User-Centered Design: Why and How to Put Users First in Software Development

Dieter Wallach and Sebastian C. Scholz

Abstract

In this chapter we provide an overview of the activities and artefacts of the user-centered design (UCD) methodology – a successful and practical approach to the design of software user interfaces. After tracing its foundational principles (early focus on users, empirical measurement using prototypes and iterative design) back to 1985s seminal paper by Gould and Lewis, we will highlight each of five central categories of design activities (Scope, Analyse, Design, Validate and Deliver) performed in UCD. Potential integration of UCD into two popular categorizations of software development (User Interface First vs. User Interface Later) will be explored and then demonstrated in a real life case study from the field of electronic engineering along with a practical takeaway regarding the relationship of UCD and eLearning.

1 Introduction

On January 9th, 2007 Apple announced three things: a widescreen iPod with touch control, a revolutionary mobile phone and a breakthrough internet communication device – all combined in a single device named iPhone. Six months, worldwide distributed photos showing long lines of staying the night customers willing to wait and buy, and massive international iPhone press coverage later, Apple's stock value had already climbed by 60 % when the iPhone finally became available on the market (and ascended by more than 600 % since then). It was not for providing new

D. Wallach
University of Applied Sciences, Kaiserslautern, Germany
e-mail: dieter.wallach@fh-kl.de

S.C. Scholz
ERGOSIGN GmbH, Munich, Germany
e-mail: scholz@ergosign.de

A. Maedche et al. (eds.), *Software for People*, Management for Professionals,
DOI 10.1007/978-3-642-31371-4_2,

functionality that made the iPhone a huge and still on-going success. Quite to the contrary: the iPhone even offered *less* functionality compared to many smart phones of that time. When presenting the iPhone in his now famous keynote, Steve Jobs focused on its revolutionary user interface as the distinguishing factor that was head and shoulders above competing devices: *And we have invented a new technology called multi-touch, which is phenomenal [. . .] It works like magic [. . .] It ignores unintended touches, it's super-smart* (Jobs 2007).

Performing as an external design partner for a variety of types of clients, industries, projects and artefacts over more than a decade we are repeatedly exposed to several fallacies regarding interface design - the first one starting with the notion that a high prioritization of the user interface is apparently something new and ground breaking. Although the above-mentioned iPhone is certainly the most often cited example, it was not Steve Jobs who invented good user interface design to support and delight users in their daily life and business. The understanding of good user interface design and methods for achieving high usability of interactive devices can at least be traced back to a seminal paper by Gould and Lewis (1985) that we will discuss in the second section of this chapter.

Another common fallacy is the equation of interface design with some form of visually bedaubing the screen – an activity that can be added later to an otherwise finished product. The iPhone itself is a perfect example for this assumption's fallaciousness with its *interaction design* being the primary innovative achievement. For a significant and sustainable impact a variety of interface design activities (covered in Sect. 4) needs to be conducted and deeply engrained throughout the complete development process (see Sect. 3).

Finally, a third fallacy – that interface design is subjective, neither measurable nor objectively discussable – is often heard. Though certain aspects such as the visual design of an interface bearing potential for more discussion than others, user interface design is a process based on gathering information relevant for informed decisions. By allocation of dedicated scoping and analysing activities upfront the actual design very limited room is left for bias and personal agenda (discussed in Sect. 4).

2 Key Principles

2.1 Twenty-Seven Years Later: Gould and Lewis (1985) Revisited

To prepare for subsequent sections of this chapter, a major potential terminological pitfall needs to be addressed at first. *Design* as a concept deals with the challenge of being used as a label for a single *activity* (e.g. icon design), a *process* consisting of multiple activities (e.g. user-centered design), a *deliverable* or *result* (e.g. design style guide) and as *field* (e.g. user interface design). This already bears much potential for confusion in establishing a common ground between stakeholders for example a board, designers and developers. Yet it even gets more complicated: These days *user experience (UX) design*, *user interface (UI) design, graphical user*

interface (GUI) design, user-centered design (UCD), interaction design (IxD), user interface developer (UIDev) and additional terms and acronyms are used interchangeably – and frequently even with different connotations – leading to additional confusion. In this chapter we will refrain from using these acronyms and will refer to "design"/"designers" as a comprehensive hull for the respective nuance variants mentioned. Instead we will concentrate on a depiction of user-centered design as a process, opposed to the more artefact driven UI/GUI/IxD/UIDev connotation or the all embracing and thus very vague and result driven UX design.

