related to the same system [43].
Due to this importance Requirements Engineering became an important field of research especially in the field of software engineering and several techniques for requirements elicitation as well as definition have been established [6][7] [22].

## 3.1 Definition

Requirements Engineering has always been an intrinsic part of any project, which gained more and more importance over the last decades. Due to the perception of the importance of clearly stated requirements, Requirements Engineering became a dedicated phase of development projects that includes learning, uncovering, surfacing and discovering the needs of the intended systems stakeholders [1–3] [28]. The result of this phase is a list of requirements that define the product in a way that satisfies each stakeholder. A single requirement can be described as a condition or capability that is needed by a stakeholder to solve a problem or achieve an objective [5]. A more formal and sound definition is stated in *IEEE Std 1233-1998* and defines requirements as

> „a condition or capability that must be met or possessed by a system or system-component to satisfy a contract, standard, specification, or other formally imposed document“. [35]

Nevertheless, this definition states what has to be included semantically in the result of requirements analysis, though it does not state how to achieve and represent those requirements.

Actually the representation as well as the acquisition of requirements depends on thee individual stakeholders as well as the project situation at hand. Generally there are three basic questions that have to be answered at the beginning of requirements analysis, which help to adequately define the detected requirements. According to [88] the three basic questions for requirements analysis are

**What should requirements be?** The development team as well as stakeholders need to agree what kind of requirements they want to be defined and at which level of detail they want to elicit them.

**How should requirements be stated?** There are several methods to describe requirements, reaching from formal tables over technical descriptions and structured language to user stories which are less technical and written in natural language.

**How should requirements be derived?** Deriving requirements is the key challenge of the phase of requirements analysis. As there are various techniques to solve this quest, the team and stakeholders need to agree on which methods they will utilize to do so.

As requirements are a clear and sound definition of what a user wants or needs the product to be like in order to help him to achieve a goal, they just state what the product should be like. It has to be stated, that requirements never specify the way of implementation or technical descriptions of how to fulfill specific requirements [1] [2]. Therefore they can be seen as a kind of wish-list describing the product as it should be without considering how this requirements can be met. Nevertheless the list should contain elements that are likely to be satisfiable.

### Functional vs. non-functional requirements

Requirements can be divided in two main categories, which are *functional* and *non-functional* requirements. The requirements that are usually defined in projects in industry practice are mainly functional requirements, which describe specific features of the products and their behavior. Functional requirements describe how the system works and usually state some measure to determine the degree of fulfillment in a quantitative way. Non-functional requirements on the other hand define 'soft-facts' about the product like the look and feel, noise levels, UI-Design as well as usability and overall customer satisfaction. Non-functional requirements may also include strategic organizational requirements affecting the product. For each project functional as well as non-functional requirements are necessary to describe the system sufficiently to ensure an implementation that satisfies all customers expectations. Research identified projects where a shift from customer- to market-driven approaches caused a changed balancing of technical user-requirements and organizational non-functional requirements . [66].

Non-functional requirements are mainly influencing a products quality, as there usually exist several ways of implementation that fulfill all the functional requirements, their accordance to defined non-functional quality attributes express which solution will fit the customers needs best. Quality attributes can be described as non-functional requirements of a system, e.g. reliability, performance and modifiability [7, 10]. [11]. One further type of non-functional requirements are usability requirements which have string impact on functional requirements since the usability goals define how the system will be utilized by the users [89].

Although non-functional requirements are meant to define a products quality beside its feature set which is described by functional requirements, research showed that in some situations the estimations of non-functional requirements is not perceived useful enough to ensure the high quality of a product [11]. Nevertheless a combination of both requirement types is required to state a sound definition of the product to be designed and in most cases non-functional requirements are an adequate method to ensure the quality of implementation for later customer satisfaction.

### State of the art

Due to the increased importance and appreciation of requirements engineering, research developed several techniques and methodologies for requirements analysis. One of the most intuitive techniques are interviews of stakeholders. The most simple form of requirements interviews is a meeting of all projects stakeholders, including the development team, where a moderator tries to identify requirements in a top down approach which are written down in natural language. During this meeting the product to be designed is described in an abstract way in the beginning to identify several fields or groups of requirements. Each group is afterward discussed and described in more detail. As each requirements needs to be discussed until there is a consensus among all stakeholders, this interviews and meetings can last for several days.
A more efficient approach which is also applied in practice are collaborative workshops which are described in [53]. Those workshops are able to deliver requirements fast and furthermore they build a healthy project community which lasts beyond the phase of requirements elicitation [24].

Although some basic techniques of requirements engineering are applied in practice, most of them are used intuitively and are not following a predefined process model. This makes it hard to ensure the correctness of the identified requirements as well as to maintain them over the whole project life-cycle and react to changes. Despite several benefits that are clearly given by according to a requirements engineering process and utilizing specific methods and techniques to identify and define requirements, the current practices in small and medium enterprises are mostly ad hoc ones, which has been shown in case studies [53].

### Usability Requirements

One specific dimension of non-functional requirements are *usability requirements* as mentioned in ISO/IEC 9126 [37]. Usability is an important topic to be covered already in the phase of requirements analysis, although this kind of requirements can hardly be captured in terms of functional requirements. Usability requirements encapsulate essential interaction steps, needed information and options for the user. Thus they specify the system behavior in a more abstract way without being too specific about concrete details of implementation [89].
Usability requirements are supporting developers in generating a conceptual model of the system

to be implemented by understanding the workflows and ways how users want to achieve specific goals. Therefore usability requirements can help to keep the gap between the designers model and the users model small. Usability and workflow requirements are synthesized into a set of solution elements which are further detailed using functional requirement by the designers [89], and therefore lead to a overall product structure in components that is reasonable for a products users.

Defining useful usability requirements and allowing designers to utilize them to design a system should be a natural part of the requirements engineering process, although it requires usability awareness in the development team. In practice many developers tend to underestimate the importance of this kind of requirements and are focusing on functional requirements. Usability requirements are ofter more complicated to fulfill as they require and integrated view to the system and can not be implemented punctual or in terms of components as this is often the case for functional requirements. Due to this fact, usability teams started to organize training in their companies to raise awareness for usability requirements and find ways how to handle the for development teams [30].

## 3.2 Motivation

Requirements Engineering gained wide acceptance over the last years and is generally accepted to be one of the most critical and complex task during the development of systems. Furthermore Requirements Engineering is especially important to the development of socio-technical systems, which nearly every of today's product are counted among as they interact with end users in some way [38], [62] . It is widely accepted that discovering errors during the phase of requirements analysis can reduce the cost of correcting the error by a factor of 100 compared to mistakes that are discovered during implementation or acceptance tests.

Due to this facts, Requirements Engineering is essential for a projects success, which was made evident by a Standish Group study in 1995, which showed that most of the causes of project failure are related to the RE process [65].

Beside this importance for project success and cost less correction of errors that are discovered early, requirements engineering offers several other benefits to project teams. One of them is the possibility of trading nonessential functionality for product quality [66]. In practice it is often the case, that one of the three dimension of a project, *time*, *budget* and *quality*, can not be met. By not exceeding the projects deadline as well as the budget the only way to go is to increase the product quality in this traditional model. Adequate Requirements Engineering, especially *requirements management* can help to cut quality in terms of a products scope and not the quality of implemented features. This can be done by skipping features that have been identified as less important to users during requirements analysis. By doing so the overall quality of the product can be kept high, except the missing features, while finishing the project in time and budget as well. Trading features against quality is not possible if there is no clear definition of required features and measures of the quality of their implementation. Beside this importance for project success and cost less correction of errors that are discovered early, requirements engineering offers several other benefits to project teams. One of them is the possibility of trading nonessential functionality for product quality [66]. In practice it is often the case, that one of

the three dimension of a project, *time*, *budget* and *quality*, can not be met. By not exceeding the projects deadline as well as the budget the only way to go is to increase the product quality in this traditional model. Adequate Requirements Engineering, especially *requirements management* can help to cut quality in terms of a products scope and not the quality of implemented features. This can be done by skipping features that have been identified as less important to users during requirements analysis. By doing so the overall quality of the product can be kept high, except the missing features, while finishing the project in time and budget as well. Trading features against quality is not possible if there is no clear definition of required features and measures of the quality of their implementation.

Development and implementation of high quality products depends on the successful collaboration of heterogeneous stakeholders. Furthermore this collaboration needs to last over all the different life-cycle activities [27]. To allow for a efficient collaboration over this long time among all stakeholders, some defined process has be applied, which is referred to as requirements management. Nevertheless any kind of collaboration in respect to product quality as well as requirements management is based on adequate requirements analysis at the beginning of the project. Without clearly stated requirements, that can be discussed and modified during the whole life-cycle, collaboration and adaption to changes in changed requirements of a stakeholder become impossible.

### Potential Risks

Although applying techniques of requirements engineering helps projects to be successful, they raise some risks. It has to be mentioned, that just a few of those risks are really originating from the applied requirement engineering process but they just become visible much better due to this. Nevertheless, some of the risks are actually caused by requirements engineering and won't occur otherwise.

Three basic types of risks regarding product requirements are distincted in literature, which are *ambiguity*, *inaccuracy* and *inconsistency* [86]. All those are meaning attributes to the list of identified requirements which, if they are present, mean a risk due to an invalid or low quality requirements definition. Ambiguity refers to requirements that allow for different interpretations by customers and developers, therefore each requirement should be described clearly and the interpretation of all stakeholders should be consolidated. Inaccuracy refers to functional requirements which define some measure to check their fulfillment. This measure may either be inadequate by itself or the required value does not cope with errors of measurement. This leads to problematic situations during the acceptance tests at the end of the project. The last attribute of bad requirements is inconsistency which applies to a combination of multiple identified requirements. Between those there may be inconsistencies or even contradictions. If so, those inconsistencies are likely to be discovered during product development, although correcting requirements at this late stage can be costly. Therefore all requirements should be checked for consistency before development starts. Overall it can be said, that all three types of risks may also be present if no RE process is applied, the used techniques just make the risks visible, therefore it allows to deal with them at an early stage as well.

Dealing with discovered risk at such an early point can be problematic for industry projects in special cases, as this means revealing all the potential errors to be made and the whole projects complexity to the customers. This may lead to an early shut down of the project before development even started. This risk is caused by applying RE techniques, but nevertheless the project would have been likely to fail otherwise as well. It is widely accepted that the earlier such potential risks are identified the easier it is to manage or even eliminate them [8].

One big risk of applying RE is the completeness of the identified requirements. Once the requirements list has been established, it is very hard to ensure that it is sound and contains all the necessary requirements to ensure a correct implementation. Although having such a list in place will make developers and other stakeholder tend to not rethink the requirements during the implementation phase and just trusting the list of requirements as it was ensured that is is complete. Dealing with this problem is part of requirements management and the whole team should be aware of this risk when implementation phase follows a list of identified requirements. Davis [19] suggests three reasons which make it unlikely to obtain a complete list of [88]:

**The limitations of humans as information processor and problem solvers** This problem is by nature and can not be avoided. In large and complex projects humans are often not capable to keep the overview of the whole project and requirements and discover differences between those.

**The variety and complexity of information requirements** There are various different types of requirements which are specific for each project. Due to this a complete field of requirements may be oversight during the requirements analysis phase. Due to the singularity of each project there is no general process of checklist to avoid this problem.

**The complex patterns of interaction among users and analysts in determining requirements** Requirements analysis and elicitation is a complex interaction process between various stakeholders coming from different domains. This fact leads to a highly dynamic interaction process during requirements elicitation which is hard to cover by defined processes to ensure completeness of the result.

Another problem regarding completeness of requirements is that often requirements are identified early but not formulated and tracked as they will become important at a later stage. In contrary to this approach, each requirement should be traced via requirements management from the first time it has been discovered. All requirements during a software's life-cycle have to be seen, independent of whether they are important and necessary at the present time or they are not but will become important in future [2].

Common requirements engineering activities in practice tend to overvalue the technology side of requirements meaning focusing too much on functional requirements and possible ways of their implementation [5]. This is due to the fact that those kinds of requirements are easier to measure and they do not need to cope with *requirements uncertainty*. Several strategies to deal with requirements uncertainty to face this problem used by experienced IS/software manager have been analyzed in recent research [56], which made clear that dealing with uncertainty of requirements is an intrinsic part of adequate requirements management by reducing the uncertainty over the projects lifetime.

Uncertainty is often caused by different background and vocabulary used by various stakeholders. As requirements are expected to be contractually satisfied, specifications are often written in natural language [86] which makes it hard to identify and eliminate uncertainties. This problem originates from a communication barrier between analysts and stakeholders caused by a gap in their domain knowledge [10]. Therefore a common basis of domain knowledge of all stakeholders as well as the analysts is the fundamental of successful requirements analysis to deal with specific vocabulary and uncertainties in formulations and specifications.

## 3.3 Requirements Representation

During the process of requirements analysis it becomes necessary to persist the identified requirements in a way that allows understanding at a later point in time, even for persons who did not participate in the process of requirements analysis. Furthermore the representation of the requirements must be adequate to be used by developers to decide about implementation details, for testers to decide about the boundaries and conditions of test cases as well as other stakeholders like customers and project management as this list of requirements is the central piece of information to be discussed over the remaining project lifetime. Due to this multiple utilization of the requirements persistence it is crucial to find a way of representation that fulfills all those needs which is commonly referred to as *Requirements Specification*. A general definition of a Requirements Specification (RS) is that it

> „provides verifiable and testable demands towards the system." [89]

Beside those general definition of the specification of requirements, there is an industry standard providing guidelines for defining usability requirements. As usability requirements are one dimension of non-functional requirements which gained importance over the last years, they follow special rules to be stated in sufficient detail to make an effective contribution to design and development as well as defining criteria of usability that can be validated if needed [6].

A sound representation of requirements in general needs to contain several pieces of information for users, designers, implementers and testers of a system. Although there may be further stakeholders relying on the representation of the requirements, at least those four groups should always be served by this kind of document. To do so, this representation must at least contain the following information [88]:

**Functional specification** The functional specification is basically a list of functions that the system must be able to perform. This can be implemented in terms of a list of functional requirements.

**System context, constraints, and assumptions** A clear definition of the scope and the systems boundaries must be given. This includes environmental conditions of operation as well as boundaries in terms of use-case scenarios where the system is not adequate to fit.

**Performance specification** Dynamic properties of the system must be defined in terms of required value ranges or similar. Basically this part includes most non-functional requirements.

**Measurement and test conditions** A testing process stating how the previously requirements are measured and validated in necessary to ensure a common understanding of test scenarios as well as the general measurability of the defined requirements.

One proposed structure of a requirements specification satisfying these constraints is to state *a functional model of the object system*, *a data dictionary defining the various components of the functional model* as well as *a set of performance and test specifications for the system* [88].

### Requirements Specification

The term *Requirements Specification* refers to the outcome of the requirements analysis process and is defined by various standards, so the IEEE STD 830-1998 defines this document generally as

> „a document that correctly defines all of the software requirements of the system to be developed.“ [14]

A similar definition of a Software Requirements Specification (SRS) is given as

> „short statement of the requirements to be fulfilled by the software.“ [15]

[33]

As these definitions are correct but very abstract and not defining what should especially contained in a (S)RS, a more detailed definition is given in a IEE standard that states that a completed SRS should include all customer requirements, as well as those needed for the definition of the system. Moreover, it should have all pages, tables and figures numbered, all terms defined, all units provided and all referenced material present. Finally, it should not have any to be determined (TBD) sections [35].

As each definition of a sound RS contains a list of requirements, there are several attributes that need to be fulfilled by the requirements to ensure the feasibility of the SRS. A main demand is the understandability of each requirement as well as the complete RS. This refers to the usage of unique words, symbols and notations and the use of grammatically correct language and symbology [5]. Furthermore feasibility is considered as a central requirement attribute as well to result in a RS that provides value during the phase of implementation and system testing [5].

A complete list of quality attributes for Requirement specifications has been stated by Wilson [86] and contains the following attributes:

**Complete** A complete Requirements Specification must define all real world situations that might occur as well as the systems behavior in those situations.

**Consistent** Consistency is given when there is no combination of requirements that contradict each other. This means all requirements are compatible to each other as well as non-functional requirements to not negate the utility of the product.

**Correct** To be correct a Requirements Specification must identify the conditions and limitations of all real world situations that might occur as well as the system's behavior in those situations and it's impact to the situation.

**Modifiable** In order to allow for later requirements management, the Requirements Specification must be modifiable and ensure that all changes are traced.

**Ranked** The requirements in the Requirements Specification need to be ranked to express the importance of specific requirements in respect to others.

**Traceable** For traceability each requirement need to be assigned an individual identifier applying to a logical scheme.

**Unambiguous** Statements of requirements need to allow only one possible interpretation. This often makes necessary to use domain specific vocabulary which should be clearly defined in a glossary.

**Verifiable** To ensure verifiability specifications at one level of abstraction must be consistent with those at other levels.

Conclusively there are several attributes to be fulfilled by a Requirements Specification, which makes the whole document complex and costly to implement in a correct manner. Nevertheless this effort pays off during the remaining project life-cycle as a qualitative SRS can server as the central document used by various stakeholders and eases communication, change management and elicitation as well as approval of the final product.

## 3.4 Team constellation

The process of requirements analysis involves multiple stakeholders represented by one or more persons. Due to this fact various persons and individuals are involved in this process which is very communication intensive [17]. Therefore human interaction plays a big role in the process of requirements analysis and management [11] [28], which makes it necessary to set up an adequate team for those activities. Requirements engineering has been identified as a mainly social interaction between people [5] therefore also research findings from the field of psychology have been applied to this process. One of those conducted the potential of cooperative inquiry which assumes that genuine cooperation means sharing both knowledge and power [53] which backed up the importance of team constellation for the process of requirements engineering.

Beside the various stakeholders that are represented in a RE team, their role in their organization and the project needs to be taken into account. Independent from the represented stakeholders only a part of the team should be made of decision makers that are responsible for project level decisions. Domain specialists as well technical professionals and usability experts should be present in the team as well to ensure an integrated view to the project. Each of those individual is driven by different motivations and biased perspectives [5], which may be of economical nature or regarding the technical feasibility or quality of the resulting product, which are equally important and need to be harmonized and negotiated during this process.

In practice usability specialists as well as people that are mainly targeting the quality of the final product are often not part of the RE team or are not in the position to influence decisions effectively. However usability specialists have been showed to be willing to conduct user stories and to increase the understanding of users and the domain for all team members [30].

An additional fact to be considered when establishing a RE team is the availability of the team members. In practice it often occurs that especially managers and decision makers as well as customer representatives are participating in initial requirements analysis and contributing to an early stage requirements specification. Although this increases the quality of the requirements, the RS will change over the projects lifetime, which causes problems if not all members of the team are available at later stages. It can cause later misunderstandings or problematic changes to the RS if the stakeholders of specific requirements are not available for further explanations or negotiations of requirements. Case studies showed that it often is the case that the customer representative is not always available and involved in the development team [7]. This problem can be reduced by establishing an internal customer proxy inside the developing organization that delicately represents the customers point of view.

### Domain Knowledge

A special consideration when setting up a team for requirements engineering is the previous Domain Knowledge (DK) of each individual. Although DK has several positive effects on the first hand it can also cause several problems when analyzing and negotiating requirements.

Domain knowledge among customers is present in most cases and is also hard to avoid. The extensive DK of customer representatives in practice often causes the developing organization to choose representatives with DK as well to appear competent to the customer and to ease communication, as specific vocabulary is understood by both parties. Research has shown that sufficient DK on the requirements analysts side support the communication between the analyst and the stakeholder. Being able to use the domain terminology furthermore enables the analyst to present questions that the stakeholder can understand and to comprehend the answers that are given [28].

Beside those advantages of domain aware analysts, this knowledge causes several problems as studies in the field of psychology [85] show that it may cause a tendency to approach situations in a way that has already been used in the past. Therefore DK can lead to fixation in problem solving and hinder innovation and new concepts. Furthermore some indication of negative effects to the cognitive processes involved in requirements definition have been observed in case studies [28].

Although analysts with deep DK are helpful for communicating with all stakeholders during RE, studies argue that a person without any domain knowledge at all is important to the success of system development. This ignoramus has not assumption about the domain, although he or she has technical understanding and motivation to participate in the RE process. The ignoramus is capable of spotting inconsistencies and will ask questions whenever some information is missing or something has been left unsaid [28]. This minimizes the risk of DK aware analysts to neglect to ask questions to which the answers seem obvious, although this is not true.

## 3.5 Process Models

As Requirements engineering is only one aspect of implementing a successful product or finishing a development project it must be integrated in the overall process model. Requirements Engineering in general is applicable to any kind of process model as it does not state any process by itself. Nevertheless it is important that Requirements Engineering includes analysis as well as management of requirements, therefore it can not be applied at the beginning of a process model and not paid attention at later stage. The requirement planning phase and management phase are executed independently for effective management of requirements and especially management of requirements is iterative in nature [61]. Although this does not hinder to apply it to traditional, non-iterative process models. Requirements analysis in the other hand as well as development of specifications should occur prior to writing the requirements document and therefore at an very early phase of the overall process model, which applies well to sequential models but can also be done in iterative environments.

Results from studies on the appliance of RE ion practice revealed that the process model used differ from those described in literature. Several case studies showed, that RE does not occur in a structured way but are implemented opportunistically when they seem adequate and therefore are simplified as well as the requirements models are restructured when they reach high complexity [51]. Due to this fact, the applied processes vary widely among organizations and projects and are seen as one aspect of the overall development process instead of a discipline in its own right. This view to RE also affects the team structures which means that there are usually no dedicated teams established responsible for requirements analysis and management but each project stakeholder influences this processes at some point in time without central organization. This is contradictorily to literature which suggests to establish specification and style standards at the outset of the project and further to train all project participants in the use of those standards [86]. The structured and disciplined process for RE as developed by research, which aims at introducing engineering principles into the practice of system analysis and establishing structured and repeatable activities, is not widely accepted in practice [51].

The Capability Maturity Model [64] explicitly addresses requirements processes on level two Requirements Management key practice area but not in the following levels. More recent definitions like the Systems Engineering CMM4 [77] has a wider coverage of RE activities but is perceived only at niche industries and therefore not widely applied. Literature defines a similar multi-level maturity definition in the dedicated to the field of Requirements Engineering by distincting the following levels [66]:

**Level 1** Organizations at the initial-level utilize ad-hoc requirement processes. They can hardly estimate and control costs due to the necessary rework of requirements and poor customer satisfaction. Their RE processes are not supported by structured documentation or reviews but are dependent on the skills and experience of individuals.

**Level 2** Organizations at the repeatable-level have standards for requirements documents and policies for requirements management in place. They utilize tools and methods to ensure high quality of their documents and their completion in time.

**Level 3** Organization on the defined-level have a defined process model based on good practices. They have an active process improvement and are able to assess new methods objectively.

Today most industry organizations are at the initial-level as their processes are mainly ad-hoc and no standards for documents are defined at an organization wide level.

### Integration to Life-cycle

The concept of software life-cycles has been proposed by IBM [12] similarly to product life-cycles. According to those terminologies, a products life-cycle starts with establishing the development project and lasts over operation and maintenance until a controlled removal from service. Traditionally RE was placed at the very beginning of this life-cycle when the development of the product starts and RE was finishing at latest when the implementation was complete. With raising complexity of products it became impossible to develop an accurate set of requirements that remain stable over the whole life-cycle [61]. Due to this trend the techniques for requirements management have been developed which are place over the complete life-cycle of a product.

In practice this means that requirements analysis is still performed at the very beginning of a project, but the identified requirements are not taken for granted and the development team must be aware of changing requirements and therefore the process model has to cope with those changes. Nowadays requirements engineering activities occur across multiple phases in most process models [51], where the maintenance of the requirements and reacting to changes is handled by implementing specific requirements management techniques.

This evolvement lead to RE as a discipline that affects a whole products life-cycle including handling requirements for a clean shut down of a product, therefore there are various RE techniques that are meant to be carried out in parallel to product development as well as maintenance and redesign.

### Phases of Requirements Engineering

Requirements Engineering contains two main disciplines which are *requirements development* and *requirements management*. In traditional process models requirements development is implemented at the beginning of the project. It is also referred to as *requirements elicitation* or *requirements gathering* and covers activities to discover, analysis and specify the relevant requirements [61]. Requirements management is implemented in later project stages and traces changing requirements and introduces those changes to the applied development process. Nevertheless both activities should be carried out in parallel to the whole project life cycle.

Requirements development contains 4 main activities which are *requirements extraction*, *requirements analysis*, *requirements documentation* and *requirements validation*. All of those activities can be implemented in parallel over the whole project duration [2]. Requirements extraction covers activities to identify the requirements, which are afterward analyzed in detail when discovered. Requirements analysis should provide detailed insights on the details of requirements and contains discussion processes to ensure a clear understanding of requirements by all stakeholders. Another activity of requirements analysis is negotiating agreements on the requirements [53]. Requirements documentation is the process of generating a requirements specification or a similar requirements document collecting all the identified requirements. The last activity of requirements validation aims at ensuring that the requirements specification is valid and understood by all stakeholders in the same way. Once the first raw requirements are extracted all the other activities can be started on those requirements while requirements extraction is still continued providing continuous input to the following phases.

Requirements management is based on the requirements specification which was documented in previous phases. Activities of requirements management ensure a structured process and implementation of the requirements development activities during the projects lifetime. Furthermore it contains activities to identify changing requirements and making sure that all stakeholders are aware of those changes and the whole requirements specification stays valid.