Even though 27 years feel to be close to eternity in Information Technology, a still required read to understand the essence of user centered design is Gould and Lewis' (1985) paper *Designing for Usability: Key Principles and What Designers Think* in which the authors present and discuss three principles for the emerging field: (1) early focus on users, (2) empirical measurement using prototypes and (3) iterative design. Gould and Lewis explicitly differentiate between *understanding potential users*, *versus identifying, describing, stereotyping and ascertaining them*. Herefor the authors stress the importance of bringing designers into direct contact with potential users and mention interviews or exchanges with and observations of users as appropriate methods to be applied prior to design. Knowledge about a user's tasks can then be aligned with the knowledge about users themselves (in terms of cognition, behavioural working conditions, literacy level and exposure to previous systems). Gould and Lewis also emphasize (p. 302) the need for testing prototypes very early in the development process – an argument that we will pick up in Sect. 4.3.1.2 of this chapter. Their reference to testing is further substantiated by the use of empirical evaluation methods where *intended users should actually use simulations and prototypes to carry out real work, and their performance and reactions should be observed, recorded, and analysed.* (p. 300).

Interestingly enough the authors already make a distinction between the creation of a functional prototype to test the feasibility or stability of technical properties of a system and a non- or semi-functional prototype following the main goal of studying a user's reaction to it: *What is required is a usability test, not a selling job*. (Gould and Lewis 1985, p. 302).

Gould and Lewis strike a crucial chord regarding the necessity to separate the roles of designers and developers for a simple reason: It is cognitively impossible for developers (and other project stakeholders) to pretend to be a novice user. An insightful demonstration to justify this separation can be used to illustrate their argument. Humans often have a strong believe in their perspective-taking abilities and especially developers are fast in replying that they themselves are users – so why would they not be able to put themselves into the shoes of a user? (Mayhew 1999). Asking participants whether they are able to identify a meaningful pattern or object when exposed to the picture shown in Fig. 1 usually leaves them in a short moment of silence.

After a while, a cue to seeing the head of a cow in this picture typically emerges in some form of light bulb moment in which participants confirm that they *now* see the cow previously unheralded. If then instructed to ignore the cow, participants are

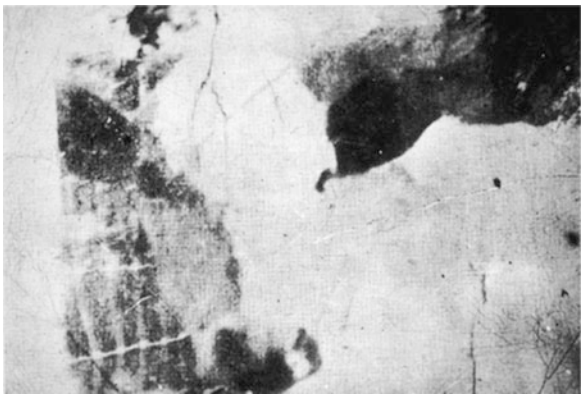

Fig. 1 Demonstrating the need for involving users

quick to point out that this impossible now. Like Heraclitus' *You could not step twice into the same river* having seen the cow once does hardly allow to switch back to the previous unknowledgeable state. In analogy to this demonstration, it is just as hard for developers, product managers, business analysts and also for designers having all the knowledge that professional life entails to go back to the unknowledgeable state attributed to a novice user. To avoid this fallacy, Gould and Lewis (1985, p. 302) recommend that *potential users become part of the design team from the very outset when their perspectives can have the most influence, rather than using them post hoc as part of an analysis team (of) end user representatives*. Isolating designers from users de facto eliminates the empathic knowledge about users as the target of design.

The final major takeaway from Gould and Lewis (1985) is their emphasis on the importance of iteration. The authors are very clear in accenting the necessity to setup a process that enables cycles of design, test, measuring (behavioural goals) and redesign frequently (p. 300) – now being a hallmark of user-centered design as discussed in section three of this chapter.