### Iterative vs. Sequential Approach

Traditional process models pursuit sequential approaches while, in contrary to that, nowadays processes which are iterative in their totality are established. A third group of process models which occur in real world situations processes with iterations between activities have been identified [51]. Requirements Engineering is implemented in all of those process models, however RE is widely iterative by nature as their are several research results that back this up.

Stakeholders are continuously adding new requirements or modify existing ones until they reach a specification of the system in a way that fits their conception [49]. When the stakeholders see other requirements from different individuals, they discover new requirements which are important to them or the become aware that the existing requirements do not fit or even contradict their needs. This can happen at any time during the process, therefore one of the biggest risks for development projects is to be unaware of this fact and treat requirements that are specified at the very beginning as fixed and valid [53]. If this position is taken at an organizational level the implementation of an adequate requirements management process becomes impossible.

Also when examining a whole product life-cycle iterations in RE are necessary as software evolution is a process which is based on continuous feedback from various sources [22]. Therefore an adequate RE process has to be iterative over the whole life-cycle of the product to be able to cope with this continuous feedback. Such a process must not stop gathering new feedback at any point in time just because the sequential flow requires to do so.

Traditional cascading models, e.g. the waterfall model, aim at recognizing and specifying requirements of stakeholders before the development starts and therefore to avoid rework which would lead to wasting time and costs. A basic flow of an sequential RE process is depicted in 3.1. This approach is based on two assumptions. At first it is assumed that the correction or change of requirements in any case leads to high costs. The second assumption is that is is possible to define complete and correct requirements at the beginning of a project [2]. Although research showed that at least the second assumption is not true and when looking at agile project teams the first assumption does not hold true in any case as well, this kind of processes is still widely applied in practice.

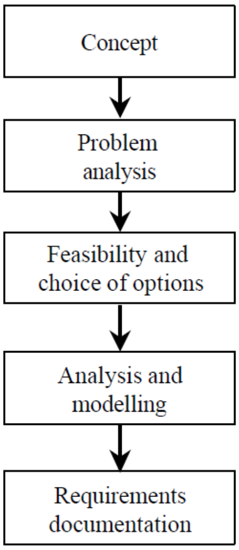

**Figure 3.1:** Linear requirements engineering process [51]

Requirements Engineering practices in agile software development, e.g. Scrum [67], widely differs from traditional RE processes [7]. In a process that is generally iterative and strongly involving the user, RE integrates smoothly and becomes an essential part of the overall process instead of being implemented in parallel to the technical development. A schematic representation of a purely iterative process is given in 3.2. This leads to the fact that RE is often less formal in iterative environments and to some extent not denoted as Requirements Engineering explicitly, but is seen as an intrinsic part of the overall development process.

A conceptual linear RE process with iterations between activities has been proposed by Sommerville in 1998 and can be seen as a compromise between both approaches. This partly iterative process states that activities are often overlapping and are implemented iteratively as depicted in 3.3. The linear progression resulting in documentation is reflected by this process as it is by the traditional and iterative approach as well.

Conclusively Requirements Engineering activities can be implemented in each of the three described process models, although in sequential models it is often the case that RE is implemented in parallel to the technical process and contains iterative elements when they are necessary. As an adaption to this the partly iterative models have been developed which allow Requirements Engineering to implement the required iterations while they enable a sequential flow of the general process. In purely iterative environments RE integrates in all elements of the process once it is established but demands some sequentiality at the beginning.
Case studies showed that one model cannot represent all processes of requirements engineering in practice. A purely sequential model was not successful and iterations have been introduced ad hoc when necessary. Conversely a pure iterative process was not able to represent the progression of RE activities which often build on preceding activities. Resulting from those observations

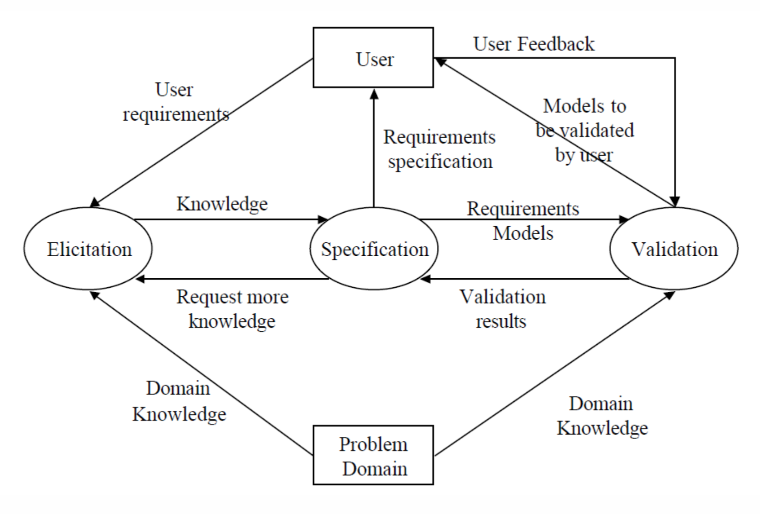

**Figure 3.2:** Iterative requirements engineering process [51]

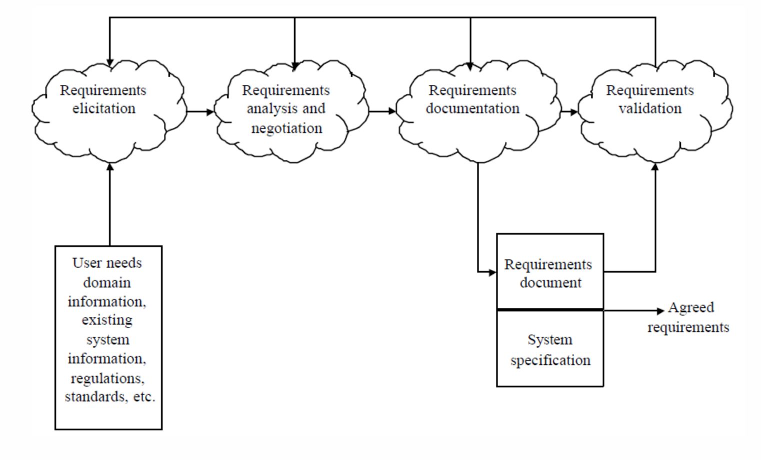
Requirements elicitation
Requirements analysis and negotiation
Requirements documentation
Requirements validation
User needs domain information, existing system information, regulations, standards, etc.
Requirements document
System specification
Agreed requirements

**Figure 3.3:** Linear requirements engineering process with iterations [51]

a RE process model would benefit from a combination of both approaches, e.g. a model that is linear in general at the beginning of the project until implementation of a prototype and continue in an iterative manner from this stage [51].

### Relation to UCD

In practice it is hard to manage and control UCD over the whole development process [68], although RE offers a good starting point to anchor UCD principles in the product. So it is recommended to evaluate the *context of use* when starting to define requirements, which means to analyze who are anticipated users and in which environment they will employ the product. Raising this awareness among all stakeholders of the RE process will increase the usability of the product to be designed later on by reflecting those consideration in the requirements specification. Furthermore concrete usability requirements can be derived from this context of use which the later on identified functional requirements need to adhere to.

Research showed that, if users are involved in the RE process intermediate results are generated at the beginning to identify weak spots, gaps or error in the requirements. Later on based on those raw results, more detailed requirements are developed which can finally be used to perform the product assessment [87]. This fact makes it difficult to apply UCD to Requirements Engineering in a sequential project workflow.

The cooperation with users usually takes place in terms of interviews or workshops as temporal locality is required for external stakeholders like users. User representatives as well as domain experts are participating only in these discussions, therefore an implementation of outgoing RE techniques are to be implemented not according to UCD principles but internally to the development team due to organizational constraints.

Beside interviews and workshops there are several other UCD methodologies that fit to the RE process, especially in the phase of requirements validation. Those methodologies are including usability tests or expert reviews which are applied to early prototypes or the requirements specification itself. The workflow description of the requirements specification are therefore evaluated with real users by comparing their workflows to the proposed ones and taking into account their feedback modifications. Furthermore sketching techniques are excellent for establishing a common understanding of all stakeholders and to reveal ambiguities and potential problems of later users caused by requirements. These early prototypes and sketches can be compared to style guides and measures defined in the requirements specification at a very early point in time if the described solution meets the expectations [89].

Summing up it can be said that applying UCD principles to a development process is best to be started during the phase of requirements analysis and specification. Involving users in this process is more easy than in the concrete development process as it is less technical and users are more likely to state their honest opinion in a creative environment as when testing an already implemented prototype. Furthermore cost of rework can be reduced by allowing for participation of users at an as early as possible point in the overall process.

## 3.6 Common Techniques

The wide field of Requirements Engineering contains various techniques and practices to be employed in various phases of the RE process, where the selection of the right technique for a given problem is necessary to implement RE effectively [18]. In general it is not necessary to select a specific technique and stick with this, quite contrary combinations of multiple techniques showed to provide the best results. The use of different RE techniques base on the project characteristics has been proofed in a case study which revealed that a pragmatic, project-oriented combination of techniques has a positive influence on the success of the developed system [53]. Further studies even evidenced the need of applying alternative approaches to analyze requirements as it often is not possible to choose the correct technique in the beginning of the process [5].

Concrete techniques in the field of requirements engineering usually aim at integrating the various view points of different stakeholders, where three main parties have been identified for usual RE scenarios. The three parties *A*, *B* and *C* as well as their relations are described by 3.4. As the most obvious participants of the RE process are peoples who represent various stakeholders, their organizations and especially the developing organization influences the whole process. This is due to the fact that individual are bound to their organizations guide lines and tend to represent their organizations interests even if their personal opinion is different. Technology plays an important role in the RE process as it is usually rough defined by the participating organizations and therefore the technical possibilities are restricted from the beginning at an organizational level. Nevertheless technology also facilitates the participating individuals by providing tools and offering possible ways of implementation and finally the produced outcome will be a technological artifact. Handling the dependencies between this various dimension as

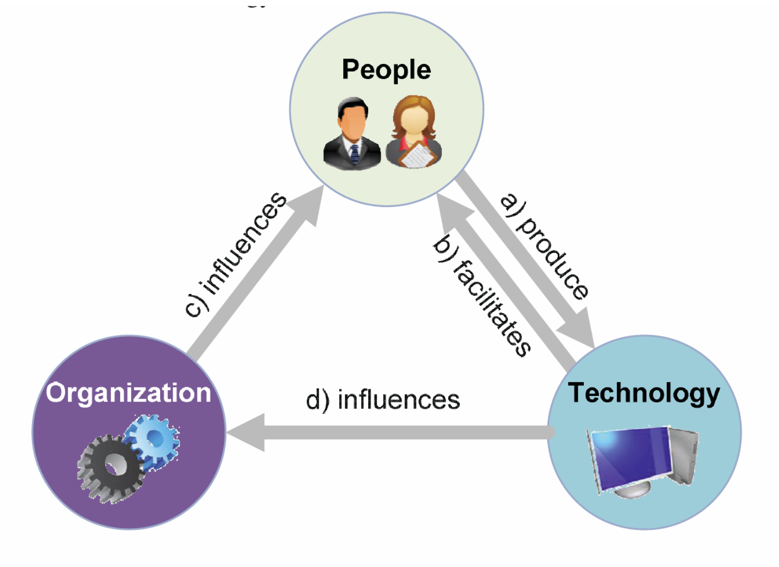

**Figure 3.4:** Three parties in the Requirements Engineering process [5]

well as different positions of participants is the main aim of RE techniques.
Practical approaches to do so mostly focus on objects, functions and states to produce artifacts that can be translated to a technical implementation of the requirements easily [32]. Furthermore this focus helps to decrease communication barriers due to prior knowledge and bias of the participating stakeholders. Widely used techniques in practice contain group elicitation, prototyping and cognitive techniques, which are often hard to apply due to varying goals of the involved parties and their physical distribution [34]. More structured approaches in use are questionnaires and surveys which are easier to implement although they lack the possibility of discussion and negotiation and therefore bear the risk of misunderstandings and ambiguities.

Requirements engineers in general should aim at facilitation of the stakeholder rather than controlling them [53]. This approach combined with adequate techniques leads to RE becoming a more cooperative process of all affected parties. In RE there need to be enough possibilities for discussion and negotiation and only detailed requirements that are agreed on should be finalized and documented to generate more stable requirements which are less likely to change during the implementation [7]. Beside the widely used techniques as interviews and document analysis [73] which are easy to implement and do not required a sophisticated process, there are several younger techniques which aim on user participation and early development of simple prototypes and sketches. Those younger techniques are often coming from the field of software engineering, especially in agile environments as those teams tend to be adventurous and give novel techniques a try. Several techniques have proven to be useful and are getting wider acceptance over the last years as they are mostly generic in their nature and therefore applicable to different domains as well.

### Interviews

A technique that is extensively used in practice for requirements analysis and gathering are interviews [55]. Various stakeholders as well as end users are presented several questions by a team of requirements engineers, which can be done face to face in meetings but as well online or using questionnaires. The questions may contain open questions as well as closed questions, although open questions are more appropriate to identify new requirements as well as gaps in existing specifications [28]. Closed questions can be an effective way to validate an existing requirements specification.
Interviews and meetings that are moderated by questions are capable of eliciting non-tacit knowledge of the stakeholders [50] where the prior domain knowledge of the interviewee as well as the interviewer plays an important role. The interviewer must be aware of in long lasting interviews and complex domains are cognitive limitations of the interviewee especially difficulties due to the limited capacity of the short-term memory as well as biases that influence responses when recalling from the long term memory. According to research one area that should receive attention in the field of interviews is recall techniques that can be used to overcome cognitive limitations of interviewees [28]. Interview questions therefore should be distinguished in domain-dependent and independent prompting schemes where the ratio of both as well as the order of questions may influence the result [9].
A categorization of interview questions is generally advised in literature in terms of question

types [46] as this categorization makes it easier for the interviewee to keep focus and keep the context of specific questions in memory.

In general it can be said that interviews are widely applied in practice, although they are mostly implemented in an ad-hoc manner. Literature and research in the field of psychology revealed several risks and misunderstandings that can be caused by inadequate interview techniques and proper interview methodologies are described. Nevertheless those sophisticated techniques are hardly used in real world projects as they require extensive setup while ad-hoc interviews often provide acceptable results.

### Workshops

The workshop technique is quite similar to interviews as it is based on social interaction and discussion. Although there are several important differences that can significantly change the outcome of both techniques. When conducting a RE workshop all stakeholders, or at least a significant group of them, are assembled in one meeting. In contrary to a usual meeting a workshop is facilitated by a requirements engineer who is not participating in discussions and may use various tools and activities for guiding the workshop. Based on this dedicated facilitation and the employed activities a valuable outcome can be ensured.

Having all stakeholders together at one big conference leads to more discussions than individual interviews as each participant is enriched by the statements of others and can either contradict or extend them. This leads to more valuable outcome and lowers the later effort for aligning the opinions of the participants.

Although there are several positive implications of the dynamics of group discussions they bare risks as well. Statements and requirements of individual participants can cause long-running and emotional discussion which endanger the positive ending of a workshop. Groups discussing requirements without dedicated facilitation tend to get stuck with details and positions of stakeholders may get hardened which makes it difficult to negotiate agreements later on. Therefore a dedicated and competent facilitator who has sufficient domain knowledge and is respected by each participant is required to perform efficient workshops. Several common techniques for group discussions and creativity workshops can be employed by the facilitator although they are not especially designed for requirements engineering.

Overall workshops should be preferred over individual interviews as they generate deeper insights and make it easier to align the requirements of all participants as they contain basic negotiation processes. Nevertheless workshops should only be chosen as a tool when there is a dedicated and competent facilitator available.

### User-stories

User stories are a special form of requirements representation and specification which originates from the area of agile software development. Requirements are stated in a less formal way by sentences expressing the users goals formulated in the end users language. A common pattern of a user story is *„As a user I want to [...].“*, where more detailed patterns take into account user groups and the larger goals they want to achieve by patterns similar to *„As a [...] I want to [...] in order to [...].“*. Although the basic principles of user stories remains to state users wishes in

their natural language, the patterns are to be seen as guidance so any kind of description of a users goals and actions are valid user stories.
During requirements analysis a list of such user stories is collected where each of them represents a single requirements for the later development. If there are complex workflows that require several user stories to be executed sequentially the concept can be extended by *epics*. An epic is stating a larger goal of a user, which is afterward split into several user stories. In the later development process, each user story is implemented as a whole before the next one is started according to agile principles. This ensures that each requirement is implemented entirely and each need of the users is met. Furthermore this leads to early testable prototypes as after each iteration of implementation there is a completed user story that can be tested by users and the stories behind can be adapted or extended if necessary.
User stories allow users to participate in the RE process easily as they can speak in their natural language and state their goals without taking care of technical constraints of formalisms. According to research, defining requirements with user stories is a way to facilitate the communication between business and engineering roles and increases the probability of capturing and meeting the customers' expectations [7].

Conclusively user stories are a way to ease the participation in the RE process for users, especially coming from non-technical domains. Nevertheless this method does not define measure to check if requirements are correctly met by the product as they treat a requirement as fulfilled when the corresponding user story and its implementation are accepted by users. User stories are especially useful in agile environments that adjust their implementation process to them and implement one complete story at a time. Introducing user stories to more sequential approaches or processes that split the whole product into components that are implemented independently, user stories can be complicate the implementation of the product.

### Prototypes

Prototypes are highly valuable for requirements engineering especially in the phase of requirements analysis as they can be used to ensure a common understanding of requirements and processes of all stakeholders. Furthermore in contrary to abstract discussions running through a demonstration scenario helps to focus users' attention [73]. Prototypes in this context can be more advanced artifacts like partial implementations but as well sketches and simulations of products. One technique that showed to be especially useful in the field of RE are *paper prototypes* which are widely used in software development.
A paper prototype is a representation where each interactive or dynamic element of a user interface is represented by a piece of paper where the respective UI element is drawled on. Those paper elements are assembled to a mockup implementation of the UI. After doing so, the user is asked to carry out a specific workflow or action by typing the elements with his finger that he would click in a real implementation. When the user has typed an element the requirements engineer reassembles the elements according to this action, therefore is simulating the systems behavior. By doing so misunderstandings and design flaws can be identified at a very early point in time before effort is put on technical implementation.
More advanced prototypes like raw implementations should be used later in the process to ensure the correctness of requirements as well as the understanding of the implementation team. Pro-

totyping techniques are especially useful for requirements re-engineering where an implementation of the system is already available. This implementation should be treated as a prototype and therefore be employed at the RE process for redesign by running through demonstration scenarios to identify usability lacks and reveal requirements that have not been covered so far. Requirements re-engineering based on an existing implementation, which can be a prototype as well, are also referred to as *artifact based* requirements engineering. Research showed that artifact based elicitation can be surprisingly effective, especially in combination with detailed questioning techniques [73].

Overall prototypes are one of the best possibilities to ensure common understanding of the system and the requirements of all stakeholders as well as they help to keep users focused. Requirements engineers need to be aware that a prototype is often the more valuable the simpler it is as users tend to retent design issues and disaffection if they think that the current implementation of the prototype already took a lot of effort. Therefore early and simple prototypes like sketches and paper prototypes should be preferred over complex interactive prototypes to gather honest feedback of users.

### Wire-frames

Wire-frames are a special type of sketches which aim at demonstrating workflows and are usually used during the design of complex software user interfaces. Executing a whole workflow in a software product usually requires the user to put several actions where each action results in a at least partial changes of the screen, as each step of a workflow is represented by different UI elements. Although simple paper prototypes and sketches are useful to design single views and interaction steps, they lack the representation of long running workflows. An exemplary wire-frame is given in 3.5.

Wire-frames cover this gap by a raw representation of each view that is to be seen at any point during the workflow. Those sketches of single screens do not need to be very detailed, they just need to contain the necessary interaction elements to proceed to another step in the workflow. After all those single screen catches are collected, each interaction element is linked to the screen which will be displayed when it is employed.

This allows users as well as requirements engineers to get an overview of the overall process and identify potential dead-ends in navigation as well as points where users can get stuck in the UI, which means they are not able to continue the workflow and have to restart.

Wire-frames are to be used for the representation of complex workflows although they are usually limited to the domain of UI design in software engineering as they are hardly capable of representing states of a mechanical system.

### Easy WinWin

The *Easy WinWin approach* has been researched by Gruenbacher back in 2000 [26] and is widely accepted in practice since then. As identified by various research projects the WinWin approach states the process of requirements engineering as heavily depending on the collaboration of stakeholders with different backgrounds, objectives and expectations. The WinWin technique aims at allowing each stakeholder to participate in the RE process and to enable them to get

their individual goals fulfilled. To do so it is necessary to provide methodology and supporting tools that are easy to understand by technically non-savvy participants although they need to be able to generate deliverables which meet the technical standard required by the implementing engineers.
To allow each stakeholder to state their opinions and motivations, the WinWin approach utilizes four types of artifacts where the first ones are *Win conditions* which are in the following extended by *Issues*. Based on those Issues *Options* are identified which finally allow to generate *agreements*. Win conditions are stating the goals and objectives of individual stakeholders where each participant is free to state as many Win conditions as there are important for him. Issues are added to those conditions by other stakeholders as they see potential problems with specific conditions. If a Win condition keeps left without Issues they can become requirements directly. The identified issues are extended by options that may either allow to adapt the Win condition to not cause the Issue or suggest possible solutions for issues without changing the Win condition. When all options are identified, the affected parties negotiate to define agreements which are added to the conflicting Win conditions in a final step. In parallel to those artifacts a glossary of terms should be maintained to ensure the correct understanding of the artifacts by all stakeholders. Generation of the described artifacts cause discussions to take place in a structured way focusing on the important parts that cause disagreements. Furthermore the produced outcome is defined in a structured way which is capable of being utilized by development teams to structure their implementation process.

An extended structure of a requirements elevation process is depicted in 3.6 and defines concrete steps to generate the described artifacts. At first a domain taxonomy is elaborated to allow for better structuring of revealed Win conditions. Consecutively the Win conditions of all stakeholders are brainstormed followed by a dedicated process step to converge on Win conditions, where the main goal is to clean out redundancies and organize the Win conditions in more abstract taxonomy elements. Beside this the used domain language is captured by establishing a glossary. The identified Win conditions are prioritized afterward and then based on their priority conflicts and constraints for those conditions are examined. Based on those constraints and conflicts are represented by structured issues and options to overcome those. In a final step agreements on those issues and options are negotiated by the affected stakeholders which are mapped to the previously defined taxonomy.

A positive affect by following this approach and the structured representation of artifacts allow for distributed and asynchronous development of requirements which is supported by special collaboration tools as well. Dedicated meetings and direct interaction of stakeholders can be limited to the steps which require negotiations on agreements, although the whole process can be implemented in terms of workshops as well. Further positive impacts by utilizing this approach have been identifier in case studies [26] and are *reduced cycle time* as well as an *increased number of created artifacts*. Reduced cycle time is achieved as the approach allows to work on multiple Win conditions in parallel as most of them will not affect each stakeholder. Therefore only the affected stakeholders are required to work on specific conditions while others can continue with different Win conditions at the same time without missing important information. The increased number of artifacts, meaning Win conditions, options and agreements which are collected in a structured manner are positively affecting the later implementation process.

### Goal oriented approach

According to goal driven approaches there are two different perspectives to be considered in requirements engineering, namely *organizational modeling* which is focused on the systems stakeholders beside *decisional modeling* which is done by decision makers on customers side as well as the side of the developing organization [23]. The goal oriented approach proposes to analyze the requirements of both parties independently and following different processes, which results in two requirements models of the system which are consolidated and negotiated afterward. The approach is mainly based on semi-structured diagrams representing each parties' point of view and their goals.

Organizational modeling is therefore divided in three phases which are *goal analysis*, *fact analysis* and *attribute analysis*. Starting with goal analysis actor and rationale diagrams are produced to provide an overview of all stakeholders and the interferences between them. This is implemented in terms of an actor diagram in which actors represent agents, roles or positions within their organizations. Afterward the the goals of each actor are analyzed in more detail producing a rationale diagram for each actor. Coming to fact analysis those diagrams are extended by facts which are valid for one or more stakeholder and which are driving their behaviors. Finally the diagrams are extended by attributes which are given a value when identifying facts [23]. The whole process is iterative where each iteration is based on the diagrams produced in the previous one.

Decisional modeling is focusing on the goals of decision makers, which means those actors that play the most important role in the decisional process [23]. This process contains the phases of *goal analysis* producing rationale diagrams, *fact analysis* extending those diagrams by facts, *dimension analysis* adding dimensions to those diagrams and *measure analysis* which are finally adding measures. Similar to organizational models stakeholders and their goals are identified and known facts are added. In the phase of dimension analysis each fact is related to dimensions that decision makers consider necessary to satisfy their goals. Finally measures are defined for each fact in respect to the relevant dimensions [23].

The goals resulting from both processes may be completely different although the acting individuals may widely overlap. Therefore this process is capable to consider the various points of view of individuals in terms of decision making and seeing themselves as stakeholders of the product, where both information is highly relevant in the later development process. After both areas are analyzed the identified requirements need to be consolidated and validated, which may require negotiations as conflicting requirements may occur between the two perspectives. Although this process takes additional effort, taking into account both perspectives allows to react to business events that dynamically happen during the projects life-cycle quickly as all the relevant information is known and relations between requirements are disclosed.

### Automated approaches

Although requirements engineering is mainly a process of social interaction and discussion, there are several approaches to automate at least parts of this process. Requirements analysis can hardly be automated as users individual needs and environments can not be captured in a structured way. Nevertheless there are various approaches for structured representation of requirements which allows for later atomization of requirements validation and management. A structured representation of use cases is contained in the UML standard, which allows for automated processing of those use cases once they are defined. One approach of formal and use-case driven requirements engineering for automated validation and tracking of the requirements has been researched [48] is possible to be implemented. Although the representation of requirements according to UML standards is not widely accepted in practice by now as it is an costly process to capture use cases in UML notation.
A different approach for automated requirements validation is described in [43], which is based on the fact that mental models of a system, as they are built by each individual during RE processes, can be represented as networks-state transition diagram which basically means to represent the system to be implemented in terms of complex state machines. Research is carried out to pathfinder networks can be used to categorize requirements and discover misunderstandings about requirements between stakeholders based on these networks in an automated manner [43]. In theory this could lead to a formal verification of the consistency of a set of requirements which is one big challenge of today's requirements engineering processes.
Finally there is also researched carried out on extracting structured requirements from natural language descriptions. A proposed tool searches informal documents used during requirements specification for terms that have been identified as quality indicators to extract a complete and structured list of non-functional quality requirements. Although is has to be stated that these tools are not capable of ensuring the correctness of the extracted requirements themselves [86].

Summing up there are various efforts on tool support and automation for the process of requirements engineering. Proposed tools are mainly focusing on later stages of RE like requirements validation and requirements management, as this are main tasks in today's RE processes and are based on processing already existing requirements. Nevertheless the process of requirements analysis requires human interaction in terms of discussion and negotiation and can therefore hardly be automated. Automated tool for requirements engineering are hardly used in practice as they mostly require a strict formal representation of requirements which is costly to generate.

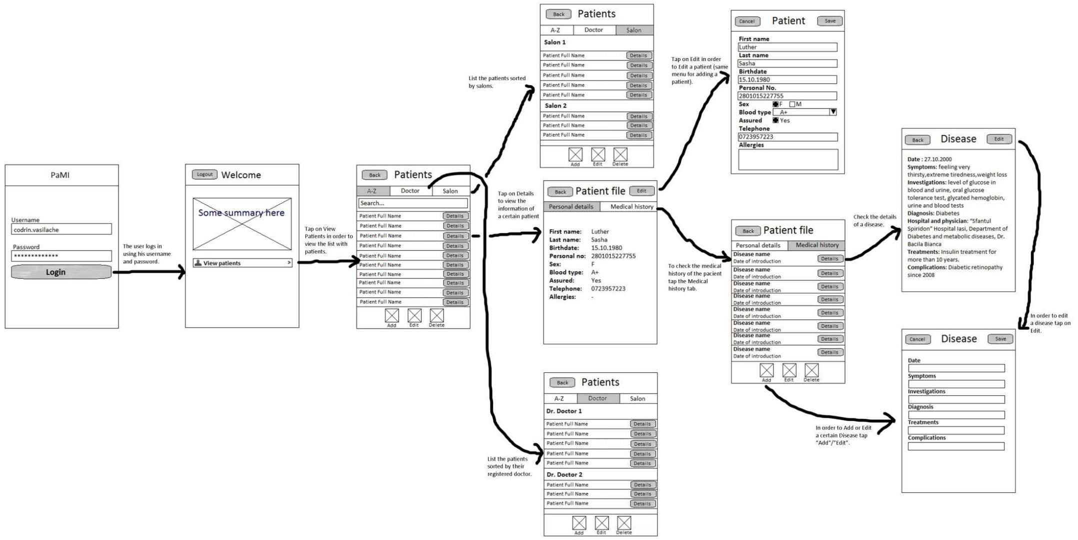
PaMI
Username
codrin.vasilache
Password
*************
Login
The user logs in using his username and password.
Logout
Welcome
Some summary here
View patients
Tap on View Patients in order to view the list with patients.
Back
Patients
A-Z
Doctor
Salon
Search...
Patient Full Name
Details
Add
Edit
Delete
List the patients sorted by salons.
Salon 1
Salon 2
Tap on Details to view the information of a certain patient
Patient file
Edit
Personal details
Medical history
First name: Luther
Last name: Sasha
Birthdate: 15.10.1980
Personal no: 2801015227755
Sex: F
Blood type: A+
Assured: Yes
Telephone: 0723957223
Allergies: -
List the patients sorted by their registered doctor.
Dr. Doctor 1
Dr. Doctor 2
Tap on Edit in order to Edit a patient (same menu for adding a patient).
Cancel
Patient
Save
First name
Luther
Last name
Sasha
Birthdate
15.10.1980
Personal No.
2801015227755
Sex ● F □ M
Blood type A+
Assured ● Yes
Telephone
0723957223
Allergies
To check the medical history of the pacient tap the Medical history tab.
Disease name
Date of introduction
Check the details of a disease.
Disease
Date : 27.10.2000
Symptoms: feeling very thirsty,extreme tiredness,weight loss
Investigations: level of glucose in blood and urine, oral glucose tolerance test, glycated hemoglobin, urine and blood tests
Diagnosis: Diabetes
Hospital and physician: "Sfantul Spiridon" Hospital Iasi, Department of Diabetes and metabolic diseases, Dr. Bacila Bianca
Treatments: Insulin treatment for more than 10 years.
Complications: Diabetic retinopathy since 2008
In order to edit a disease tap on Edit.
Date
Symptoms
Investigations
Diagnosis
Treatments
Complications
In order to Add or Edit a certain Disease tap "Add"/"Edit".

**Figure 3.5:** Example of a wireframe [63]

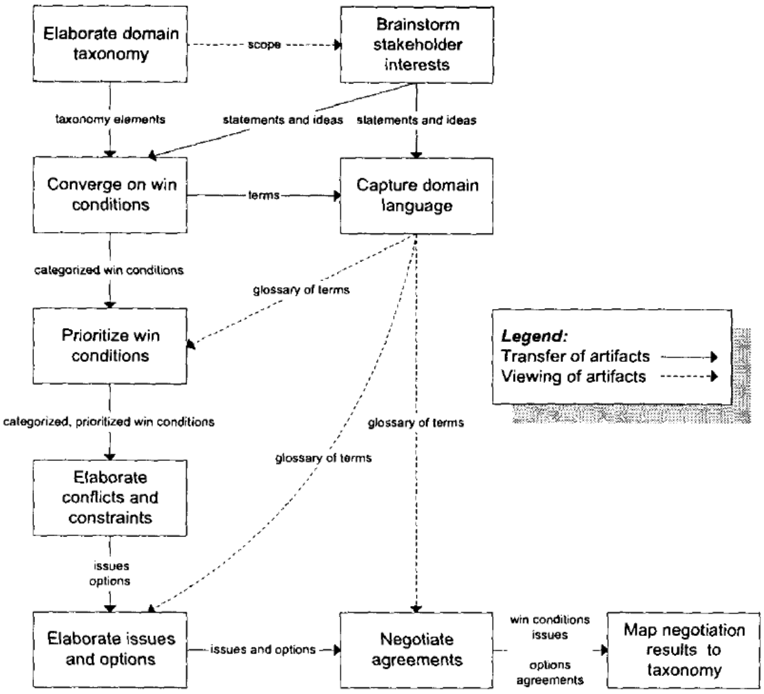

**Figure 3.6:** Easy Win Win activities [26]

CHAPTER 4

# Function Analysis

## 4.1 History

A basic definition of the techniques of function analysis was given back in 1947 for describing systems and products. Since then various techniques have evolved based on than original methodology which are used in manifold application areas. Though the basic principles from the originating definition of functional analysis remain the same until nowadays.

### Relation to Value Analysis

Function analysis did arise as a single approach but together with *value analysis* as the basic method of *value engineering* originally proposed by Lawrence D. Miles in 1947 [52]. In this original description of the methodology Miles defined three basic paradigms:

- Practical values for the customer are to be converted to functions. The functions of a product found its serviceability and provide the user with the desired worth.
- Practical values require functions which make the product fit for purpose while the prestige value require functions that cause the selling of the product.
- The recording of the desired functions of a product or the expected execution of a service are the basis for determining its value.

According to that the clear determination of the desired functions is the cornerstone of value analysis. The first of the three basic steps is the identification of the functions, where each product fulfills at least one main function and as the case may be additional auxiliary functions. Each of these functions can be defined sufficiently to define the core of the function by using to words - one verb and one substantive. This verbal denotation of functions is the base of functional analysis.

### FAST - Function Analysis System Technique

By applying this basic concept of function analysis a large amount of functions is generated to describe an arbitrary system or product, although these functions are not structured and the main function can not be distinguished from various auxiliary functions. So called FAST (Functional Analysis System Techniques) proposed by Charles W. Bytheway helps to find remedy for this problem as it is capable of not only identifying the main function but as well revealing the relations between the individual functions. To do so, nine core questions are applied to the functions which allow for the creation of a FAST-chart. Such a FAST-chart is the certificate of a concluded function analysis. Bytheway described the value of a FAST-chart as not as big as the value of thoughts and creativity which where put to its creation. Therefore the FAST-chart is just a tool to stimulate creativity in the process of its application.

The area of application of FAST broadened and FAST was used as a tool for better plannings and their enforceability as well as an efficient communication tool, which supported the cooperative understanding during planning processes. During the further procedure the focus of approaches shifted towards FAST-charts which where originally seen as a tool to stimulate creativity. Defined formats where developed which FAST-charts should adhere to and further terms like *superior functions*, *base function*, *derived function*, *constant function*, *parallel function* as well as *undesired function* where introduced. Another extension by T.J. Snodgrass denotes superior functions as *tasks* with subordinate *overall functions* as well as four *additional functions*. Those additional functions are defined as *gain/comfort*, *reliability*, *satisfaction* and *impression/prestige*. Those dimensions should ensure the acceptance of the product by the customer.

### Value Engineering Function Charts

Another extension of the basic function analysis technique from Miles are given by Arther E. Mudges principle of Value Engineering Function Charts (VEFC) which where proposed in 1965 [57]. The basic idea of this approach to develop a chart which reveals overlapping and cost intensive functions and therefore enable an objective evaluation and cost reduction of the construction without harming the essential function of the product. In a VEFC the essential functions are denoted at the top while the functions of its parts are listed below. There is no special logic in the linking of the functions, therefore VEFC is a method for the structuring of functions for the goal of cost reduction. This method relies to the basic definition of functions as verb and substantive given by Miles.

### Enhancement of function analysis in Japan

Back in 1967 professor Masatusi Tamai founded the development of Function Family Trees base on the function analysis according to Miles in his essay 'How to build Functional Family Trees' [75]. In the same year he published the book Function Analysis [74] in which he gave a critical analysis of VEFD, FAST and the method of Function Family Trees. When introducing Function Family Trees, Tamai defined the following six goals of this method:

- Inspection of the adequacy of functions

- Revealing of omitted functions
- Definition of function fields
- Revealing of relations between functions
- Identification of a development concept for the fulfillment of certain functions
- Definition of basic functions

In contrast to Miles and Mudge the Function Family Tree was described using technical terms and therefore emphasized the technical aspect of this methodology. Furthermore FAST was described as a method for the analysis of functions whereas the Function Family Tree is a diagram resulting from those analysis. Therefore it holds, that FAST emphasizes the analytical process and Function Family Tress emphasize the analytical results. In Japan the Function Family Tree is the preferred form of function analysis since 1967. This fact was documented in 1984 by a survey among 123 Japanese corporations, where 66% of those used Function Family Trees whereas only 26% used FAST and 8% different techniques of function analysis [76]. Function Family Trees are thereby still used in combination with the original definition of functions as a verb and a substantive by Miles.

## 4.2 Terminology

The term *Function Analysis* is built by the terms 'function' and 'analysis', where each of both words requires a clear definition in this context which is to some extent different to the aldays use of these words. The meaning of the terms is defined in [1] as follows:

### Function

In the field of function analysis the term *function* has three meanings:

**The effect and activity of things** 'Thing' can be interpreted in several ways, where material objects can be treated as static and fixed whereas immaterial things are defined as observable events. The term 'function' does not relate to static, fixed or observable circumstances but to operations or accomplishments of dynamic, interconnected things. Those operations are what is analyzed by function analysis.

**Functions emphasize purpose** Operations can be ascripted to various things, so there are actions in the area of natural phenomenons or random events which analysis is a challenge of natural sciences. Functional analysis deals with analysis of operations in the field of products or services which aim at serving a purpose and provide value for human beings.

**Functions are concepts** Things that are static and fixed can be defined by there color and shape, which is not the case for dynamic functions which are integrated in processes. Therefore functions are concepts and the function analysis is the process of analyzing such abstract concepts.

### Analysis

The area of management methods comprises various analytical methods, but in general the term 'analysis can be defined as follows':

- The dissection of something in its various components, elements and aspects.
- The classification and assignment of each of the different characteristics and attributes that define a concept.

Therefore an analysis can be described as the definition of a concept or object by dissecting and characterizing its essential components.

### Meaning of Function Analysis

According to the definitions above he term *function analysis* deals with dividing concepts which serve a defined purpose into their various components and defining their characteristic attributes. Subsequently the following three dimensions of function analysis can be defined:

**Impact of products and services** Goods and services have diversified effects. The goal of function analysis is to reveal these effects to define the character of a product or service.

**Purpose of products and services** All products and services have been created by there originator to serve a certain purpose. Function analysis reveals the intentions and goals that caused the creation of a product thereby defines the character of the object for analysis.

**Concepts of products and services** Each product or service is based on an idea - a concept - how it should fulfill its mission. Function analysis defines the character of the object of analysis or the service by revealing these concepts and therefore the basic idea.

## 4.3 Field of application

Products are developed to be utilized by customers to support them in their operational procedures. Nevertheless product do not only have to satisfy customer requirements but as well they must be producable and marketable efficiently for the manufacturer. This causes various problems ranging from quality and security over technical constraints and customer support to pricing and cost of use of products. To solve these problems it is necessary to analyze them, where two main procedural steps are distinguished, namely *analysis* and *synthesis*. The analysis copes with revealing the problem and its causation whereas synthesis applies methods to eliminate the problem and its source completely. To do so function analysis can be utilized in different domains, where the most important one is the field of product development beside process optimization and the design of services.

### Product Design

Product design and development in general means as well the gathering of requirements to the product or the recognition of problems as the planning of the implementation of the developed steps and the accompanying process during the phase of implementation. As a cognitive human process the development is a process which starts with the identification of goals [1]. Afterward this goals are approximated vie repeated phases of design and testing.
According to that function analysis can support product development by defining the product to be designed in terms of functions and structure diagrams. In doing so the purpose of the product is focused over the functions that are to be fulfilled by the product for the operator. Due to this way of definition the solution space is not restricted while all relevant functions and relations between this functions are recognized.

### Process improvement and Process design

A process in general defines a combination of a special kind of actions or a defined sequence e.g. a method of manufacture. In contrast to this general definition, the four areas of Industrial Engineering (IE) - manufacturing, controlling, transportation and latency/storage - are denoted as basic procedures. Are these basic procedures joined together, this composition is called a process [1].
Therefore a process is designed to shape an object of arbitrary type and a related system is characterized by a conversion process of diversified input variables (humans, material, information) into output variables like products. Process improvement therefore means enhancing such a system in terms of effectivity, whereas process design means to design such a system. Traditional approaches for process improvement have been covered by industrial engineering which caused several problems as those techniques usually focus on a detailed analysis and the recognition of dissipation and their removal. This kind of techniques are hard to apply in process design. A further issue is their focus on observable event, which means concrete conversions of the processed objects, which often hinders abstract and exhaustive improvements of the whole process while it optimizes sub-steps of it.

Function analysis on the other hand concentrates on functions as purpose driven actions of objects, in the case of processes this means the analysis of purpose driven actions which are defined for the conversion of physical objects. A functional point of view to processes shows explicitly which conversion functions a process should execute and enables an approach to development for production systems. This systems can therefore be developed based on defined processes, where these processes are analyzed and defined with regard to their functions to be executed.

### Design of services

Despite the wide use of the term *service* or *rendering of service*, there is no distinct definition for services. Things that are designed for consumers are usually divided into manufactures and services, subsequently everything that does not include a physical good and still creates value for a customer can be treated as a service. Once characteristics of services there is, that they are immaterial, which means that quality and cost are usually hard to measure. To the same extent the imaginations of producing and development are often blurred. Activities of indirect management widely share those characteristics of services and can therefore be covered by function analysis as well. Function analysis allows for the improvement of services by treating them as object of analysis and therefore focusing on their functions from a customer's perspective. According to this, services can be analyzed and structured referred to the functions to be fulfilled by them. Such a representation of services in turn enables the development of alternative solution scenarios, the recognition of dependencies between partitions as well as the identification of not needed components. Summing up it can be said that function analysis is capable of determining the components in factual manner based on its functions and enables essential improvements based on this functions.

Therefore the areas of application of functional analysis are eclectic, where the representation of the object of analysis in terms of structured function charts constitutes a level of abstraction that is common among all areas of application. The creation of this function charts differs among the different domains while the structure of the definition of functions in substantive-verb-form stays constant as well as the focus on representing the purpose and the created value of the function. Based of the abstraction of the object of analysis in terms of functions the same or at least similar techniques can be applied over various domains to reveal room for improvement and effect improvements.

## 4.4 Methods of functional analysis

The methods of function analysis can be grouped in *naming of functions* and *structuring of functions*. The process of the naming of functions copes with the definition and bounding of the essential functions of the object of analysis. Based on this functions are put in relation to each other to generate an clearly arranged picture of the system during the process of structuring of functions.

### Function Naming

The core objective of function naming are the definition, identification and clarification of functions. In particular this means to define the borders of functions and fixing their meanings as well as to recognize identical functions respectively to clearly distinguish between different functions. To do so, function naming should be generally understandable and objective.
A product can generally be defined clearly by its physical properties, although this definition is not sufficient from a users perspective as he is not primary interested in the physical properties of the product but the functions that the product performs. This constitutes the importance of function naming, which provides the possibility to describe a product in a way that reflects the concepts and goals of the customer. To do so functions are divided in *utilitarian functions* and *prestige functions*. The former fulfill a purpose for the user, e.g. a fridge is used to conserve food. Prestige functions do not influence the operative effect of a product but cover other aspects that are given value by the customer like design and presentation. During product development the goals of the customer must be revealed which means to define the core functions

of the product. These functions are consecutively translated to construction plans and a physical implementation of the product. Customer requirements can be understood as functions which therefore allow a clear description of the object to be developed.

The naming of functions is an essential tool of function analysis and contributes to the achievement of superior goals like

- Determination of functions to be developed or improved
- Evaluation of relations between those functions
- Creation of high quality products and services
- Evaluation of alternative products and services

A basic goal of function analysis is the development of value-based products and services, which means to execute the required functions efficiently in terms of costs. To do so each function has to be named appropriately to enable the following evaluation of relations between functions and possible implementation scenarios.

### Approach for function naming

To define the function of an object the question 'What is its effect' has to be answered in a structured way by using a verb and a substantive. This kind of verbal function naming conveys the focusing on the target goals as the function is expressed in an easy manner. Further advantages of this approach are the clear definition of each function and its general understandability. Furthermore this objective way of definition which is focused on the functions effects conveys novel ideas for technical implementations. Therefore the following basic rules should be considered when expressing functions:

- generally easy understandable phraseology
- well-considered, unambiguous terms
- using word that are not directly related to the object of analysis

A function is generally described by one verb and one substantive, where the following definitions need to be considered:

**Verb** The verb is the predicate, thus the meaning of the sentence and therefore the actual naming of the function. Verbs are divided into transitive and intransitive verbs. In function naming typically transitive verbs are used, as a function represents an (external) operation.

**Substantive** The substantive in this context is the object on which the function operates. Ideally substantives with a specified function are used and identifiers of parts and products are avoided. When such denotations are used this deprecates the function of its general meaning in most cases.

### Abstraction

Function naming by using verb and substantive provides the basis for idea generation and the recognition of promising alternatives. This means that a function on the one hand has to be technically correct and on the other hand fairly abstract to not reducing the room of solution. Therefore it should not be limited to specific instruments of its execution. The naming of a function using verb and substantive is an abstraction by itself as words are abstract notions, although the room of solutions is not extended when the tightness of the term used by the functions naming bounds the concept meant by it. Therefore special attention must be paid to the naming of verbs and substantives, additionally this should not be done only for single words but as well for the whole substantive-verb denomination. Although the room of solution is extended by the naming, boundaries of the meaning of the function must be defined by it as well. Functions have to be adequately concrete.

### Quantifiability

Function naming also serves as the basis to define conditions for the evaluation of ideas and alternatives. Thus the function should be easy to quantify to define the degree of fulfillment corresponding to adequate criteria. The quantifiability in general mainly affects the substantive, therefore the concerned object of the function. This should be named in a way that suggests a quantifiable respectively measurable dimension. In some cases it can although be reasonable to use a easily measurable verb as the basis for the degree of fulfillment.

### Criteria in function naming

Function naming can either be restricted to substantive-verb-form or contain additional criteria. Function in a narrow sense, meaning substantive-verb-form, servers as catchword or idea generator for finding solutions. Function in a wider sense contains criteria for the evaluation of the degree of fulfillment and for the purpose of rating alternative ideas and solutions. E.g. the function of a watch is 'displaying time', where this does not state that the current time should be displayed correctly. Via an additional criterion like the maximum deviation in seconds a measure can be created which allows for the rating of alternative solutions. Criteria of functions are dependent on customer requirements and are identified based on specifications and data. Not all criteria are required for function naming, as function analysis focuses on the customers perspective only those criteria that are based on customer specification have to be considered. Overly fixed criteria influence the process of idea generation as they limit the room of solution, although they ease the separate determination of the degree of fulfillment and therefore the evaluation of solution proposals.

## Types of functions

Functions can be examined from different perspectives and serve the fulfillment of various needs. This results in a rough categorization into the following types of functions, which relations are depicted in 4.1.

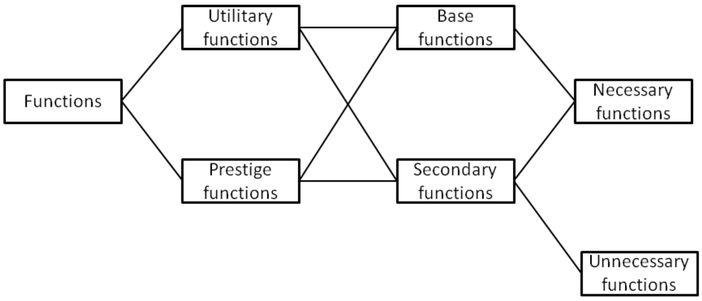

**Figure 4.1:** Function types and their relations [1]

**Utilitary functions**

Utilitary functions serve for the fulfillment of a particular functional purpose. Nearly all functions that refer to the inner mechanisms of products or their materials are utilitary functions.

**Prestige functions**

Prestige functions are meant to satisfy the various senses of the customer. This functions usually turn towards the artistic and decorative design of products. In particular fields, e.g. retail products, it can occur that the value of prestige functions overcomes the value of utilitary functions.

**Base functions**

Base functions give the right to exist to the object of the function naming, therefore they vary depending on the object and the scope of the function naming and define the actual purpose of parts of the product.

**Secondary functions**

Secondary functions are all remaining functions which support the realization of the base functions, respectively additional function that are caused by the presence of objects although they would not justify its presence on their own.

**Necessary function**

Necessary functions are functions that are required by the customer or are related to their requirements.

**Unnecessary function**

This functions are present although they are not needed, therefore they provide no additional value. Unnecessary functions can be removed by changes to the construction. This category also includes functions that become unnecessary due to changes to the constructions. The removal of such functions can contribute to the cost optimization of products.

## Function structuring

The process of function structuring aims at revealing relations between functions and ordering functions according to a defined logic. To do so the functions are depicted in terms of a diagram. For the creation of a FAST-Diagram the *How-Why-Logic* is used whereas a Function Family Tree is based in the *Purpose-Medium-Logic* [1].

There exist several possibilities to express a circumstance, where the main ways comprise descriptive explanation, explanation via cause-effect relations as well as explanation based on goals. Function analysis makes use of explanation based on goals, thus a purpose-medium-logic, where circumstances are analyzed with respect to their nature to reach defined or revealed goals.

Products are in general described based on their observable appearance, thus their physical characteristics, whereas services are describes by involving their business activity and behavior. According to function analysis each product and service aims at fulfilling specific functions. Therefore each product as well as each service can be captured by capturing and structuring the functions that they execute. To do so, FAST-Diagrams and Function Family Trees are utilized.

Function structuring is, as well as to function naming, a basic step of procedure in the course of function analysis and primary pursues three goals:

- Determination of the next higher function that is to be fulfilled by the product or service
- Determination of relations between those functions
- Creation of alternatives to the analyzed products and services

Products and services are decomposed to multiple function where some of them are partial functions and others are overall functions. Function structuring reveals partial and overall functions and thereby defines explicitly which functions the product or service must fulfill. The identified functions are interconnected in manifold manners. Some functions only exist to execute other functions and some can not be realized without others. Therefore functions can not be analyzed independently and isolated as they are interconnected in many different ways. Furthermore function structuring reveals the philosophy that stand behind the development of a product. When relations between functions are depicted using FAST-Diagrams or Function Family Trees, this allows for the creation of alternative solutions for parts of the product and recognizing their interplay with other functions and parts of the product at an early stage.

### FAST-Diagrams

The structuring of functions using FAST-Diagrams serves to detect goal functions and to search for the respective next higher classified function by asking *why*. This process stimulates the creativity and allows to focus on the purpose of functions and thus to identify alternative solutions. The persons that are engaged in the analysis of the object are in general also engaged in the creation of FAST-Diagrams. The creation process of the diagrams helps to improve the communication among team members by supporting the collaborative capturing of the functions and their relations. The FAST-Diagram widely corresponds to a bottom-up approach where by selective questioning for the 'why' of functions the superior functions and goals are identified.

### Function Family Trees

Function Family Trees are based on the purposive-instrumental-logic. Thereby a goal function gets assigned multiple instruments that are required to fulfill this function. So in general there are various instrumental functions related to a single purposive function or are subordinated to it. The instrumental function on the same level are thereby independent of each other and only related by their common purposive function. The top-level purposive function or when indicated also several purposive function that have no superior function can therefore be treated as the base functions of the product, where all subsequent instrumental functions can be treated as secondary functions which are necessary to fulfill the base functions. As instrumental functions on the same level are independent of each other, this means a fragmentation to various components of the system. A Function Family Tree therefore is proper to apply a top-down approach where, based on base functions, the required instrumental functions are identified by asking 'how'.

A combination of both approaches seems legit and reasonable, where based on a given function according to FAST-Diagrams its base function is identified and by using a Function Family Tree the therefore required secondary functions are determined. It has to be considered, that independent of the combinability of the approaches, they both use slightly different forms of representation of the resulting diagrams.

## 4.5 Functional analysis in practice

The practical fields of application range from product improvement and process improvement over product- and process development to improvement of immaterial services. Although the individual areas differ in respect to details of practical implementation, the basic concepts are common among all of them. Due to that, the following investigation focuses on the application of function analysis for product development and improvement.

Product development means the alignment and determination of characteristic attributes of a new product that should satisfy a certain need of a customer. Product development with the aid of function analysis is composed of the three procedural steps of *Capturing the object of analysis*, *function naming* and *structuring of functions* [1].

### Capturing the object of analysis

Product development is usually based on customer- or internal corporation requirements, which are defined by a requirements specification. Although it can happen that this specification is not complete and product development based in inadequate planning documents takes place. Beside sound documents, the application of function analysis requires the gathering of required information and the complete capturing of all requirements to the product. Are all necessary requirements and information collected they have to be structured to get an overview of the required functionality. After a classification of the heterogeneous information they can be collected in a information-form and integrated into development. Thereby a compromise between functions, production capacity as well as caused cost needs to be found. It is important that all requirements and constraints are considered when doing so.

### Function naming

Functions are effects that the products that are to be developed should cause. The explicit definition of this functions is therefore the starting point of the development process. Due to that the functions that allow the derivation of product functions are selected among all the already developed functions. The gathered requirements contain demands to the product that can be rough partitioned into its functions, distribution and economic aspects as well as fabrication and supply. Some of those requirements can be translated to product functions, although this is not possible for all of them. Such requirements that can not be translated to product functions nevertheless have to be considered during the development process, as they affect the construction of the product without directly influencing its functions. To do so this functions are converted into criteria, attributes or solution caused constraints for the identified functions.

For each requirement that has been selected for function naming the corresponding function is denoted in substantive-verb-form. The respective requirement is described in an narrative form which has to reflect the function to be executed be the product exactly.

All functions that have been identified underlie certain restrictions in respect to the degree of fulfillment. The following step is therefore the definition of this restricting measures and conditions and their assignment to functions. Such criteria are based on requirements to quality, distribution and production. A essential question hereby is 'Which criteria restrict the functions?'. The final result of this step is a list of functions that have to be implemented by the product, where each function may have assigned several criteria which restrict it or define its measurability and degree of fulfillment.

### Structuring of functions

Each function that has been captured during the phase of function naming is transferred to a *function card* [1]. There have to be as many function cards as there are functions in the list, where the substantive-verb-form of the function is denoted on the card at the proper location. At first one function is selected randomly and its goal is analyzed by asking 'why?'. Consecutively the one function card is selected that answers this question and is put to the left side of the initial card. If there is no card containing a function that answers the question appropriately a new card with a new function is created and put to the left side of the initial card. This process is repeated until a answer is outside the scope of the current goal of development. The functions are structured by applying this process to all the initial function cards. In case it is not possible to produce explicit goal functions, the function namings have to be reconsidered or the collection of functions has to be completed.

When all functions are positioned according to their functional sequence, the correctness of this sequence is approved by asking the following questions:

- Can the goal function be satisfied by its instrumental functions reliably?
- Are the instrumental functions dispensable when the goal function is dispensable?
- Are the base functions that the product has to fulfill depicted completely?
- Can the concept of product development be derived from the functional sequence easily?

Functions should be added or corrected when:

- instrumental functions do not satisfy the goal function reliably
- instrumental functions serve others than the necessary goal functions

The corresponding criteria to the functions are afterward added to the final Function Family Tree. When all criteria that have been identified by the process of function naming are enlisted in the tree it has to be checked wether all criteria that are caused by development requirements are represented in the tree and they have to be added in case.

The resulting function tree represents a technical basis for the execution of product development. The tree contains as well main functions as additional secondary functions that have to be fulfilled by the product. Due to that the function tree becomes the basis of solutions that satisfy all required functions and therefore contributes to the completion of concepts.

The functions that are depicted in the tree are of different kinds, where the overall product development is completed by implementing the main function as well as the instruments to fulfill this functional sequence and those of the secondary functions. Thus the final step of function analysis is the definition of this main functional sequence in the function tree. which meas the one functional sequence that defines the basic procedure of product development. In most cases there is only one main function which covers the most important basic function, in some cases there may occur two main functions [1].

### Adaptions for product improvement

Product improvement means the redevelopment of a product based on an existing product to achieve improvements, optimize costs or increase quality. The procedure for product improvement is very similar to the one used for product development, where prior to the capturing of the object of analysis the topic of the improvement needs to be defined which influences the subsequent activities respectively their input data.

The selection of a topic means the selection of a certain product of the sortiment to be improved or, in most cases, a certain component of a product. Is this component used in multiple products, a directed improvement of the component can cause simultaneously improvements of multiple products. In rare cases product improvement means improvement of the whole product [1], where the reason is that problems like quality deficits or cost usually are not caused by all but by certain parts of the product. Once a topic for improvement is selected its boundaries must be clearly specified to define the range in which function analysis is applied.

The following steps for *capturing the object of analysis*, *function naming* and *structuring of functions* take place as in the case of product development. Thereby it has to be considered at the time of analysis, that only requirements and criteria are taken into account which affect the selected topic and such that affect different parts of the product are not brought to the following process. At the time of function structuring respectively the identification of the goal function, the scope of improvement that is defined by the topic has to be considered to know when the appropriate level of abstraction is reached.

## 4.6 Discussion

The methods of function analysis originate back in 1947 in the first proposal of the technique by Lawrence D. Miles and have been improved and developed since then, thus this methods experienced a much longer development process as many of today's techniques for product design and product development especially in the field of software engineering. The methods of function analysis are per se not appropriate to carry out classical requirements engineering as they assume that the requirements to the product are already known. Although function analysis constitutes as a connector between requirements gathering and the product implementation itself by translating the unstructured requirements and needs of customers into a structured representation of the product. Function analysis is therefore a appropriate way to derive a structured representation of the product to be developed from qualitative descriptions on different levels of abstraction, where it takes care that all the gathered requirements are included in the final representation of the product, measures are defined in an appropriate way and a sufficiently technical depiction of the product is created to support the following process of product implementation.
The techniques of function analysis mainly originate from the field of manufacturing processes and their management wherelse many of today's common techniques for requirements engineering originate from the field of software engineering. None of the widespread processes attempts to combine the advantages that arise from both methodologies that originate from different domains. Although commonalities can be seen at various points, so the differentiation of functions and criteria basically equals the differentiation in functional and non-functional requirements. Similarly the analogy between base functions and instrumental functions to user stories and epics in agile software engineering is almost obvious.
This at least partial similarity of basic concepts suggests to combine the benefits of both approaches, especially as they apply at different phases of the overall development process and therefore could setup on each other.

CHAPTER 5

# Integrated Approach

The preceding investigation on common requirements engineering techniques in the field of software engineering as well as User Centered Design and Function Analysis reveals that there is potential for approaches which combine those disciplines. Although all of those fields of study are related to product design and definition of requirements they originate from different fields of application where the mentioned techniques for requirements engineering are common in today's Software Engineering projects, especially in an agile environment. UCD and especially Function Analysis on the other hand are widely used in production environments which focus on manufacturing processes and product design in general.

There have been several approaches in the past that combined best practices from one field and either combined them to existing approaches in a different one or even ported whole approaches to a different domain. The most widely used of this approaches is Kanban in the context of agile software development. The process of Kanban defining a production process by different phases which provide the input for following stages based on a pull-strategy was initially implemented at Toyota in the field of car manufacturing. Nevertheless the abstract process was translated to the needs of lean software development and serves as the basic development process for agile software development teams nowadays [47] [41]. Success-stories like Kanban show, that combining approaches which are originating from different domains offer large potential for improvement of development and design processes.

## 5.1 Potential

The desccribed approach is combining the strengths of agile requirements engineering in the field of software development and the strengths of Function Analysis and UCD which is widely applied in manufacturing domain. Agile requirements engineering offers lots of tools to discover requirements of users in their totality, including latent knowledge of users. Most of those techniques are mainly based on the involvement of the users of a product which makes those techniques very tightly related to User Centered Design approaches. On the other hand most of those techniques lack the capability to develop a structured, abstract representation of the product to be developed. In common approaches requirements are collected in a quite unstructured manner in terms of user stories or similar textual representation. The entirety of those user unstructured requirement is referred to as *product backlog*, which is during the further procedure implemented step by step without a structured preparation. This leads to the fact, that agile processes in software development are capable of dealing with this unstructured backlog of requirements by implementing iterative approaches which make them highly adaptable to changing requirements as each iteration focuses on some particular elements of the backlog. Nevertheless this approach provides high flexibility to uncertain domains, it is not adequate for all kinds of projects. Especially projects in a well known domain like the re-design of a product or projects causing traditional development models for any reason loose the capability of generating a structured representation of the product and therefore upfront planning if they employ this kind of requirements engineering.

Function Analysis on the other hand provides the capability to prepare and abstract description of a product to be developed at the design time. This allows for more accurate planning in later stages of the development process. Function Analysis allows in particular to generate a structured description of the product in a way that is sufficient for the following implementation process to be based on this description. Nevertheless Function Analysis by itself does not define how the initial requirements for a product are collected. Therefore this approach can be combined with UCD techniques for collecting requirements directly from users an das well be well supported by requirements engineering techniques from agile software development. In later project phases Function Analysis is as well capable of dealing with changes to requirements at a later stage in a structured way which is currently implemented by a unstructured backlog and flexible development process in the field of agile software development.

## 5.2 Combination of techniques

The described approach is referred to as *Function-Requirements-Elevation* as requirements engineering in a way that aims at apllying function analysis directly to the revealed customer requirements. To do so common techniques of requirements engineering are used to reveal functional requirements as well as both functional and non-functional requirements of users which are contained in latent knowledge. This process of identifying customer requirements goes hand in hand with the principles uf User Centered Design, which allows for discovering of the individual requirements of specific groups of users, which may differ drastically between groups. This fact is not especially targeted by most wide spread approaches for requirements engineering nowadays. Requirements are collected in a semi-structured way which allows for an easy translation to functions, where each function correponds to a requirement that has been identified beforehand. Based on this functions a Function Family Tree is developed, which represents the product to be developed while it contains all the users requirements. Changes to this requirements at later stages can be added to this Function Family Tree as well as this tree allows to define the course of action for the product implementation basedon the prioritization of paricular features.

The fact that this approach is explicitly able to deal with requirements of multiple user groups offers high potential for the development and design of products which are used by diverse types of user which is especially relevant today in the area of gender sensitive product design. Futhermore this fact makes the proposed approach as well applicable for Lead-User based development approaches. The whole approach does not require the user to be aware of Function Analysis by using the strengths of nowadays techniques for requirements engineering and user centered design which allow for the participation of users without any previous knowledge of the techniques.

The main steps of the integrated approach for requirements engineering which is proposed in this work are depicted in figure 5.1. Each step is described in more detail in the subsequent sections. The two last steps of the depicted process are seperated in terms of color as they are not crucial for the requirements engineering process itself. Nevertheless they can be useful in specific situation when either there is an existing implementation of the product that is to be revised or a more detailed analysis of the differences between the user-groups is required.

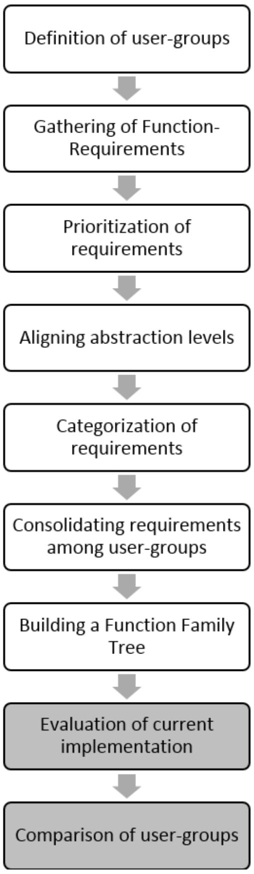
Definition of user-groups
Gathering of Function-Requirements
Prioritization of requirements
Aligning abstraction levels
Categorization of requirements
Consolidating requirements among user-groups
Building a Function Family Tree
Evaluation of current implementation
Comparison of user-groups

**Figure 5.1:** Main steps of the proposed integrated approach

**Defintion of User Groups** At the very beginning the various groups of users need to be outlined and each participating user is assigned to their corresponding groups. This step is optional as defining just one large group of users is valid as well and will simplify the further process. Nevertheless the capability of handling requirements of multiple distinct user groups is one of the biggest surpluses of this approach.

**Gathering of Function-Requirements** For each user groups requirements are gathered separately. In contrary to common requirements engineering techniques the resulting requirements are documented in terms of structured functions instead of user stories.

**Prioritization of Requirements** The collected requirements are prioritized by classifying them in a predefined scheme. The prioritization is done for each group individually with participation of users as well as the product design team.

**Aligning abstraction levels** Each of the collected requirements needs to be transformed to fit to a certain level of abstraction. This step is performed by the product design team based on the users Function-Requirements.

**Categorization of Requirements** The Function-Requirements which are defined on an common level of abstraction are categorized in modules of the product to be design or in clusters collecting Function-Requirements that correspond to related features.

**Consolidating Requirements among user groups** The Function-Requirements of all groups are merged together, where duplicates can be eliminated. Contradictions in requirements between groups should be identified and handled at this point.

**Building a Function Family Tree** Based on the resulting list of Functional-Requirements a function family tree can be built which serves as guidance for the further implementation process as well as requirements management.

In the case of product re-design, two additional steps become necessary:

**Evaluation of current implementation** After the step of *Prioritization of Requirements* the current implementation of the described functionality is rated by the participating users. The evaluation may range from 'perfectly implemented/ do not change' to 'not usable/ not implemented at all'.

**Comparison of user groups** Product re-design is often triggered a the fact, that specific groups of users are not satisfied by the product. To counter this situation, before the phase of *Consolidating Requirements among user groups*, the requirements of the analyzed user groups are compared to each other. Doing so may reveal insights about which requirements are specific to a certain group and which are common among several groups of users. This knowledge allows to prioritize requirements for the following re-design process as well as it enables tailoring of the product to fit the needs of specific groups in particular

## 5.3 Definition of User Groups

In a first preparative step to requirements analysis the future customers of a product need to be identified. Doing so is a necessity for each requirements engineering process. In case the potential user base is very large as it may be the case for retail products, the field of User Centered Design provides various methodologies to select an appropriate subset of users that represent the overall customer base and are willing to participate in the process of product design, e.g. the Lead User approach.

Once the users that will participate in product design are identified, they should be classified into several user groups according to specified criteria. The criteria which are characterizing the user-groups are not limited an may include biological properties like user's age, body height or gender as well as socially defined properties like education, experience, income or marital status. Depending on the product to be designed there may be various factors which are supposed to influence users requirements and view to the product. Each of those factors should be expressed by a characteristic which can be measured for each participating user and therefore allows to divide the users into initial user-groups. Selecting participating users and defining the criteria that divide them into groups should consider the fact that each group should be about the same size of users. If one group is noticeable larger than the others, this may result in the fact that the requirements of this group gain a higher presence in resulting Function Family Tree and therefore the resulting product.

Once the user-groups are defined and each participating user is assigned to one group the initial definition of user groups is finished. Nevertheless it is absolutely valid to re-assign certain users to a different group at any point of the process if it turns out that they are not comfortable with the received opinion of their current group. In this case, the relevant users may be simply assigned to another existing group and participate there for the following process. In special cases it may also happen, that the initial definition of user groups turns out to be not valid in parts or at all. This is likely to be the case when there is a user-group that hardly finds a consensus on their requirements and priorities, or if several users are misplaced in their groups. In this case, the respective users can be arranged to a completely new group. If the user groups seem to be inappropriate overall it may be the best choice to redefine all user groups and start the whole requirements gathering from scratch.

## 5.4 Gathering of Function-Requirements

Once the user-groups are set up, each group developes their requirements. Similar to agile techniques which often make use of user stories, requirements are collected in a defined format where each member of the group should participate and state their personal requirements.

In contrast to user stories, requriments are defined as functions, which means they are deonted as a verb and a noun. This representation is referred to as *function-requirement*, where each function-requirement should describe one specific task that a user wants to be accomplished by the product. At this point the level of abstraction of this requirements is not defined, therfore the functions may describe very specific parts of the product as well as the overall workflow that is represented by the product. Nevertheless more detailed function-requirements should be preferred as they allow for a more detailed product description later on. Similar to many other agile techniques, requirements are collected during moderated discussions or workshops with a group of users usually. In some cases it may also be more adequate to have individual interviews with single users to identify function-requirements. Nevertheless it should be considere, that group dynamics may reveal unconscious requirements that would not have been collected in personal discussions. A especially useful technique to collect function-requirements are *focus group discussions* [40]. In particular this means widely free group discussions, where one moderator guides the discussion to avoid sticking at details or focussing on special areas of the product that are the considered as especially important.

Once a user discovers or states a requirement that they want to be fulfilled by the product, they write their requirement down on a special function card, which defines dedicated areas for formulating a funtion in the verb-noun format.

## 5.5 Prioritization of Requirements

The function-requirements that have been collected in the previous step need to be weighted according to their priority for users. This is not done immediately during the phase of requirements gathering for two reasons. First, users should get an overview among all the identifier function-requirements to ensure correctnes of relative weightings among requirements, second is that it may take other users than the one who identified the requirement some time to fully understand the impact of the requirement in the overall product context. As all users should participate in prioritization of requirements it is reasonable to perform this weighting in a separate step.

In the most simple case requirements are just marked as *important* or *not important* where the first group collects requirements that are essential to be met by the product whereas the second group collects requirements that represent nice-to-have features. In this scenario it is very likely that the majority of requirements will be marked as important unless the phase of requirements gathering lasts very long to give the participants enough time to identify nice-to-have requirements once they are done with collecting crucial requirements. In more complex scenarios each kind of graduation of importance is valid which may be expressed by labeling as well as by a set of numbers. The important thing to keep in mind at this phase is, that requirements are prioritized relatively to each other as this prioritization affects the sequence of implementation later on. Nevertheless it is valid to assign the same relative priority to several function-requirements to express that they are equally important and there sequence of implementation is not defined at this point and therefore driven by technical constraints. Assigning a unique priority value at this stage would mean to force the users to define the complete implementation sequence which is not reasonable from a technical perspective.

A different approach for prioritization is to order all the collected requirements of the group according to their importance. Nevertheless this approach will serve it's purpose of a relative ordering it is hard to implement as this phase of prioritization takes place separately for each user-group. This is necessary as only the users of the corresponding group participated in the discussion that revealed the requirement and therefore are able to see the function-requirements in their context. If the collected function-requirements are ordered corresponding to their importance in groups without assigning them a weighting value, this makes it hard to merge the lists of all groups without an additional discussion of all function-requirements with all users. Although this is possible and may be reasonable in some situations, in most cases this discussion will be very long-lasting and furthermore is likely to affect formulation of particular function-requirements or even remove some of them which should not be the case at this stage.

As a result of this phase the initial list of function-requirements for each group should be ordered by their importance of implementation where each function-requirement should have a value or label assigned which expresses its priority relatively to other function-requirements.

## 5.6 Aligning abstraction levels

Function Requirements when expressed by users are likely to be on different levels of abstraction. Where in some cases requirements of users are very concrete and even dictate parts of the technical implementation, there are other requirements that are rather abstract and just describe the desired behavior of the product without any technical implications. Nevertheless all of this different kinds of requirements are valid and equally important from a users perspective and therefore all of them need to be gathered.

To ensure comparability of requirements and identify duplicates among user groups in later stages it is necessary to align the levels of abstraction which means find a description for each requirement which resides at a defined level of abstraction. Aligning the abstraction levels is the first phase in the process which is done by the design team and does not require any user participation. Nevertheless there may be some additional information from users required in special cases which can be collected selectively.

In general there are four levels of abstraction which function-requirements should be assigned to as they arise from the phase of gathering requirements. At each of this levels it is possible to describe the users wishes or intended behavior as functions in the noun-verb form. The different abstraction levels can be differentiated as follows:

**Technical level** At this level of abstraction technical deficiencies or necessities are described. Functions at this level do not take into account the users intention or implications to their work-flow. This is the most abstract level of functions that can be transformed into user requirements although the users perspective is not included at this level of abstraction.

An example for a function-requirement at the technical level is ’fan causes noise’, which is a purely technical description stating that the fan of a device causes some noise and not including how this affects the users and what are the wishes of a users regarding the fan and its noise.

**Personal level** The personal level describes which problem should be solved for the user. Based on the technical abstraction level there is some behavior of the product which either causes a personal problem for the user or solves one. Functions at this level are user-centered without considering any technical implications or constraints, they just express the motivation of the user to state this requirement.

An example for a function-requirement at the personal level, which may be related to the example from above, is ’concentration is reduced’ or ’ambient noise is drowned’.

**Requirement** At this level of abstraction the real requirements that the user wants to be fulfilled is stated. Those requirements are formulated as function in the noun-verb form as well. Based on the technical description and the problem that the user wants to be solved, a requirement states clearly what should be fulfilled by the implemented solution.
Continuing the example from above, a function at this level is 'reduce fan noise'. If the used language is English, there is a rule of thumb saying that functions at the technical and personal level usually start with a noun followed by a verb, where functions at the requirement level are started by the verb.

**Proposal for solution** In some cases it may happen that users have concrete suggestions for technical solutions of requirements. It may even be the case that users are able to spontaneously describe the solution they want to be implemented without being clear about what their requirement behind this solution is. As suggestions for solutions of users are valuable information for later implementation this information should be collected as well, furthermore it may help users to realize the requirements behind their suggestions.
According to the fan example, a function at this level is 'reduce fan-speed'.

As it can be seen from the examples above, there may be multiple function-requirements at a particular level which are related to the same function-requirement at a different level.

During the phase of aligning abstraction levels for each function stated during the phase of requirements gathering a function-requirement at the requirement level has to be found. To do so in a first step the functions that have been stated by users need to be assigned to the appropriate level. Once this is done, functions at the technical level can be transferred to the personal level by asking 'what problem does this behaviour cause' or simply asking 'why'. Functions at the personal level can be further transferred to the requirements level by asking 'what does the user want' or 'what has to be changed/done'. Functions that are initially formulated as a proposal of solution can be transferred to the requirement level by asking 'why should this be done'.

When this step is completed the function-requirement at the requirement level should be defined for each requirment from the phase of requirements gathering. It is not necessary to fill all the other level for each requirement, although it is recommended. Especially by defining the technical and peronal level of function requirements a lot of context is can be described very efficiently. In the end the description of a requirement in all three level (technical, personal, requirement) is as expressive as a textual description or user story, although it consists of only 6 word and is stored in a structured representation.

## 5.7 Categorization of requirements

To allow an overview among the collected function-requirements they should be categorized into several groups. The definition of groups my be widely technology driven which means each component of the product is mapped by a specific group but as well usability driven as there may be groups of requirements that are defined by special usability properties that they are related to. There may be as many groups defined as it seems necessary to provide a reasonable semantic clustering of requirements. Nevertheless each function-requirement should be assigned only to

one single group. If it seems necessary to assign one requirement to multiple groups, the definition of the groups should be reconsidered.
Categorization of requirements is not vital for the overall process, nevertheless it eases the following phase of consolidating requirements among user-groups- Furthermore especially in the case of product re-design this semantic clustering of requirements may provide valuable insights on which group of users is especially affected by particular parts of the product or which features are especially favoring particular groups of users.
At this stage it is sufficient to process the function-requirements at the requirements level from the preceding phase of aligning abstraction levels. Other abstraction levels should be stored for contextual information about particular requirements, although they are not explicitly needed in the following phases.

## 5.8 Consolidating Requirements among user groups

Based on the categorized lists of requirements, the goal of this phase is to build a sound and consistent description of the future product in terms of function requirements. To do so the function requirements of all participatin user-groups have to be merged. Although merging in this case basically means appending the lists, there are several special cases which need to be considered to avoid conflicting requriments as well as redundancies. To keep an overview among the requirements of the various groups it is recommended to use the categories developed in the preceding phase and merge one category after another.
To perform the merging, the requirements of one arbitratry group are taken as they are as the initial list. The plain list of requirements is extended by columns that indicate for each requirement in which groups this particular requirements has been identified. For each following user-group each requirement to be merged into this list needs to be compared to all the other requirements that are already contained. In most cases it is sufficient to compare functions at the requirements level, although it may happen that there are functions that seem very similar at this level. Once this occurs all the other abstraciton levels of this requirement should be considered for comparison to decided if this requriment is really redundant among groups or just very similar. In case it is just a similarity both requirements are added to the resulting list as they are. In case of a redundancy, both requirements are merged into one entry in the final list and get an additional marker indicating the groups which raised this requirement.

In case that there are requirements that are likely to cause conflicts or are obviously contradictory these requirments need special treatment as well. The first approach to resolve the contradiction is trying to find a technical solution which means a way of implementation that would serve both requirements and describe this solution by a new requirement. If this is possible, the newly created requirement is added to the resulting list while the initial contradicting requirements are removed. An example for this scenario is requirement A from group A stating 'decrease machine height' while requirement B from group B states 'increase machine height'. At first sight, this requirements may be contratdictory although there may exists a technical solution described by the requirement 'adjustable machine height'. In this case requirement A as well as requirement B are not part of the resulting list as they are represented by the newly created one. This newly created requirement although gets the markers indicating that this was

requested by both groups.
In case the requirements are really contradictory and there is no possible technical solution, the affected user-groups need to be consolidated, describing the conflict in requirements to both of them and trigger a new discussion to find a solution for this contradiction. In practice contradictory requirements are quite rare, although this process of initiating a new discussion is time intensive, therefore a possible alternative may be to just decide that one of the affected groups is considered more important for the products marketing strategy or considered more relevant for other reasons. In this case the requirement of this group may be just taken as it is, while the requirement of the other group is simply ignored. Although this approach is not clean in terms of the described approach, it may be reasonable in special situations in practice.

## 5.9 Building a Function Family Tree

Once all function-requirements at an adequate level of abstraction are collected and categorized, building a Function Family Tree is straight forward according to the process described in [1]. The Function Family Tree to be built does not explicitly reflect the particular user groups that have been used for gathering requirements as the Function Family Tree mainly serves as guidance for product implementation. As teh rough structure of the Function Family Tree is widely defined by various components of the product the assigned categories of function-requirements serve as a good starting point for building this tree.
The Function Family Tree which is initially built at this stage serves as an living artifact of documentation, which means if user's requirements change during the development process, this changes must be added to the Function Family Tree. As the Function Family Tree is the main artifact for communication between the design team and the team for technical implementation it needs to be kept actual at any time. This also means to mark areas of the tree that have already been implemented by the technical team, which means that design changes at this components are more cost intensive and should only be done if absolutely necessary. Finally, as the Function Family Tree is a structured representation of the complete product and its features it may also be used a tool for customer communication. Function Family Trees proved to be useful to explain a complex product and the contained technical constraints and dependencies to customers. This is especially the case when prototypes of the product are built in higher frequencies to allow for user tests as decribed by User Centric Design.
As a guidance for product implementation the functions contained in the Function Family Tree are priorized accoding to their customer value, which basically reflects the prioirty values that have been collected when gathering function-requirements 5.4. This assigned values lead to a sequence in which the different features should be implemented, which is very similar to an agile or lean development approach.

## 5.10 Product re-design

In the case of re-designing an existing product, some additional steps in the described approach may become useful or even necessary. This additional steps are related to evaluation of the current implementation which may be a valuable indicator to decide if the re-design and re-implementation is really adequate and how much increased value can be expected therefrom. On the other hand this steps allow to evaluate how adequate the current implementation is for particular user groups and which groups of users are served best by the current implementation. This analysis are especially valuable for product strategies and marketing activities.

### Evaluation of current implementation

Evaluation of the current implementation makes it necessary to created a sound description of the current implementation in terms of functions. This kind of product description tends, depending on teh complezity of the product, to become quite large. Due to this fact it should be considered if just a particular component or aspect of the product shoudl be evaluated to decrease the required effort to build this tunfion-based description.
The creation of the list of function should be done by domain experts from the developing organization based on their knowledge as well as availble documentation like manuals and technical documents. The process of creting the list of functions should focus on properties and capabilities of the product that provide value for users or at least are designed to do so. THe question to be asked for each function shoudl be similar to 'which user need is fulfilled by this?'. By doing so the resulting description is a list of functions-requirements at the requirement level of abstraction. More details on how to describe existing products in terms of functions are defined by classical function analysis in [1].

Once this kind of list describing the existing product is created, it is compared to either each list of function-requirements of the user-groups or to the already merged list. It depends on the goal of the analysis which lists are to be compared. When comparing the lists three types of function-requirements need to be distinguished, which are *implemented and required*, *implemented but not used* and *requrired but not implemented*. Obviously a function requirement is of the first category if it occurs in both of the compared lists, while it is part of the second goup if it occurs just in the decription of the current implementation and to the last one if it was requested by users but is not implemented currently. From the results of this comparison it can be derived how many of the required functions of users are already implemented as well as which and how many of teh currently available functions are not employed by users and could therefore be removed during the process of re-design.

### Comparison of user groups

In the context of product re-design it may become valuable to analyse which groups of users are surved how well by the current implementation. In a more scientific, nevertheless practical relevant, setting it can be of interest to analyse where differences between various groups can be identified. This is of scientific relevance on the one hand to get a deeper understand of particular groups of users, e.g. in the context of gender correctness. An adequate definition and calssification of users groups is of high economic value as well for marketing activities and strategic product design.

All these kinds of analysis can be implemented by implementing one step of analysis in addition to the presented approach. To gain this kind of information, it is basically sufficient to compare the lists of function-requirements of different user groups. Similar to the approach for evaluating the current implementation, each function-requirement may either be *relevant for both groups* or *relevant for only one group*. Assuming that all teh collected function requriments have been structured in categories, as stated above, and the categories are representing certain aspects of the product, is can be clearly seen how important each aspect is for a particular group of users as well as in which areas the biggest differences can be found. An exemplary analysis of the differences between various user groups has been carried out following this approach, based on an empirical study. The details of this investigation are given in II.

## 5.11 Comparison to existing approaches

The proposed approach offers several advantages for product-design when there is an heterogenous user-base. Evaluating the requirements of all users together in such situations may lead to a corruption of results. Usually the largest group of the one which is most comfortable with the implemented requirements engineering process overrules the other minor groups which leads to discrimination of smaller groups as their requirements perish. The distinct treatment of user-groups allows minorities among the user-base a fair participation.

Requirements engineering processes in general have to fit the need of allowing an easy participation of user, according to UCD, but as well a structured representation of the results to allow efficient implementation subsequently as described by function analysis. Nowadays common techniques can be categorized by this properties and analyzed independently, which allows a more clear comparison to the proposed approach.

### User centered approaches

Well known representatives of this kind of approach are user-stories and workshops or discussions. Those techniques aim at an easy participation of users without requiring special skills or knowledge from them. Therefore this techniques are based on natural language and do not restrict the way users state requirements. This avoids falsification of requirements and requirements being not collected as users are not capable of stating them in a complexly structured way. Nevertheless these techniques face the problem of having lots of text in natural language as results which are hard to process later on. Therefore this approaches work well when the development team is capable of keeping the requirements in mind and having an overview of the whole product. They do not scale very well for large products and require an implementation process that is designed to handle this kind of requirements, e.g. modern process models for agile software devvelopment.

The proposed integrated approach allows for the collection of requirements in terms of interviews and focus group discussions, which means there are hardly any restrictions for users. Stating requirements in the format of functions is very similar to natural language, especially as

the level of abstraction is not restricted by this technique. Due to this every user can state all of his needs and it turned out to be helpful to state requirements in this semi-structured manner to avoid dubiety for users and help them to distinct their requirements explicitly.
Compared to other techniques the presented process provides results that are at least semi-structured at a very early stage and become fully strucutured throughout the subsequent steps. This means this process offers the required formalism that is required for later implementation and requirements management while it does not restrict user input at early stages and therefore adheres to UCD principles.

### Structured approaches

Nowadays structured approaches for requirements engineering aim mainly at supporting later phases of development like implementation and maintenance. This processes apply well when requirements are collected by a team of professionals or trained persons but they often provide bad and incomplete results when implemented with users without technical awareness. In this cases the users are overwhelmed by the proces sitself which makes it hard for them to state their requirements efficiently and correctly. This leads to incomplete requirements catalogues and a falsified representation of the real requirements. Furthermore it can cause user-frustration at a very early stage which affects later user involvement, e.g. for prototyping, negatively.

The proposed approach offers all the required flexibility at early stages that is required for efficient user involvement and allows them to state their needs in a most easy way. Nevertheless it keeps the advantages of strucutured approaches in later stages when the requirements have to be implemented and managed.

# Part II

# Evaluation

# CHAPTER 6

# Empirical study

## 6.1 Situation

The empirical study contained in this work has been carried out as a part of the Ge:MMaS [1]research project. The aim of this project was to survey the demands that various users make in laser engraving machines produced by Trotec GmbH, an Austrian manufacturer. Based on an existing implementation of this machines, it should be determined how satisfied certain groups of users are whith this implementation respectively what their specific demands are.

For this purpose the three most relevant user-groups have been identified during the GeMMaS project. As a main focus of the project was put on gender-aspects, the two initial groups split users of the laser engraving machines in male and female users. As an additional distinct subset the group of managers was identified, who, in contrast to the other users, do not work with the machines every day and whose focus was assumed to lie on economic and organizational aspects.

Based on this classification, focus group discussions have been implemented with each of the three groups individually. These discussions were moderated, although the development of the discussion and the addressed topics were widely left to the participants. Each requirement that the group decided to be relevant was stored on a function card as mentioned in I. At the end of the discussion these function cards have been arranged to a matrix under supervision of the moderator. The matrix expressed the importance of the cards topic as well as the degree of current fulfillment and satisfaction. For a more detailled analysis later on, the whole discussions have been recorded on video.

[1] Projecttitle: Genderspezifische Anforderungen für Entwicklung neuer Maschinen unter Berücksichtigung der Mensch Maschine Schnittstelle
Supported by Österreichische Forschungsförderungsgesellschaft (FFG)
Projectnumber: 826182

## Scope

The aim of analysis in the context of GeMMaS was to reveal, which different demands the various groups of users make on the examined laser engraving machines. The results of this investigation should subsequently be used, to allow the manufacturer Trotec GmbH to design a new generation of this machines, which satisfy all relevant groups of users and covers their specific needs. The scientific interest in the result of the analysis for the GeMMaS project is to show, that different gender groups, where gender refers to the social and not the biological gender, make different demands shared products. The differences between the groups should be ananlyzed in terms of the degree of differentiation in general. Furthermore, it should be examined if specific groups differ from others in specific fields of use. Finally, the revealed specific requirements should be compared to the current implementation. This allows shed light if in current requirements engineering processes dominant groups mask the requirements of less dominant groups.

## Structure

The analysis relevant for the GeMMaS project have been carried out partially in the empirical study of this work. In the context of this study the focus was put on a function based approach for requirements engineering. It was analyzed if the proposed approach 5 is capable of revealing individual requirements of specific groups. Furthermore it should be showed that differences between the various user-groups exist in principle. To validate the proposed process, the case study should proof that this process leads to a final list of requirements which can support the further process of development. The analysis are very similar in their implementation in both cases, the scientific questions of the GeMMaS project as well as the experimental application of the proposed approach. Especially the extensive comparison between gender groups, which are relevant for evaluation of the hypothesis of the GeMMaS project, are hardly relevant to evaluate the functionality of the proposed requirements engineering approach. As for the situation of the GeMMaS project comparisons of gender groups have been implemented during this empirical study, which are not especially relevant for gender-focussed analysis and are not required steps of the proposed approach. Nevertheless they provide valuable insights to the opportunities that are offered by this approach and point out its relevance. All analysis of this kind, that have been carried out during this study are marked as such, the rest of this study may be treated as an exemplarily implementation of the proposed approach.

Concluding, the experiences during this exemplarily implementation of the proposed requirements engineering process are discussed and potential weak points of the approach are investigated in detail. In a closing theoretical investigation, the practical relevance and feasibility of the approach is discussed.

## 6.2 Definition of user-groups

As described in section 5.3 an appropriate definition of user groups is crucial for the proposed approach of requirements engineering. In the context of this work user groups have been predefined as a part of the Ge:MMaS project. Nevertheless the underlying thoughts and considertaions of the definition is stated for better understandability of the subsequent process as well as the conlusions.

For the field of laser engraving machines there is one major usergroup which comprises individuals that are directly working and interacting with the machines in their daily routines. For the analysis of requirements of various gender groups, this groups has been further divided in female and male users. Beside individuals that are interacting with the machine directly an additional group of managers has been identified. This is due to the fact that this group of users is affected indirectly by the provided human-machine-interface but especially focussing of capabilities of integrating those machines in producution processes and facility environments. As the needs of the group of managers are widely independet from direct interaction which is supposed to be influenced by gender, a further distinction of managers according to their gender has not been made. This results in three initial user groups, namely *male users*, *female users* and *managers*. Those three initially defined user-groups remained stable for the whole process and hast not been changed during later stages as described in 5.3.

Special considerations where necessary in the context of this work, as all the users that participated in focus group discussions of their corresponding group are customers of the machine supplier Trotec GmbH and therefore may also be competitors in their market segment. The relation of customers and Trotec GmbH as well as relations between competitors made it necessary to allow for anonymous participation in those discussions. To do so, each participant was assigned a fake name, which was used throughout the whole process. Obviously it is not possible to anonymize participants completely when it comes to describing usage scenarios and individual problem situations when interacting with the machines. Nevertheless, anonymization enables participants to speak more freely and leads to revealing deeper insights during discussion. Therefore anonymous participation of users appeared as a useful method for any kind of user centered approaches which requires several users to interact with each other.

## 6.3 Gathering structured Function-Requirements

The first step of analysis of the outcomes from the focus group discussion was based on video material as well as pictures of the resulting pin boards. An example for the function-cards that have been collected during the focus group discussions is given by figure 6.1. The main goal of this phase was capturing the gathered data in a more structured manner that allows for efficient further processing. The result of this procedure is a table listing all the identified function requirements. For a better understanding the structure of this table is described in detail before the process of transferring the actual data from discussion to this table is described. Finally the perceived challenges and advantages of this stage of the proposed process are discussed briefly.

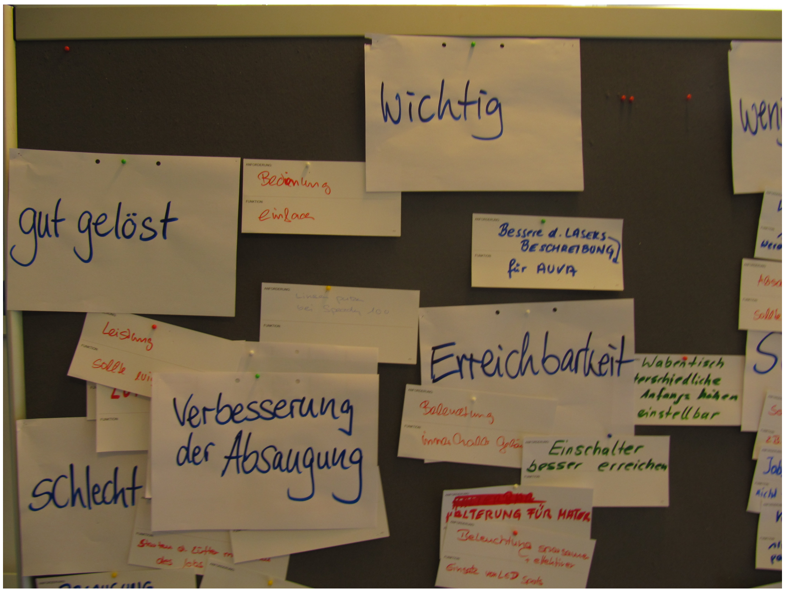
wichtig
gut gelöst
Bedienung
einfach
Bessere d. LASERS-BESCHREIBUNG für AUVA
Erreichbarkeit
Wahrnehmbar unterschiedliche Anfangs haben einstellbar
Verbesserung der Absaugung
schlecht
Einschalter besser erreichen
Beleuchtung

**Figure 6.1:** Function cards collected in focus-group discussions

### Structure

As the output of this phase is a table which represents all the gathered data in a structured manner, the configuration of this table needs to be well defined and several considerations need to be taken into account. At this stage of data structuring it is crucial to preserve the original data as accurate as possible. Therefore the *original formulation* of the function cards that resulted from the focus group discussions are recorded in the resulting chart. To allow an early semantic grouping of this formulations a *category* is present in the table as well.
As described in I requirements can be expressed at several levels of abstraction, namely a *technical level*, a *personal level*, as *function requirement* and as *suggested solution*. Therefore each of those two levels of abstraction needs to be reflected by the table.
When converting original formulations, which often do not cleanly adhere to the definition of functions, to a noun-verb-form it may become necessary to add constraints or comments. Both of them are to be added in seperate columns of the resulting chart. Finally the assigned priority as well as the degree of satisfaction with the current implementation are enlisted, where the values for each function-requirement is defined by the value assigned to its original formulation which it corresponds to.

A template structure for a resulting table of this phase of data structuring, taking into account the mentioned influencing factors, is depicted in table 6.1.

| Original formulation | Category | ID | Abstraction | | | | | | | | Constraint | Comment | Current | Priority |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | technical level | | personal level | | function-requirement | | suggested solution | | | | | |
| | | | verb | noun | verb | noun | verb | noun | verb | noun | | | | |
| Original requirement A | C1 | #1 | change | component 1 | solve | problem 1 | affect | object 1 | implement | solutionX | constraintA | some comment | bad | high |
| Original requirement B | | #2 | improve | handle 2 | reduce | property 2 | influence | object 2 | | | constraintB | other comment | good | low |
| Original requirement C | C2 | #3 | fix | object 3 | handle | object 3 | avoid | situation 3 | | | constraint C | | bad | low |

**Table 6.1:** Template table structure for Gathering function requirements

### Procedure

Once the structure of the chart to be produced is defined, each function card that was filled during focus group discussion was translated to one or more corresponding entries in this table. As a first step to do so, the original formulation of each card was entered to a seperate line of the chart. Additionally, the prioritization as well as degree of implementation were enlisted at this point. Once all the data from function cards was captured, the original formulations were reordered and grouped in semantic categories.

Subsequently each formulation is paraphrased in a noun-verb-form and therefore defined in terms of a structured function. At this point the formulation at the function card need to be assigned to one of the abstraction levels. Usually users tend to formulate their requirements as personal problems rather than in structured funtion-requirements. Therefore the paraphrased requirement needs to be assigned to the corresponding level of abstraction. Once this was done, each requirement that was not defined at the level of function-requirements needed to be transformed to this abstraction level as described in I. When carrying out those transformations it became necessary to state multiple funcion-requirements for a single orginal formulation, which is valid in terms of the described process. Nevertheless the decriptions at different levels of abstraction provided useful information about the context of particular function-requirements. To allow tracking of particular function requirements to map their context at later stages, each line of the table was assigned a unique ID. When assigned the ID it is important to be aware that the requirements of all groups are merged at a later stage, therefore the ID ideally somehow encodes the user-group which this requirement arised from. If the function card contained a suggestion for solutions, the corresponding funtion was added at the proper abstraction level as well.

Implementing this paraphrasing showed to be a time consuming task as it needs quite some feel for language and deliberation to paraphrase requirements without changing their meaning or context. It is crucial at this point to avoid extending or constraining the original formulation. In some cases this process needs more understanding of the context than it is provided by the function card. In this cases it became necessary to investigate the video material to ensure a proper understanding of the intention of function cards. Reflecting this additional context information was done by using the *constraint* or *comment* columns. Furthermore it became necessary in several cases to consult domain experts to ensure proper understanding of domain specific vocabulary.

The resulting tables containing all gathered function requirements of all three user-groups are available on the accompaning data medium.

For an easier understanding a few examples are outlined here to illustrate the results of this steps. The resulting table has been derived from dunction-cards in combination with audio streams of the focus group discussions. The exemplary results are depicted in table 6.2 and are as well contained in the complete evaluation on teh accompaning data medium. As this data is part of an empirical study which was conducted in German language, the examples are given is German language as well to avoid falsification of contents by translation.

| Originale Formluierung | Kategorie | ID | Abstraktion | | | | | | | | Bedingung | Kommentar | Wertung | Priorisierung |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | technische Ebene | | persönliche Ebene | | Anforderung | | Lösungsvorschlag | | | | | |
| | | | Nomen | Verb | Nomen | Verb | Nomen | Verb | Nomen | Verb | | | | |
| Absaugung verbessern (Filter war vor 13 Jahren viel besseres Material) | Emissionen/Wartung | #F20 | Qualität des Filtermaterials | ist minderwertig | Geruchsbelästigung | nimmt zu | Filtermaterial | verbessern | | | | | bad | high |
| | | #F21 | Lebensdauer der Lamellenfilter | ist kurz | Filtertausch | ist oft notwendig | Lebensdauer der Lamellenfilter | erhöhen | | | | | bad | high |
| | | #F22 | | | | | Geruchsbelästigung | vermeiden | | | | | bad | high |

**Table 6.2:** Examples of gathered function-requirements

### Discussion

The main finding of implementing this phase of the proposed process was that defining functions in the verb-noun-form is crucial for transporting the correct information. Therefore this task is very time consuming and it is especially important to be aware of the context and scope of various requirements and function cards. Ideally at least one person that participated in the focus group discussion should be involved in this process to ensure propoer formulations of function-requirements. If this is not possible video material of the focus group discussion is absolutely necessary.

Categorizing function-requirements also needs special attention as in several cases a definitve categorization is hardly possible. Although the assigned categories are not vital for the overall process, proper assignment of categories strongly influences the calrity of the resulting list of function-requirements which can potentially be very large. Proper assignment of categories may make coordination with domain experts necessary.

In very specialized domains, as in the case of this work, domain specific vocabulary may be necessary to understand the requirements and their context. This case also makes discussion with domain experts necessary, as wrong interpretations lead to incorrect function-requirements. This can have serious impact on the whole process and finally lead to erroneous or unintended implementations.

A final finding of this part of the empirical study was, that several function-requirements were added to the list which were not represented by task cards. This is because of the fact that when investigating in the discussiond of all user-groups it may occur that several topics are mentioned in each group but not discussed long enough to be written down on a card. Especially if the common opinions of the groups differ it is reasonable to add function-requiremetns for those tpoics at this early stage.

## 6.4 Consolidating user-groups

Once the function-requirements of each analyzed user groups are gathered in structured tables, it becomes necessary to consolidate the requirements stated by all groups. At this point contradictions in requirements among groups as well as duplicate requirements are handled. The structure of the chart resulting from this step is described in detail before focusing on the procedure of consolidating the three table from the previous step. Subsequently the experiences and findings that showed up during this step are discussed. Finally the process of transforming the resulting chart to a function family tree is described.

### Structure

The goal of this step is a table containing all the function requirements stated in all user groups in a structured manner. This table should especially allow to identify requirements that are common among several user groups respectively differences on certain aspects and requirements between groups. For better readability this table is just containing the abstraction level of function requirements while providing the ID(s) for each of those to enable traceability to corresponding entries in the table from the last phase. Beside the function requirements in verb-noun-format, the constraints and comments for each of the function requirements are stated as well as satisfaction with current implementation and priority for each group separately.
At this point it is important to mention, that several entries from the preceding tables may be merged to one entry in this step if several user-groups stated the same requirement. Due to this fact, the columns for constraints and comments, which are not separated for each group, contain merged comments and constraints. Merged means, that constraints must not be untightened here and comments must not be omitted. The column for IDs contains the IDs of all entries among all user-groups that the corresponding line originated from. This allows for later backtracking to the corresponding entries in case more context information is required.
Taking into account all this consideration, an example structure of the table resulting from this step is given in table 6.5. For the ease of formatting the priorities as well as satisfaction with current implementation are expressed by symbols. In terms of *Current Implementation* a '+' means *good* while a '-' means *bad* and a 'N' means *currently not implemented*. For priority a '+' simply stands for *high* while a '-' stands for *low*.

| | | function requirement | | | | Current | | | Priority | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Category | ID | verb | noun | Constraint | Comment | f | m | b | f | m | b |
| Category 1 | #1 | process | object 1 | constraint 1 | some comment | + | + | + | - | + | - |

**Table 6.3:** Template table structure for consolidating user-groups

### Procedure

For consolidating the requirements among user-groups initially one group was selected which requirements have been transformed to the described table structure. When processing the second group, each requirement was treated separately and checked with the current resulting table to check for a matching entry that is already in this table. If no matching entry could be found it was added as is. If there is a corresponding entry from another group that is represented by exactly the same verb-noun-form the entries have been merged. Performing the merge in particular means adding the additional ID, generating the most tight constraint from both entries as well as concatenating the comments. Finally the corresponding assessment of the current implementation as well as priority needs to be added for the corresponding group. At this point it may happen, that constraints or comments are contradictory although the function-requirements are identically. In this case the entries can not be merged and the new entry is added to the list separately.
When processing big amounts of requirements the process of looking up the resulting table for each requirement to be added can become a very time consuming and complex task. Using the categories that have already been introduced in the previous step allows for splitting up the table which helps to keep track of the overall list and accelerates finding duplicates.

It needs to be underlined, that just looking for entries with the exact same verb-noun-from is not sufficient when consolidating the requirements of various user-groups. If this was possible this task could be completed in a widely automated manner, but in real world situations it happens to be that different groups of users use slightly different vocabulary. This is especially the case as in different discussions the groups face certain aspects of the system from different points of view. This leads to the fact that consolidating the lists of different groups requires semantic understanding and comparison of each function-requirement. For each pair of requirements, that seems to be very similar or even semantically equivalent it needs to be decided, if and how those requirements can be merged. In most cases this means to state a new verb-noun-form for the affected requirements, which again requires some feeling for language as well as sufficient domain knowledge. When merging function-requirements this way it is crucial to ensure afterward, that the new formulation does neither extend the scope or meaning nor limit those for each of the originating function-requirements. If no formulation can be found that fulfills this need in every dimension, the merge of the function-requirements would be invalid as it adds or removes information.

For better understanding of the structure of the resulting data of this step, the example started in refch:gatheringFunctionRequirements is continued in table 6.4.

### Discussion

During the implementation of this procedure it became apparent that keeping track of all the function-requirements and finding all potential duplicates becomes highly complex when there are more than 2 user groups. Nevertheless categorizing requirements semantically eases the look-up for potential duplicates it was often the case that due to different wordings requirements were added to different categories although they were referring to the same things. An accurate processing of each single entry of the originating tables is absolutely crucial for resulting in a

| Kategorie | ID | Anfoderung | | Bedingung | Kommentar | Wertung | | | Priorität | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Nomen | Verb | | | w | m | c | w | m | c |
| Wartung | F20 | Filtermaterial | verbessern | | | - | | - | + | | - |
| Wartung | F21 | Lebensdauer der Lamellenfilter | erhöhen | | | - | | | + | | |
| Emissionen | F22 / F24 / C19 | Geruchsbelästigung | vermeiden | | | - | | | + | | |

**Table 6.4:** Example of consolidated function-requirements

correct list of requirements without redundancies.
The most complicated part although was merging function-requirements that seemed to be semantically equivalent although used different wordings. Finding adequate formulations for the merged requirements without violating or adapting the meaning and intention of the original statement requires deep understanding of the domain and consideration of all context information that is available. As for some requirements from the previous steps it happens to be the case, that they were originally stated at a different abstraction level, they already have undergone one step of rephrasing and potentially experience a further rephrasing at this phase. This fact can lead to dramatic changes in the meaning of requirements overall if the meaning is just changed slightly in each step. To prevent this effect, each merged requirement was compared to its initial formulation, even if on a different level of abstraction, to ensure a correct overall transformation.

Carrying out this consolidation of user groups makes it necessary to find the correct tradeoff for each merge, as being too cautious leads to the fact that there are no merges, which basically means that no information about the commonalities of the user-groups is generated. On the other hand generating commonalities by incorrect merges leads to invalid results. Meeting the correct grade of adapting formulations requires feeling for the language as well as understanding of the domain and the context of function-requirements. Furthermore it was experienced, that some amount of routine is necessary to develop some feeling for when to merge things and when to keep them as they are. In any case it showed to be essential that the persons that implement this step and create the resulting list of requirements need to be the same persons that created the initial lists for each user groups. Otherwise a correct understanding of all the requirements and their context can not be ensured when requirements are represented in this compressed and structured manner.

### Building a function family tree

The list of function-requirements that results from this step basically contains all the information that is required to build a function family tree according to 4.4. As a function family tree was not required for the Ge:MMaS project it has not been created as a part of this empirical study. Nevertheless the information that has been generated by the process so far and the structured information that is contained in the generated table is a viable and sufficient basis for creating a function family tree.
When doing so, the tree should be based on the categorization of the requirements on the one hand and the priority over all groups of users on the other hand. The categories that are assigned in the list allows for splitting the tree at a very high level of abstraction which supports overview and readability. The priorization overall user-groups is the basis for the sequence of implementation that should be expressed by the resulting tree as well. Without implementing the described steps, it is hardly possible to find out the importance of single requirements over all user groups. Building a tree based on this information is essential for a later implementation of a product that is aware of the requirements of all the different user groups and treats them in a fair and rational manner.

## 6.5 Differences between user-groups

Based on the collected data and performed evaluations differences and commonalities between different user groups can be analyzed. Although this step is not essential when applying the described process in a real world project, it may provide viable insights to the interconnection and importance of distinguished user-groups. In terms of real world application this kind of analysis allows deeper insights and understanding of different groups of users and customers. It enables identification of customer segments that are discriminated or privileged by current implementations. This deep of understanding for different groups of users is a potential basis for a more sensible process of product development, handling of requirements and building barrier-free products.

In the scope of this work, beside the insights to different gender groups among the users of Trotec laser engraving machines, this analysis are the basis to proof the initial hypothesis of gender dependent requirements as well as a validation of the proposed approach for requirements engineering.

### Structure

In the course of the GeMMas research project a defined set of categories has been developed for which differences between user-groups should be analyzed. This categories have been the basis for a final requirement profile for the evaluated laser engraving machines. Therefore the discovered requirements are categorized by this defined structure in a first step. Based on this categorization the nominations of requirements in each category are compared by their absolute and relative frequency. A comparison of the frequencies for different categories allows on the one hand to identify main focusses of certain user groups and a comparison of their weightening of various aspects of the products. Furthermore an interpretation of the results enables to state if differences between user-groups are significant at all and if the proposed approach is able to handle those differences in a requirements engineering process. Finally the main facts of the confrontation of the groups results are stated briefly and are depicted by diagrams. A discussion and interpretation of this findings and results is performed in chapter 6.6.

### Categorization for requirement profile

The re-assesed categorization of the identified requirements from preceding steps is specific for the GeMMaS project and became necessary as the final categories to be used have been developed in parallel to the initial evaluation and processing of the data gathered by focus group discussions. The categories that have been identified for the GeMMaS project are listed in table 6.5, a more detailed desciption of teh categories can be found in the final report of the Ge:MMaS research project. Categories identified by a number are referred to as *top-level categories*, while those identified by a number and a letter are subordinated categories referred to as *detailed categories* As the whole GeMMaS project has been carried out in german language, the definition of categories is given in the original german formulation to avoid corruption, nevertheless an english translation is provided as well. Furthermore it has to be mentioned, that questionaires have been evaluated in parallel to the evaluation of the focus group discussions. Aligning the

results of both processes to this defined categories allows for a comparison of the results of both survey-techniques as well.
The structure of the resulting table of this step is the same as stated given by table 6.5 with an additional column for the assigned category. The complete result of this evaluation is available on the accompaning data medium.

| ID | Original | Translation |
|---|---|---|
| 2 | Funktionalität | Functionality |
| 2a | Hauptanwendungen | Main Applications |
| 2b | Leistungsbereich | Power Range |
| 2c | Geschwindigkeit | Speed |
| 2d | Zuverlässigkeit | Reliability |
| 2e | Lebensdauer / Garantiezeit | Durability |
| 2f | Umweltauswirkungen | Environmental Impact |
| 2g | Wiederverwendung von Bauteilen | Reuse of components |
| 2h | Modularer Aufbau | Modular Design |
| 2i | Genauigkeit | Accuracy |
| 2j | Anpassbarkeit | Adaptability |
| 3 | Bedienerfreundlichkeit | Usability |
| 3a | Software | Software |
| 3b | Schnittstellen/Kommunikation | Interfaces/Communication |
| 3c | Servicefreundlichkeit | Serviceability |
| 3d | Reinigung/Absaugung | Cleansing |
| 3e | Erreichbarkeiten/Positionierung | Accessibility |
| 3f | Bedienpanel | Control Panel |
| 3g | Kommunikation/Marketing | Communication/Marketing |
| 3h | Sicherheit am Arbeitsplatz | Safety |
| 4 | Design | Design |
| 4a | optisches Erscheinungsbild | Visual Appearance |
| 4b | Gehäusefarbe | Case-color |
| 4c | Baugrösse | Size |
| 4d | Gewicht | Weight |
| 6 | Zubehör/Optionen | Accessories |
| 7 | Arbeitssystem | Work System |
| 7a | Arbeitsteilung | Dibision of labour |
| 7b | Technischer Anspruch von Werkstoffen | technical sophistication of materials |
| 7c | Mitentscheidungsfähigkeit bei der Bearbeitungsreihenfolge | Decision capability on processing order |
| 7d | Abwechslung bei der Arbeit | Diversity in work |
| 7e | Gutes Gefühl bei der Arbeit | Good feeling at work |
| 7f | Stressiges Gefühl bei der Arbeit | Stressful feeling at work |
| 7g | Anstrengende Situationen | Strenuous situations |
| 7h | Ursache für Fehler/Ausschuss | Cause of error |

| | | |
|---|---|---|
| 7i | Problembewältigung | Problem solving |
| 7j | Materialhandhabung | Material handling |
| 7k | Datenmanagement | Data Management |
| 8 | Anlernen der Systembenutzung | Teaching the system use |
| 8a | Anlernen der Systembenutzung generell | Teaching general use |
| 8b | Erfahrungshintergrund | Experience |
| 8c | Handbuch Anforderungen | Manual requirements |
| 8d | Schulung Anforderungen | Training requirements |
| 8e | Wichtig zum Lernen | Important for learning |
| 8f | Vorraussetzung für die Arbeit | Prerequisite of work |
| 8g | Art der Einschulung | Type of enrollment |
| 8h | Zusammenhang wer wie engeschult wird | Dependecy who is trained how |
| 8i | Learning by Doing/Trial&Error | Learning by Doing |
| 3a | Software | Software |
| 3a.1 | Dialoggestaltung | Dialogue Design |
| 3a.2 | Funktionale Kriterien | Functional criteria |
| 3a.3 | Kriterien Ein/Ausgabe | Input/Output criteria |
| 3a.4 | Dialogtechniken | Dialogue techniques |
| 3a.5 | Darstellung visueller Informationen | Presentation of visual information |
| 3a.6 | Organisation der Informationen | Organzation of information |

**Table 6.5:** Categories for requirements defined for the Ge:MMaS requirement profile

### Counting nominations of categories

Once the categories have been assigned for each requirement stated by the various user-groups, the weightening of certain categories by user-groups was analyzed. To do so, the absolute number of requirements for each category was counted. As different user-groups stated a different number of requirements in total, this absolute number are a valid measure to determine the main focussed categories within a group, although they are inadequate to compare groups. To enable this kind of comparison, the relative frequency of nominations fo each group and category was calculated. This relative numbers allow a direct comparison of a certain category and the weight that is put on this categories by the various groups. Therefore this relative numbers were the main basis for the following interpretations of the results. The evaluated results are stated in table 6.5.

|  | absolute frequency |  |  | relative frequency |  |  |
|---|---|---|---|---|---|---|
| Category | females | males | supervisors | females | males | supervisors |
| 2 | 20 | 8 | 17 | 31,25% | 21,62% | 30,36% |
| 2a | 3 | 3 | 5 | 4,69% | 8,11% | 8,93% |
| 2b | 4 | 0 | 0 | 6,25% | 0,00% | 0,00% |
| 2c | 0 | 0 | 0 | 0,00% | 0,00% | 0,00% |
| 2d | 3 | 0 | 0 | 4,69% | 0,00% | 0,00% |
| 2e | 0 | 0 | 0 | 0,00% | 0,00% | 0,00% |
| 2f | 6 | 3 | 5 | 9,38% | 8,11% | 8,93% |
| 2g | 2 | 0 | 0 | 3,13% | 0,00% | 0,00% |
| 2h | 0 | 1 | 0 | 0,00% | 2,70% | 0,00% |
| 2i | 0 | 0 | 0 | 0,00% | 0,00% | 0,00% |
| 2j | 2 | 1 | 7 | 3,13% | 2,70% | 12,50% |
| 3 | 36 | 18 | 28 | 56,25% | 48,65% | 50,00% |
| 3a | 7 | 8 | 10 | 10,94% | 21,62% | 17,86% |
| 3b | 1 | 0 | 2 | 1,56% | 0,00% | 3,57% |
| 3c | 5 | 0 | 1 | 7,81% | 0,00% | 1,79% |
| 3d | 8 | 0 | 3 | 12,50% | 0,00% | 5,36% |
| 3e | 11 | 10 | 9 | 17,19% | 27,03% | 16,07% |
| 3f | 0 | 0 | 0 | 0,00% | 0,00% | 0,00% |
| 3g | 4 | 0 | 1 | 6,25% | 0,00% | 1,79% |
| 3h | 0 | 0 | 2 | 0,00% | 0,00% | 3,57% |
| 4 | 0 | 2 | 0 | 0,00% | 5,41% | 0,00% |
| 4a | 0 | 0 | 0 | 0,00% | 0,00% | 0,00% |
| 4b | 0 | 0 | 0 | 0,00% | 0,00% | 0,00% |
| 4c | 0 | 2 | 0 | 0,00% | 5,41% | 0,00% |
| 4d | 0 | 0 | 0 | 0,00% | 0,00% | 0,00% |
| 6 | 5 | 0 | 1 | 7,81% | 0,00% | 1,79% |
| 7 | 3 | 9 | 10 | 4,69% | 24,32% | 17,86% |
| 7a | 0 | 0 | 1 | 0,00% | 0,00% | 1,79% |
| 7b | 2 | 2 | 1 | 3,13% | 5,41% | 1,79% |
| 7c | 0 | 1 | 1 | 0,00% | 2,70% | 1,79% |
| 7d | 0 | 0 | 0 | 0,00% | 0,00% | 0,00% |
| 7e | 0 | 1 | 1 | 0,00% | 2,70% | 1,79% |
| 7f | 0 | 0 | 0 | 0,00% | 0,00% | 0,00% |
| 7g | 0 | 0 | 0 | 0,00% | 0,00% | 0,00% |
| 7h | 0 | 0 | 0 | 0,00% | 0,00% | 0,00% |
| 7i | 0 | 1 | 1 | 0,00% | 2,70% | 1,79% |
| 7j | 0 | 0 | 0 | 0,00% | 0,00% | 0,00% |
| 7k | 1 | 4 | 5 | 1,56% | 10,81% | 8,93% |
| 8 | 0 | 0 | 0 | 0,00% | 0,00% | 0,00% |
| 8a | 0 | 0 | 0 | 0,00% | 0,00% | 0,00% |

| | | | | | | |
|---|---|---|---|---|---|---|
| 8b | 0 | 0 | 0 | 0,00% | 0,00% | 0,00% |
| 8c | 0 | 0 | 0 | 0,00% | 0,00% | 0,00% |
| 8d | 0 | 0 | 0 | 0,00% | 0,00% | 0,00% |
| 8e | 0 | 0 | 0 | 0,00% | 0,00% | 0,00% |
| 8f | 0 | 0 | 0 | 0,00% | 0,00% | 0,00% |
| 8g | 0 | 0 | 0 | 0,00% | 0,00% | 0,00% |
| 8h | 0 | 0 | 0 | 0,00% | 0,00% | 0,00% |
| 8i | 0 | 0 | 0 | 0,00% | 0,00% | 0,00% |
| 3a | 7 | 8 | 10 | 10,94% | 21,62% | 17,86% |
| 3a.1 | 5 | 5 | 6 | 7,81% | 13,51% | 10,71% |
| 3a.2 | 1 | 0 | 2 | 1,56% | 0,00% | 3,57% |
| 3a.3 | 1 | 2 | 0 | 1,56% | 5,41% | 0,00% |
| 3a.4 | 0 | 0 | 0 | 0,00% | 0,00% | 0,00% |
| 3a.5 | 0 | 0 | 0 | 0,00% | 0,00% | 0,00% |
| 3a.6 | 0 | 1 | 2 | 0,00% | 2,70% | 3,57% |

**Table 6.6:** Absolute and relative frequency of nominations among user-groups

### Comparison of user-groups

Comparing the results of the various groups to identify differences between groups needs to be done at a proper level of abstraction to preserve readbility. For this reason, the comparison is mainly based on absolute and relative diagrams that are representing the top-level categories, more detailed diagrams for each separate category can be found in A. Nevertheless the interpretation of results given here is sound an not limited to top-level comparison.

The absolute numbers of nominations for each groups are given in 6.2. Female users had the most nominations overall, although most of the overplus was related to category *2 Functionality*. The same holds true for supervisors at a lower rate. The only further difference among groups at this level of comparison is to determine that female users have been less focussed on requirements regarding the *work system* than the other groups have been. Based on this diagram no significant differences between the groups can be identified beside the fact that there have been different absolute counts of nominations to category 2, which nevertheless is the main foucs for each group.

When evaluating the detailed diagrams for the category this impression does not hold true any more. Especially when considering the detailed categories for *usability* 6.3 and *functionality* 6.4, the nominations are distributed differently among the groups, although there total number is roughly the same. Despite the fact that the distribution of nominations is differing between groups (represented by the different shapes of the graphs without respect to their actual amplitudes), a comparison of groups based on this aboslute charts is not adequate and should be based on relative frequencies.

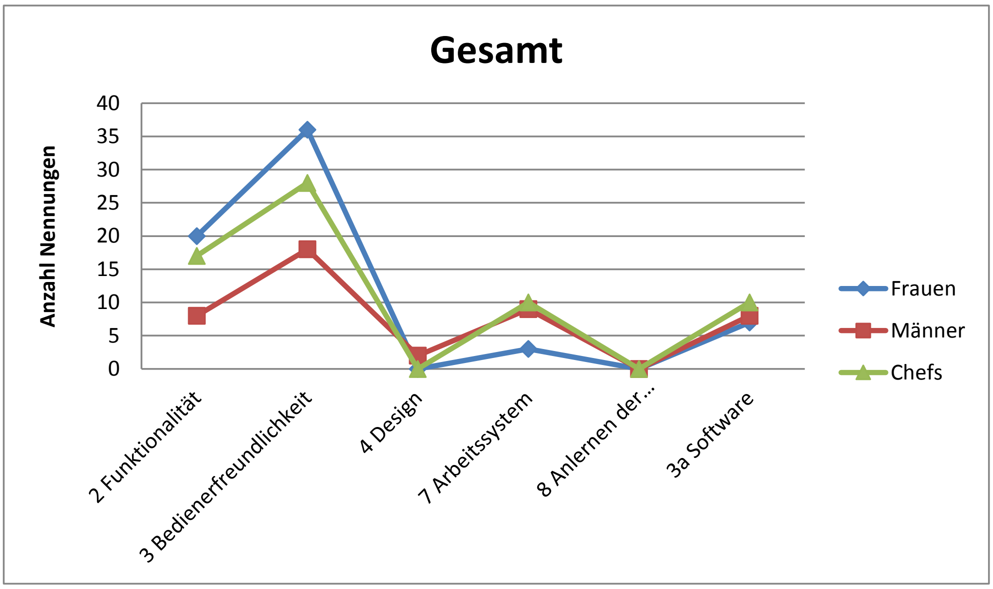

**Figure 6.2:** Absolute frequencies for top-level categories

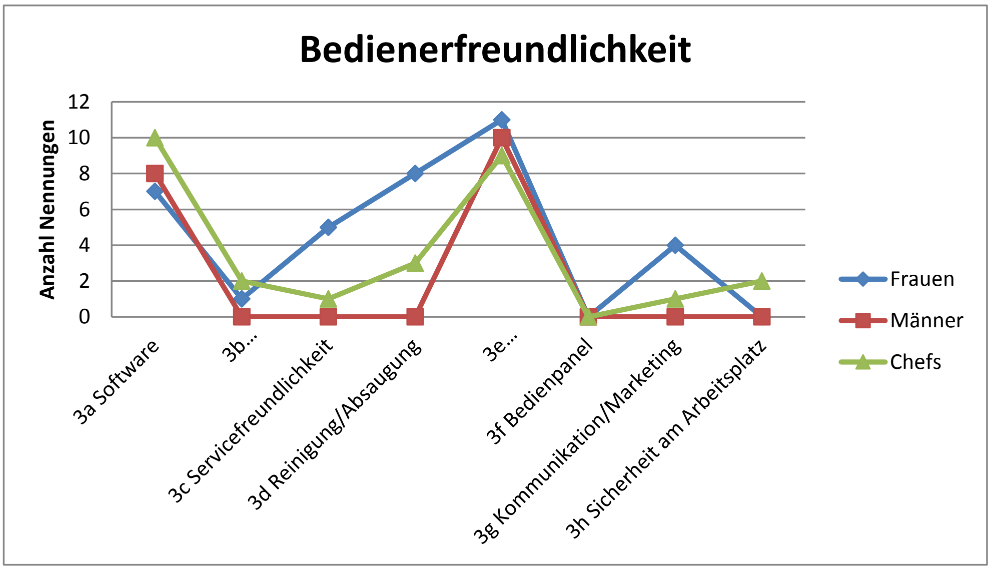

**Figure 6.3:** Absolute frequencies for detail categories on usability

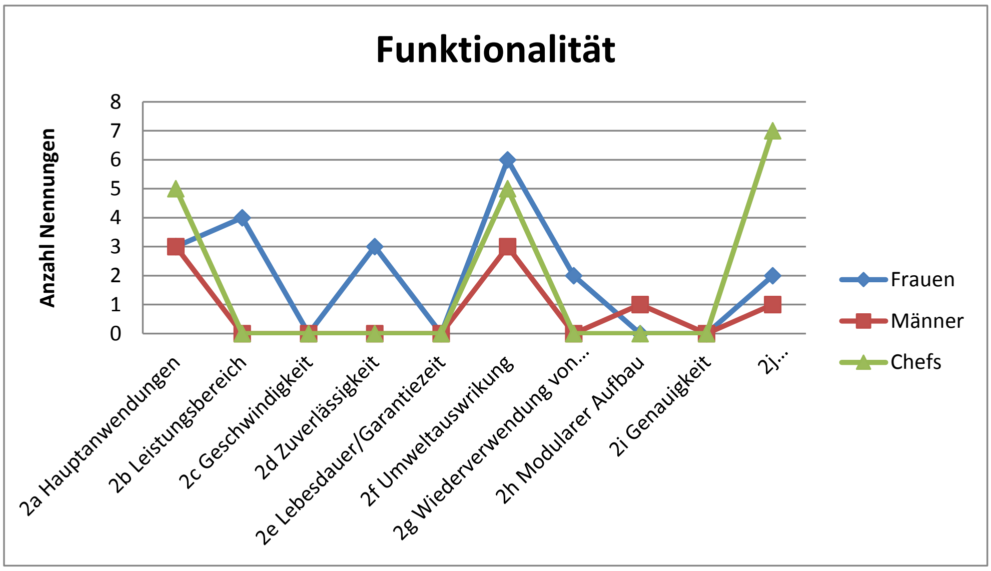

**Figure 6.4:** Absolute frequencies for detail categories on functionality

The relative frequencies for the top-level categories are given in figure 6.5. As already supposed from the analysis of the absolute diagrams when looking at top-level categories the differences between groups are hardly significant. This is likely to be caused by the coarse granularity of this categories. When looking to the detailed categories again it comes clear that there are large differences between groups in certain fields. As given in figure 6.7 supervisors pay by far more attention in *adaptability* than other users do. Furthermore it is interesting to notice that while male users are interested in a *modular design* and *adaptability*, female users pay about the same amount of attention on *durability*. Further significant differences occured in the field of *usability* as depicted in figure 6.6. Here it comes clear that male users are different from the other groups by focussing only on *accessibility* and *software*, while female users and supervisors have similar points of focus which are more equally distributed among all the categories.

## 6.6 Discussion

This section state the results and findings of the empirical study and gives a domain scoped discussion of those. It is discussed if there are significant differences in the requirements of various user-groups and especially for the case of this empirical study how this differences may be gender related. Based on this discovered differences several factors of influence that may have affected the results are stated and discussed and finally the practical implications of this results are outlined.

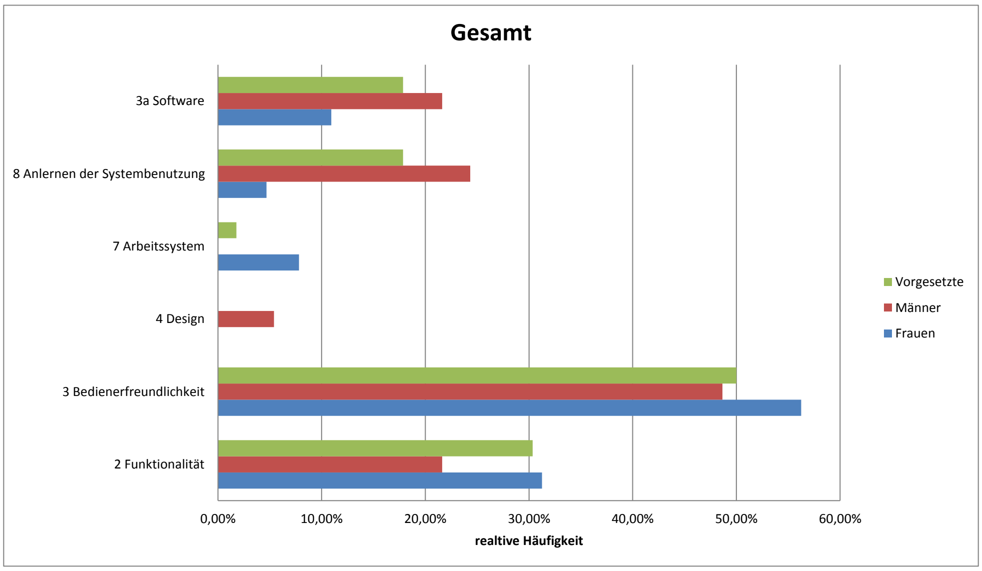

**Figure 6.5:** Relative frequencies for top-level categories

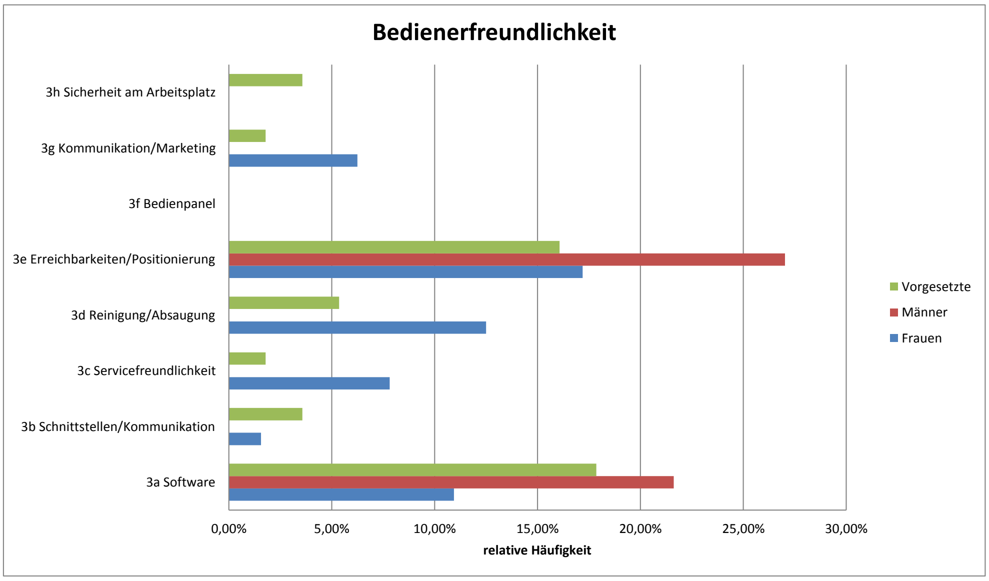

**Figure 6.6:** Relative frequencies for detail categories on usability

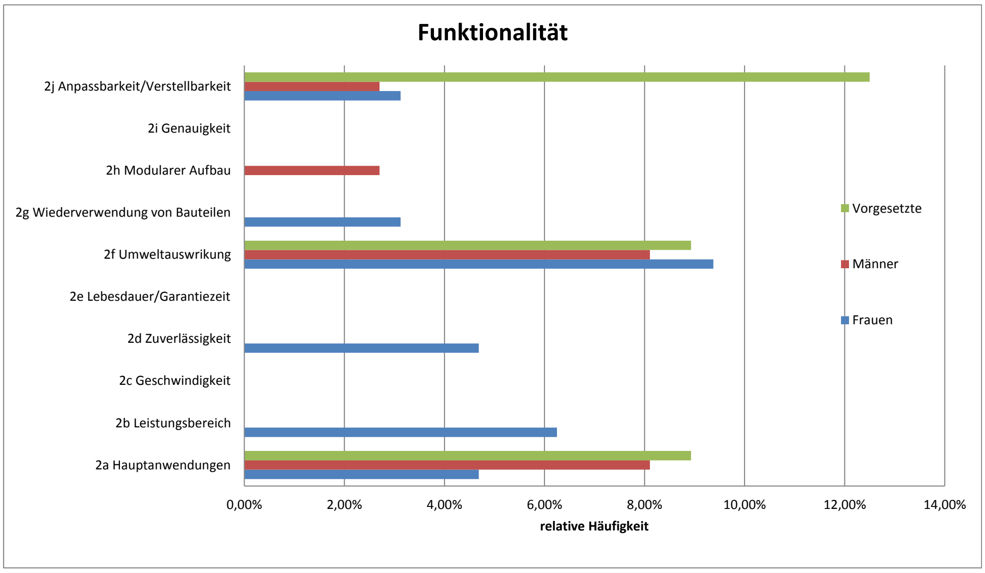

**Figure 6.7:** Relative frequencies for detail categories on functionality

## Differences between user-groups

Looking at the big picture of the results differences between the requirements of the various user-groups can be identified, although it is not obvious if this differences are significant. The global trend among all groups regarding which topics they are focusing is very similar, although the weighting of the focused points differs remarkably. This is expressed by the relative weights that are given to the top-level categories. This is a first strong indicator, that differences between user-groups should be handled in the process of requirements engineering. Beside this different prioritization of categories, much bigger differences between the groups can be observed when evaluating the detailed categories. In terms of the analyzed domain this means that male users attach less importance to the serviceability of the laser engraving machines, whereas this area is very important for female users in terms of cleanliness and maintenance. It was observed that only the group of female users, in contrast to other groups very certain, have a need for reliability of the machines and they claim constant behavior of the machine over time.

### Impact of gender

Based on this widely technical and methodical analysis it can hardly be stated whether this differences are caused by the fact that the groups where distinguished by gender. Nevertheless some interpretation in this context seem appropriate. The first thing to be observed is, that male users have a distinct approach of working than managers, which have been predominantly male in this study as well. The group of male users is the only one paying attention to design considerations, which may eventually be caused by the fact, that they have a stringer personal identification with the machine they use in everyday work.

Regarding the group of managers, it happens to be the case at several points that they differ drastically from the other groups regarding topics like modularization and adaptability. This is likely to be caused by the fact that they have a stronger focus on economical considerations and therefore flexible utilization of the machines. The same holds true for requirements regarding the safety of work.

### Influencing factors

During the evaluation of the empirical study and interpretation of the results, several potential influencing factors have been identified, that may affect the results when implementing a similar study and therefore complicate the interpretation of results. A main influencing factor is the general domain and type of product, which means that a pure retail product that is designed for private customers servers a much larger base of users and therefore there may be more user-groups to be distinguished which are likely to differ even stronger that the user-groups defined in this empirical study. A further factor that influences the results is the dynamics of group discussions. Although the discussion are moderated it may be the case that different user groups put strongly different focuses in their requirements just because of the course of discussion. Considering this it seems reasonable to have multiple discussion for each user group, maybe even with changing individuals, to clean out this factor.

### Practical consequences

There are several considerations for a subsequent phase of product implementation that can be derived from the results of this empirical study. At first it becomes obvious which are the *must-have features* of the product, namely those that are strongly focused by all user-groups. The current implementation seems to have deficiencies in terms of positioning the workpieces as well as regarding the software, especially the dialogue design. Furthermore key features for all user-groups are in the area of usability, environmental implications as well as supporting individual main areas of application. Those points should be implemented and prototyped at an very early phase as they hold the most business value. Once this is done, the specific needs of single user groups should be considered.

If the economic strategy is to focus on special user-groups, e.g. for marketing purposes, the results clearly state the specific needs of each group. This information can be used either for direction of development but as well for separated marketing strategies and advertisements, e.g. emphasizing the flexibility when talking to manager, while putting focus on cleanliness for the group of female users.

In addition the results allow to see if there are any groups strongly discriminated, which seems to be not the case for this case study. An interesting fact that was revealed is that there is hardly any need for easier teaching to new colleagues and the design of the product whereas usability is clearly emphasized by all groups. This acknowledges the need for a user centered approach for requirements engineering and product design.

At some points it was observable that some groups put a stronger focus on missing functionality than others. This may be the case because those groups are not aware of functions that are already available. Further analysis would be necessary to prove if this assumptions is true.

Summing up it can be said that the global trends are widely identical over all user-groups and therefore they define clear priorities for the early phases of product development and implementation. The requirements differ significantly in details, which means once the main, global requirements are fulfilled the further process can be steered in a directed way defining which group should be paid how much attention.

CHAPTER 7

# Conclusion

The initial hypothesis of this work was, that there are differences in the requirements of various user-groups, which are not perceived and handled adequately by current requirements engineering processes. Based on existing approaches for requirements engineering from different domains a theoretical analysis of their strengths and weaknesses was performed. This analysis also took into account how well the requirements engineering approaches could be embedded in larger product development strategies. Based on this theoretical research became apparent that very structured processes exist in the fields of construction and product development stand in contrast to more flexible and user centered approaches especially known in the areas of software development and lean techniques. Furthermore that different processes showed to be complementary in their strengths and weaknesses. In consequence of that fact, a novel approach for requirements engineering was proposed which attempts to combine elements of both areas, considering key elements from User Centered Design, although as soon as possible aiming for structured representation of the collected information which can be fed into a defined and structured process.

The described approach is based on user interviews and group discussions at early phases, which are transformed to a structured process later on that is tightly related to functional analysis. This offers the advantage that the transition from function analysis to a following process, e.g. using Function Family Trees, is already known from literature. This means that it is compatible to current larger processes and strategies for product development that include implementation, logistics etc. Therefore the presented approach is capable of to append to existing processes that currently are based on function analysis approaches.

To prove the validity of the proposed process the data gathered for an empirical study in the context of the Ge:MMaS project was analyzed. The Ge:MMaS project is mainly focusing on the relevance of gender specific requirements, which is a valid classification of user-groups in the scope of this work. Therefore this empirical study is an appropriate scenario for the proposed approach. Based on the data of this empirical study the described novel approach was applied to process the gathered data. On the one hand this was done to prove that there are differences between gender groups in the context of the Ge:MMaS project, on the other hand to prove that the developed process is capable of handling the unstructured data from discussions with various user-groups and to transform the to a combined catalog of requirements.

The evaluation of the empirical study showed, that different groups of users state significantly different requirements to a product. In the scope of Ge:MMaS this means, that an requirements engineering process that purposely perceives different gender groups is reasonable. In the scope of this work, this furthermore shows that the initial hypothesis of different requirements by different user-groups is correct. How significant this differences are depends on the one hand from the domain and on the other hand from the classification of the groups. It can be said, that products that are closer to end-users and serve a larger user-base are very likely to possess much stronger differences in the requirements of their user-groups. The meaningfulness and practical relevance of the proposed approach is therefore taken for granted, at least for specific products.
Furthermore the proposed approach was tested in terms of it's applicability during the evaluation of the empirical study and slightly adapted when necessary. It has been shown that the process as described here is capable of gathering requirements as stated by UCD directly from the user in a widely unstructured format. However, the gathered data can be processed in a way that allows them to be fed into structured processes based on function analysis that are already described in literature and implemented in practice.

In contrast to previous requirements engineering processes this approach offers the advantage, that agility is given where necessary, i.e. at phases that are tightly related to users, whereas it also provides structure where it is needed, i.e. in later stages of the overall process. In addition to this, the proposed method enables a clear and sound tracking of each requirement over the whole life-cycle in both directions, which is the basis for successful requirements management over a products life-cycle. For practical application, at least a partial automation or tool support of the approach is desirable, as the manual effort increases disproportionately with rising complexity of the product and as well the number of classified user-groups. This kind of tool support requires expertise in the field of language processing and software that is capable of building synonyms of functions without adding or loosing any information and meaning.

From today's perspective the proposed approach seems practicable for certain fields of application and products, that intentionally want or need to focus on the needs of different user-groups. This trend is coming more and more in nowadays. In the scope of this work it could be shown that differences in the requirements of various user-groups exist and it could furthermore be shown that the proposed approach is capable of handling this differences and supporting practical relevant product development processes.

# Bibliography

[1] Kaneo Akiyama. *Funktionenanalyse: der Schlüssel zu erfolgreichen Produkten und Dienstleistungen*. Verlag Moderne Industrie, 1994.

[2] Mina Attarha and Nasser Modiri. Focusing on the importance and the role of requirement engineering. In *Interaction Sciences (ICIS), 2011 4th International Conference on*, pages 181–184. IEEE, 2011.

[3] Bainbridge and William Sims, editors. *Berkshire encyclopedia of human-computer interaction*. Berkshire Pub. Group, 2004. (e-book) 0-97430912-5 (hardcopy).

[4] Carliss Y. Baldwin and Eric Von Hippel. Modeling a Paradigm Shift: From Producer Innovation to User and Open Collaborative Innovation. *Social Science Research Network Working Paper Series*, November 2009.

[5] Fernando Belfo. People, organizational and technological dimensions of software requirements specification. *Procedia Technology*, 5:310–318, 2012.

[6] Nigel Bevan. Common industry specification for usability–requirements, 2006.

[7] Elizabeth Bjarnason, Krzysztof Wnuk, and Björn Regnell. A case study on benefits and side-effects of agile practices in large-scale requirements engineering. In *Proceedings of the 1st Workshop on Agile Requirements Engineering*, page 3. ACM, 2011.

[8] Frederick P Brooks Jr. No silver bullet-essence and accidents of software engineering. *IEEE computer*, 20(4):10–19, 1987.

[9] Glenn J Browne and Michael B Rogich. An empirical investigation of user requirements elicitation: Comparing the effectiveness of prompting techniques. *Journal of Management Information Systems*, 17(4):223–250, 2001.

[10] Jim Van Buren and David A. Cook. Experiences in the adoption of requirements engineering technologies. *CrossTalk - The Journal of Defense Software Engineering*, December 1998.

[11] Rafael Capilla, Muhammad Ali Babar, and Oscar Pastor. Quality requirements engineering for systems and software architecting: methods, approaches, and tools. *Requirements Engineering*, 17:255–258, 2012.

[12] Abhijit Chakraborty, Mrinal Kanti Baowaly, Ashraful Arefin, and Ali Newaz Bahar. The role of requirement engineering in software development life cycle. *Journal of Emerging Trends in Computing and Information Sciences*, 3(5):723–729, 2012.

[13] Chee Kai Chua, Kah Fai Leong, and C Chu Sing Lim. *Rapid prototyping: principles and applications*. World Scientific, 2010.

[14] IEEE Computer Society. Software Engineering Standards Committee, Inc. Electronics Engineers, and IEEE-SA Standards Board. *IEEE recommended practice for software requirements specifications: approved 25 June 1998*, volume 830. IEEE, 1998.

[15] IEEE Computer Society. Software Engineering Standards Committee, Inc. Electronics Engineers, and IEEE-SA Standards Board. *IEEE recommended practice for software requirements specifications: approved 25 June 1998*, volume 830. IEEE, 1998.

[16] Alan Cooper and Paul Saffo. *The inmates are running the asylum*, volume 1. Sams, 2004.

[17] Jane Coughlan and Robert D Macredie. Effective communication in requirements elicitation: a comparison of methodologies. *Requirements Engineering*, 7(2):47–60, 2002.

[18] Alan M Davis. *Software requirements: objects, functions, and states*. Prentice-Hall, Inc., 1993.

[19] Gordon B Davis. Strategies for information requirements determination. *IBM systems journal*, 21(1):4–30, 1982.

[20] KD Eason. Ergonomic perspectives on advances in human-computer interaction. *Ergonomics*, 34(6):721–741, 1991.

[21] Nikolaus Franke and Eric von Hippel. Satisfying heterogeneous user needs via innovation toolkits: the case of apache security software. *Research Policy*, 32(7):1199–1215, 2003.

[22] Tilei Gao, Tong Li, Zhongwen Xie, Jiandong Xu, and Ye Qian. A process model of software evolution requirement based on feedback. In *Information Technology, Computer Engineering and Management Sciences (ICM), 2011 International Conference on*, volume 2, pages 171–174. IEEE, 2011.

[23] Paolo Giorgini, Stefano Rizzi, and Maddalena Garzetti. Goal-oriented requirement analysis for data warehouse design. In *Proceedings of the 8th ACM international workshop on Data warehousing and OLAP*, pages 47–56. ACM, 2005.

[24] Ellen Gottesdiener. Requirements by collaboration: getting it right the first time. *Software, IEEE*, 20(2):52–55, 2003.

[25] Jonathan Grudin. Systematic sources of suboptimal interface design in large product development organizations. *Human-Computer Interaction*, 6(2):147–196, 1991.

[26] Paul Gruenbacher. Collaborative requirements negotiation with easywinwin. In *Database and Expert Systems Applications, 2000. Proceedings. 11th International Workshop on*, pages 954–958. IEEE, 2000.

[27] Paul Gruenbacher. Integrating groupware and case capabilities for improving stakeholder involvement in requirements engineering. In *Euromicro Conference, 2000. Proceedings of the 26th*, volume 2, pages 232–239. IEEE, 2000.

[28] Irit Hadar, Pnina Soffer, and Keren Kenzi. The role of domain knowledge in requirements elicitation via interviews: an exploratory study. *Requirements Engineering*, pages 1–17, 2012.

[29] Guozheng He and Jianan Yu. Identify lead users by customer competence. In *Industrial Engineering and Engineering Management (IEEM), 2010 IEEE International Conference on*, pages 1305–1308, 2010.

[30] Juho Heiskari, Marjo Kauppinen, Mikael Runonen, and Tomi Mannisto. Bridging the gap between usability and requirements engineering. In *Requirements Engineering Conference, 2009. RE'09. 17th IEEE International*, pages 303–308. IEEE, 2009.

[31] Cornelius Herstatt and Eric Von Hippel. From experience: Developing new product concepts via the lead user method: A case study in a llow-techffield. *Journal of product innovation management*, 9(3):213–221, 1992.

[32] Elke Hochmüller. Requirements classification as a first step to grasp quality requirements. In *Dubois & al.: Proceedings of the Third International Workshop on Requirements Engineering, REFSQ*, volume 97, 1997.

[33] Hubert F Hofmann and Franz Lehner. Requirements engineering as a success factor in software projects. *IEEE software*, 18(4):58–66, 2001.

[34] Syed Alim Hussain, Arun Kumar Rathore, and Rajesh Singh. Globally accepted requirement elicitation methods. *4D International Journal of management and science*, Vol. 1:52–59, 2012.

[35] IEEE. Ieee guide for developing system requirements specifications. *IEEE Std 1233, 1998 Edition*, pages 1–36, 1998.

[36] ISO. Ergonomie der mensch-system-interaktion – teil 210: Prozess zur gestaltung gebrauchstauglicher interaktiver systeme, 01 2011.

[37] ISO/IEC. *ISO/IEC 9126. Software engineering – Product quality*. ISO/IEC, 2001.

[38] Natalia Juristo, Ana M Moreno, and Andrés Silva. Is the european industry moving toward solving requirements engineering problems? *Software, IEEE*, 19(6):70–77, 2002.

[39] Turkka Keinonen. User-centered design and fundamental need. In *Proceedings of the 5th Nordic conference on Human-computer interaction: building bridges*, pages 211–219. ACM, 2008.

[40] Jenny Kitzinger. Qualitative research. introducing focus groups. *BMJ: British medical journal*, 311(7000):299, 1995.

[41] Henrik Kniberg. *Kanban and Scrum-making the most of both*. Lulu. com, 2010.

[42] Izumi Kohno, Hiroko Yasu, Satoshi Sugawara, and Masahiro Nishikawa. Pragmatic approach to cost benefit analysis of user centered design. In Aaron Marcus, editor, *HCI (9)*, volume 8012 of *Lecture Notes in Computer Science*, pages 525–534. Springer, 2013.

[43] Udai Kumar Kudikyala and Rayford B Vaughn. Software requirement understanding using pathfinder networks: Discovering and evaluating mental models. *Journal of Systems and Software*, 74(1):101–108, 2005.

[44] Sari Kujala and Marjo Kauppinen. Identifying and selecting users for user-centered design. In *Proceedings of the third Nordic conference on Human-computer interaction*, pages 297–303. ACM, 2004.

[45] Sari Kujala and Martti Mäntylä. How effective are user studies? In *People and Computers XIV - Usability or Else!*, pages 61–71. Springer, 2000.

[46] Marianne LaFrance. The knowledge acquisition grid: A method for training knowledge engineers. *International Journal of Man-Machine Studies*, 26(2):245–255, 1987.

[47] Dean Leffingwell. *Agile software requirements: lean requirements practices for teams, programs, and the enterprise*. Addison-Wesley Professional, 2010.

[48] Xiaoshan Li, Zhiming Liu, and Jifeng He. Formal and use-case driven requirement analysis in uml. In *Computer Software and Applications Conference, 2001. COMPSAC 2001. 25th Annual International*, pages 215–224. IEEE, 2001.

[49] Martin Lopez-Nores, Jose J Pazos-Arias, Jorge Garcia-Duque, and Belen Barragans-Martinez. An agile approach to support incremental development of requirements specifications. In *Software Engineering Conference, 2006. Australian*, pages 10–pp. IEEE, 2006.

[50] NAM Maiden and Gordon Rugg. Acre: selecting methods for requirements acquisition. *Software Engineering Journal*, 11(3):183–192, 1996.

[51] Sacha Martin, Aybüke Aurum, Ross Jeffery, and Barbara Paech. Requirements engineering process models in practice. In *7th Australian workshop on requirements engineering. Deakin University, Melbourne, Australia*, pages 41–47, 2002.

[52] Lawrence D Miles. *Techniques of value analysis and engineering*, volume 4. McGraw-hill New York, 1972.

[53] Deepti Mishra, Alok Mishra, and Ali Yazici. Successful requirement elicitation by combining requirement engineering techniques. In *Applications of Digital Information and Web Technologies, 2008. ICADIWT 2008. First International Conference on the*, pages 258–263. IEEE, 2008.

[54] John Money. The concept of gender identity disorder in childhood and adolescence after 39 years. *Journal of Sex & Marital Therapy*, 20(3):163–177, 1994. PMID: 7996589.

[55] Janette W Moody, J Ellis Blanton, and Paul H Cheney. A theoretically grounded approach to assist memory recall during information requirements determination. *Journal of Management Information Systems*, 15(1):79–98, 1998.

[56] Tony Moynihan. Coping with 'requirements-uncertainty': the theories-of-action of experienced is/software project managers. *Journal of Systems and Software*, 53(2):99 – 109, 2000.

[57] Arthur E. Mudge. How to construct and use a ve function chart. *Value Engineering*, 1965.

[58] Donald A Norman. *The psychology of everyday things*. Basic books, 1988.

[59] Donald A. Norman. *The design of everyday things*. Basic Books, September 2002.

[60] Donald A Norman and Stephen W Draper. *User centered system design; new perspectives on human-computer interaction*. L. Erlbaum Associates Inc., 1986.

[61] Dhirendra Pandey, U Suman, and AK Ramani. An effective requirement engineering process model for software development and requirements management. In *Advances in Recent Technologies in Communication and Computing (ARTCom), 2010 International Conference on*, pages 287–291. IEEE, 2010.

[62] U Suman Pandey and AK Ramani. Social-organizational participation difficulties in requirement engineering process-a study. In *National Conference on Emerging Trends in Software Engineering and Information Technology, Gwalior Engineering College, Gwalior*, 2009.

[63] patientmo. Exemplary Wireframe. `http://patientmo.files.wordpress.com/2012/04/compressedandresized1.jpg`. [Online; accessed 2013-09-10].

[64] Mark C Paulk, Charles V Weber, Bill Curtis, and MB (Ed.) CHRISSIS. *The capability maturity model: Guidelines for improving the software process*, volume 441. Addison-wesley Reading, 1995.

[65] S.L. Pfleeger and J.M. Atlee. *Software Engineering: Theory and Practice*. Pearson Prentice Hall, 2006.

[66] Pete Sawyer, Ian Sommerville, and Stephen Viller. Capturing the benefits of requirements engineering. *Software, IEEE*, 16(2):78–85, 1999.

[67] Ken Schwaber and Mike Beedle. *Agile software development with Scrum*, volume 1. Prentice Hall Upper Saddle River, 2002.

[68] Ahmed Seffah and Eduard Metzker. The obstacles and myths of usability and software engineering. *Communications of the ACM*, 47(12):71–76, 2004.

[69] Ben Shneiderman and Shneiderman Ben. *Designing The User Interface: Strategies for Effective Human-Computer Interaction, 4/e (New Edition)*. Pearson Education India, 2003.

[70] Herbert Alexander Simon. *The sciences of the artificial*. MIT press, 1996.

[71] Rashmi Sinha. Persona development for information-rich domains. In *CHI'03 extended abstracts on Human factors in computing systems*, pages 830–831. ACM, 2003.

[72] Carolyn Snyder. *Paper prototyping: The fast and easy way to design and refine user interfaces*. Morgan Kaufmann, 2003.

[73] Alistair Sutcliffe. Requirements rationales: integrating approaches to requirement analysis. In *Proceedings of the 1st conference on Designing interactive systems: processes, practices, methods, & techniques*, pages 33–42. ACM, 1995.

[74] Masatosi Tamai. *Function Analysis*. 1697.

[75] Masatosi Tamai. How to build fufunction family trees. *ValValue Engineering 8 (7)*, 1967.

[76] Masayoshi Tanaka. A survey concerning target ccost, ve and price estimates in the product development and design process. *Japan VE Association*, 1985.

[77] CMMI Product Team. Cmmi for systems engineering/software engineering/integrated product and process development/supplier sourcing, version 1.1, continuous representation. *CMU/SEI*, 2002.

[78] Glen L. Urban and Eric von Hippel. Lead user analyses for the development of new industrial products. *Manage. Sci.*, 34(5):569–582, 1988.

[79] Eric Von Hippel. The dominant role of users in the scientific instrument innovation process. *Research policy*, 5(3):212–239, 1976.

[80] Eric Von Hippel. New product ideas from lead users. *Research Technology Management*, 32(3):24–27, 1989.

[81] Eric Von Hippel. Democratizing innovation: the evolving phenomenon of user innovation. *International Journal of Innovation Science*, 1(1):29–40, 2009.

[82] Karel Vredenburg. Increasing ease of use. *Communications of the ACM*, 42(5):67–71, 1999.

[83] Karel Vredenburg, Ji Y. Mao, Paul W. Smith, and Tom Carey. A survey of user-centered design practice. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves*, CHI '02, pages 471–478, New York, NY, USA, 2002. ACM.

[84] Nigel Whiteley. *Design for society*. Reaktion books, 1993.

[85] Jennifer Wiley. Expertise as mental set: The effects of domain knowledge in creative problem solving. *Memory & cognition*, 26(4):716–730, 1998.

[86] William M Wilson, Linda H Rosenberg, and Lawrence E Hyatt. Automated analysis of requirement specifications. In *Proceedings of the 19th international conference on Software engineering*, pages 161–171. ACM, 1997.

[87] Natalie Woletz. Evaluation eines user-centred design-prozessassessments: empirische untersuchung der qualit&auml; t und gebrauchstauglichkeit im praktischen einsatz. 2006.

[88] Surya Yadav, Ralph Bravoco, Akemi Chatfield, TM Rajkumar, et al. Comparison of analysis techniques for information requirement determination. *Communications of the ACM*, 31(9):1090–1097, 1988.

[89] Dirk Zimmermann and Lennart Grötzbach. A requirement engineering approach to user centered design. In *Human-Computer Interaction. Interaction Design and Usability*, pages 360–369. Springer, 2007.

APPENDIX A

# Comparison of user-groups

The following diagrams depict the differences between the various user-groups that have been evaulated based on data of the empirical study. As the whole empirical study was performed in german language, the captions are in german language as well. Line charts depict the count of function-requirement nominations for the respective categories and user-groups. Bar charts are based on the same data but are representing the count of nominatins in a nomalized manner by using relative weighting.

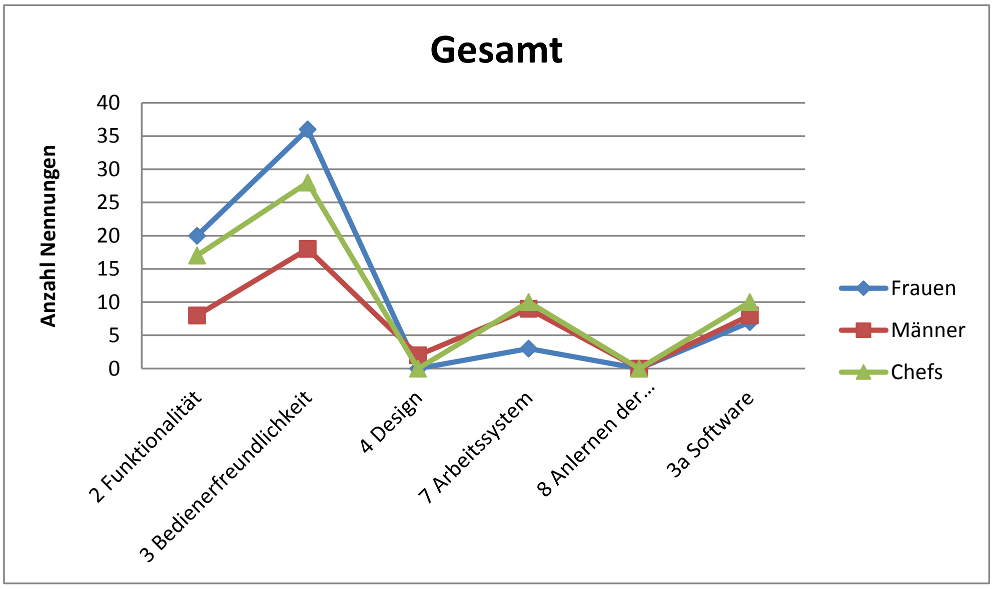

**Figure A.1:** Absolute frequencies for top-level categories

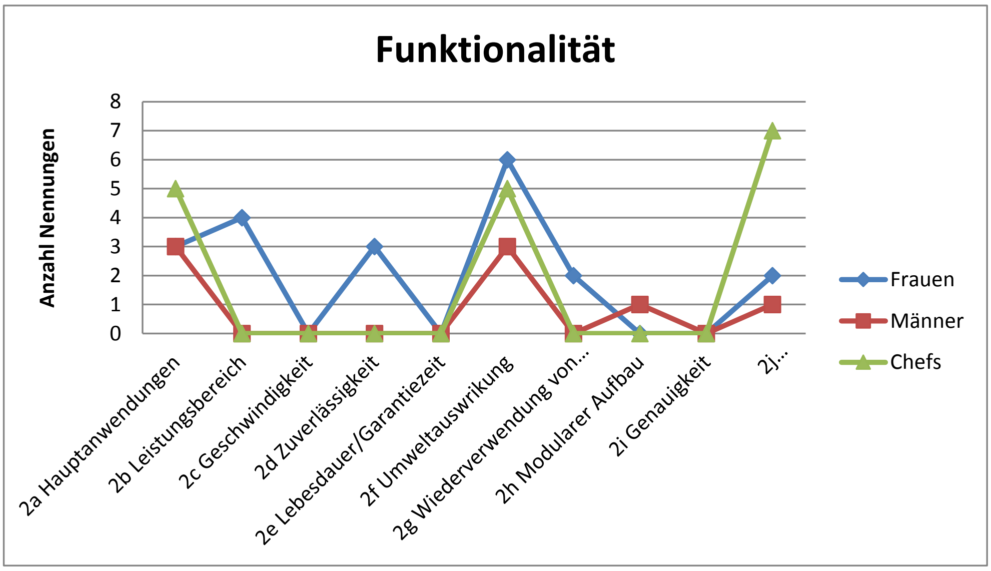

**Figure A.2:** Absolute frequencies for detail categories in functionality

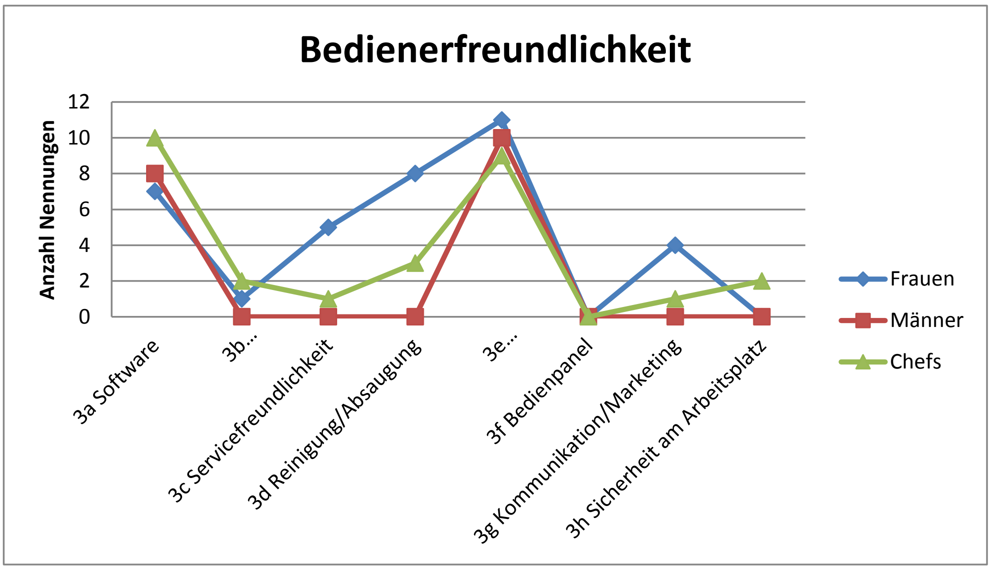

**Figure A.3:** Absolute frequencies for detail categories in usability

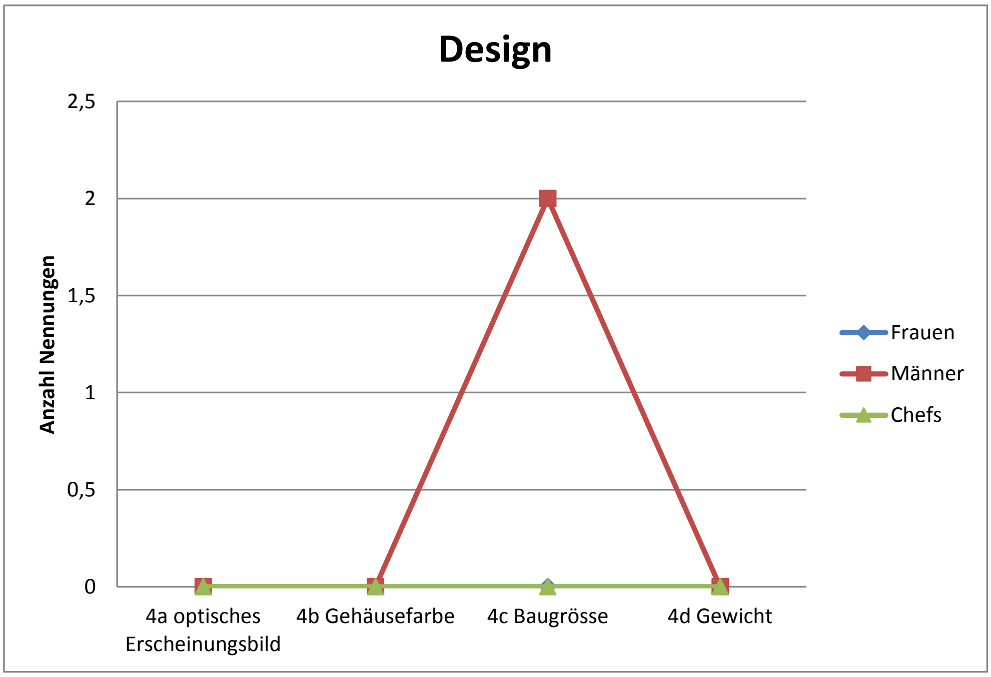

**Figure A.4:** Absolute frequencies for detail categories in design

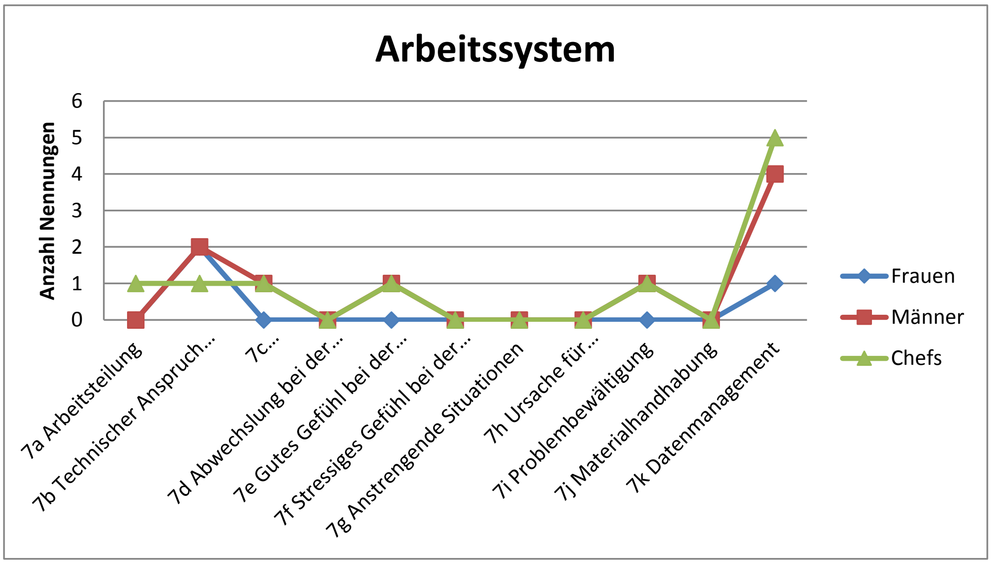

**Figure A.5:** Absolute frequencies for detail categories in working system

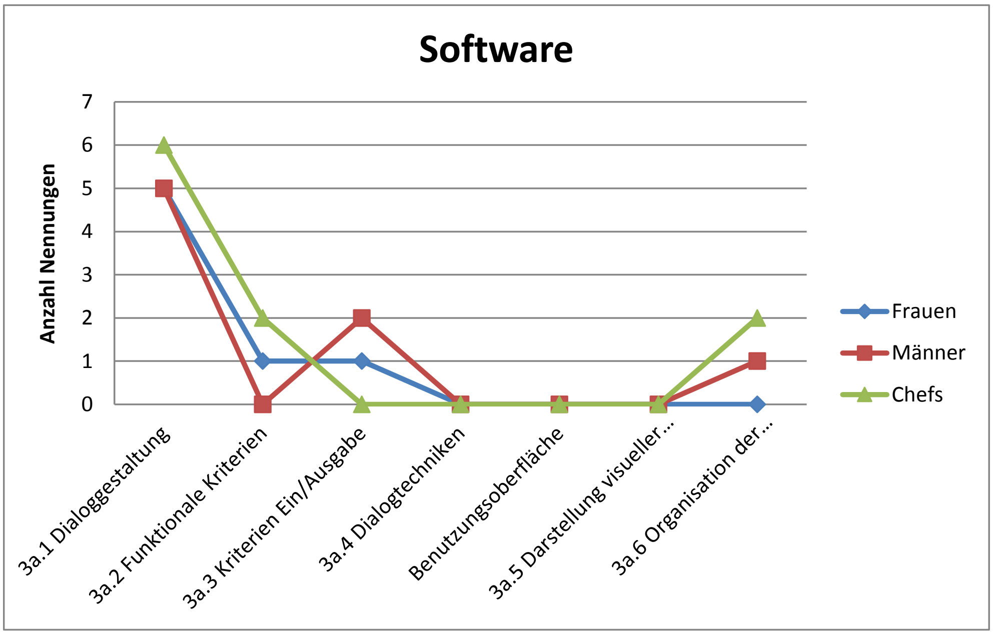

**Figure A.6:** Absolute frequencies for detail categories in software

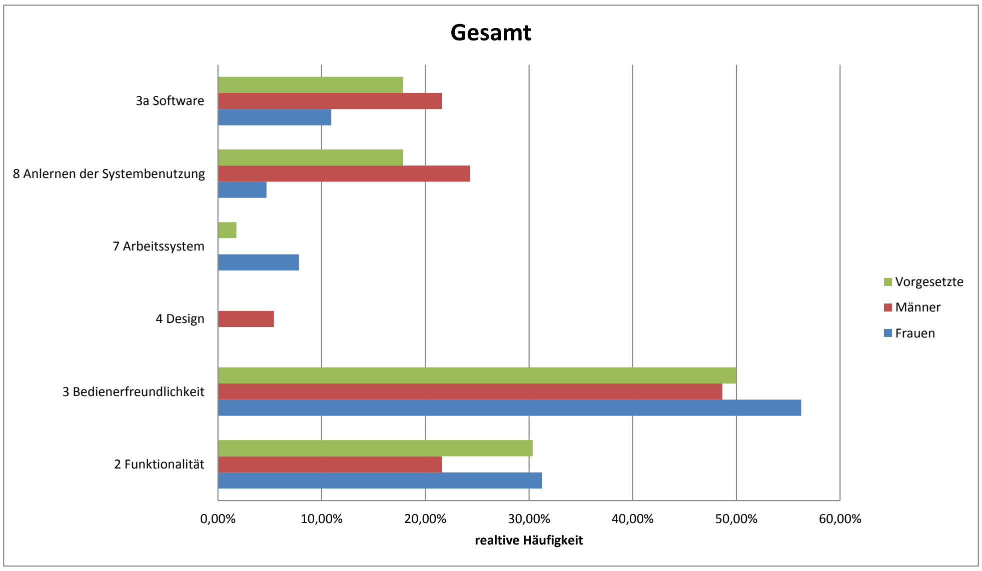

**Figure A.7:** Relative frequencies for top-level categories

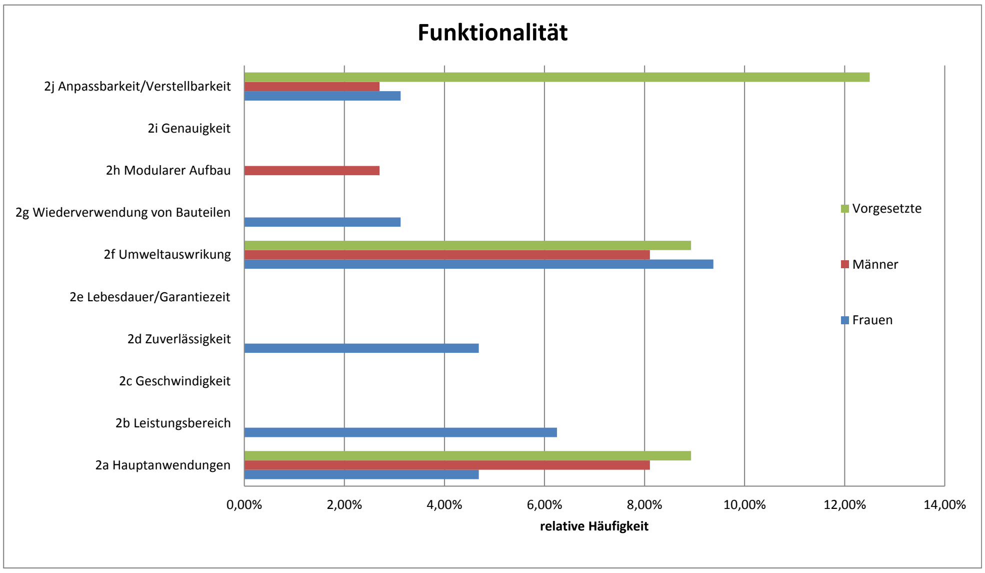

**Figure A.8:** Relative frequencies for detail categories in functionality

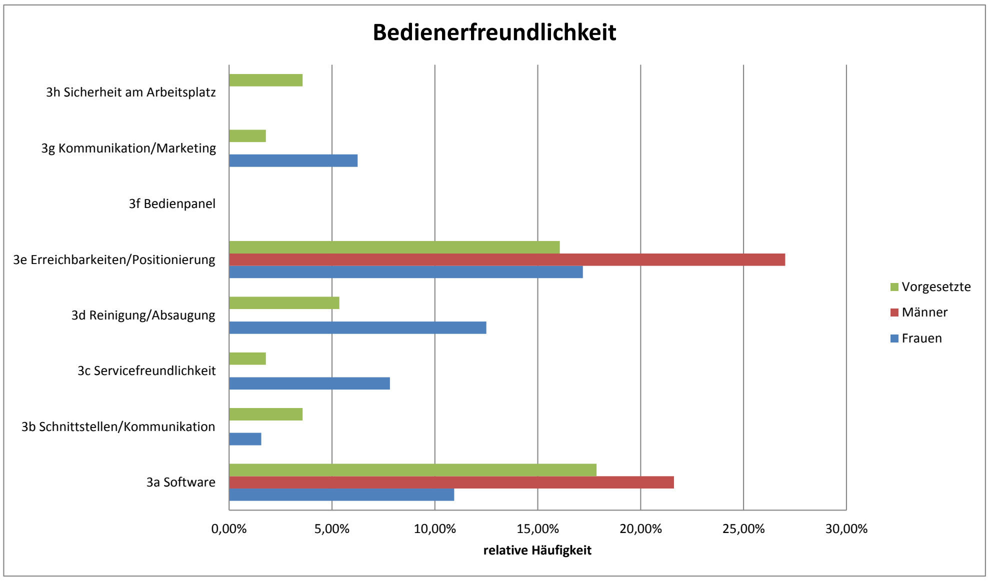

**Figure A.9:** Relative frequencies for detail categories in usability

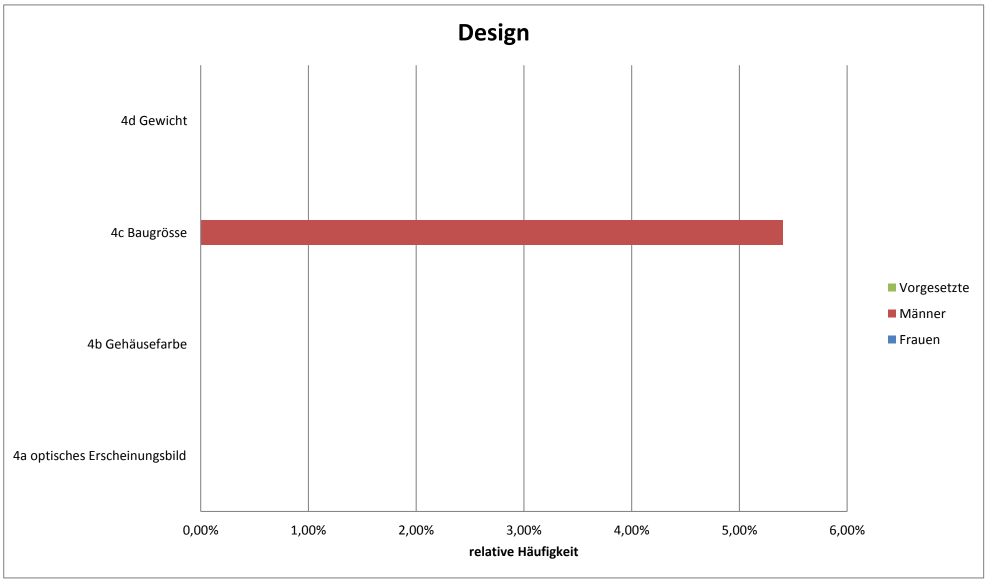

**Figure A.10:** Relative frequencies for detail categories in design

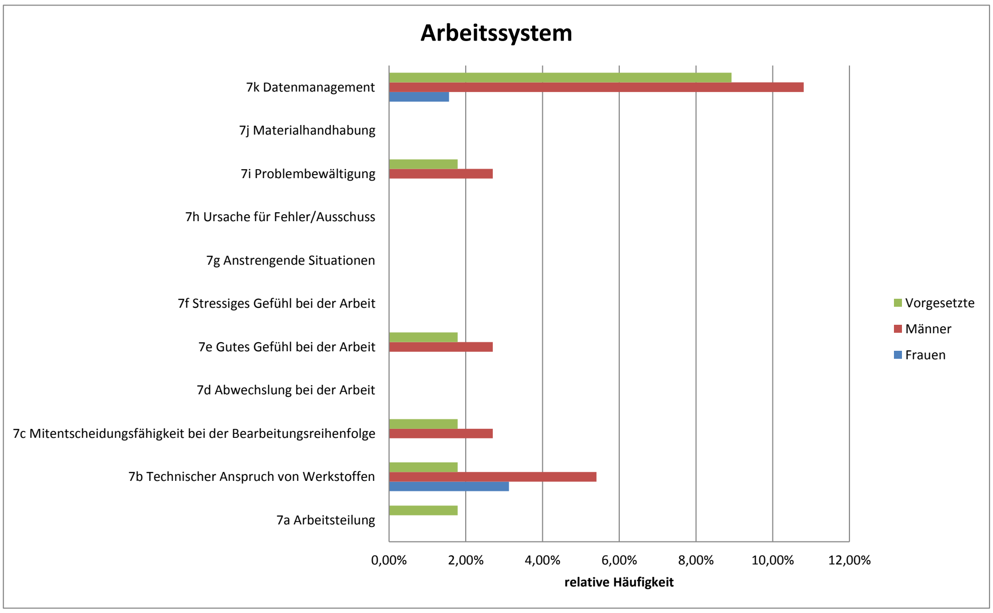

**Figure A.11:** Relative frequencies for detail categories in working system

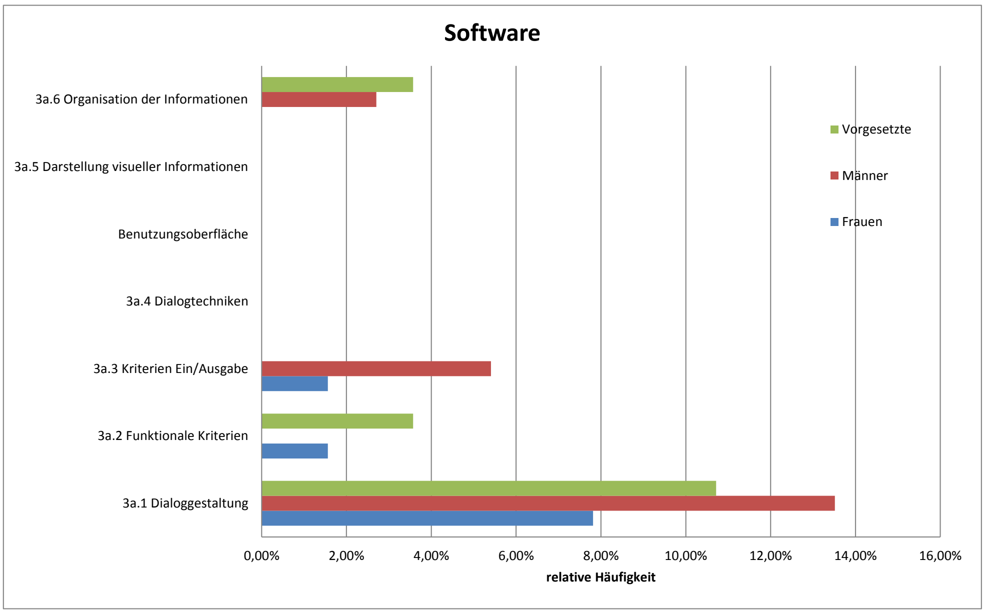

**Figure A.12:** Relative frequencies for detail categories in software