2.2 Usability

Gould and Lewis base their résumé on key principles of designing for usability on three very solid pillars: *user-centricity*, *empirical measurement* and *iteration*. Achieving usability then presupposes the fulfilment of the following conditions (1985, p. 307):

- *A description of the intended users must be given.*
- *The tasks to be performed, and the circumstances in which they should be performed, must be given.*
- *The measurements of interest, such as learning time, errors, number of requests for help, or attitude, and the criterion values to be achieved for each, must be given.*

While Gould and Lewis themselves do not provide an explicit definition of usability in their paper, the aforementioned conditions are in close correspondence to the widespread ISO EN 9241-11 definition of usability. The ISO EN 9241-11 defines usability as the *Extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use*. The metrics mentioned in the ISO definition (efficiency, effectiveness, satisfaction) can be mapped to the last condition claimed by Gould and Lewis that we will come across again in Sect. 4.4.2 when discussing the concept of *usability goals*. The mentioning of a specified user working in a specified context of use to achieve specified goals in the ISO definition have their counterparts in Gould and Lewis' description of intended users, given tasks and the circumstances in which they should be performed.

It is without doubt that Gould and Lewis laid the foundations of user-centered design by providing the key concepts on which current approaches for developing usable interactive systems are still built. It was also Gould and Lewis who came up with the memorable and often heard but rarely correctly attributed leading record that from a user's perspective *with computer systems the product is the user interface* (1985, p. 306).

Creating a truly usable interface to provide access to the functionality of an application calls for an interdisciplinary team to arrive at a comprehensive understanding of the *cognitive, behavioural, anthropometric, and attitudinal characteristics* (Gould and Lewis 1985, p. 300) of users that are to be supported by *studying the nature of the work expected to be accomplished* (Gould and Lewis 1985, p. 300). In the next section we will briefly discuss different approaches to software development in their relation to design before then providing a more detailed overview about user-centered design by differentiating its underlying core phases.

3 Development Approaches and Design

We have already adverted to the multiplicity of meanings of the term *Design*. In the context of development models the term *Design* comes with at least two different semantic references: Software Design (i.e. designing the internal structure of an application, *the aspects of the software that affect functionality and execution performance*, Biddle et al. 2007) and the design of a user interface (following the semantic scope discussed in the previous paragraph). Depending on the respective approach followed, resources devoted to the design of a user interface are concentrated at different stages of a project. For the sake of clarification we will refer to a dichotomy of *User Interface first* (UIF) versus *User Interface later* (UIL) stances to distinguish different process models of development, albeit acknowledging that actual project realities often proceed on the continuum between these semantic oppositions.

A corollary of UIF is that an entire user interface – providing interactional access to the full functionality of an application – can be completely specified before a

single line of code is written. It has to be noted though, that comparable to the floor plan of an architect that presupposes a comprehensive understanding of material properties and structural analysis, user interface design (while preceding programming like floor plans precede actual building activities) requires a thorough appreciation of technical capabilities.

UIF approaches typically have a broad understanding of the concept of a user interface that is not restricted to providing a visually appealing and aesthetically pleasing wrapping of functionality and enabling efficient interaction with it. In lieu of successful interface creation, preparatory steps are required to identify the right functionality (i.e. to select the appropriate feature set) for supporting a user in achieving her working goal in a given environment. An underlying assumption in such a user-centered approach is that the cost of initially planning, validating and specifying the user interface is less than the cost of fixing implementation code due to change requests – not speaking about maintaining revised code. Due to its high amount of (formal) documentation UIF, also mitigates the risk of losing knowledge if resources leave an organization.

While the traditional waterfall model of development exemplifies a steady approach of complete specification of software before implementation, its explicit reference to a Design stage between does not necessarily imply being UIF. On the contrary: projects following the waterfall approach frequently leave the details of interface design decisions to later stages, focusing instead on the specifics of the functionality and the internal structure of the software to be engineered.

The agile approach, which can be more adequately characterized by the UIL acronym, tries to leverage small self-organizing teams working in short time boxes without long-term planning for the user interface dominating. Tasks and components are commonly broken down into sprints realizable in short periods even down to 1-week, each contributing to a successively feature-complete and sprint-wise working implementation. While agile projects might start with some initial vision of an application's user interface in an imaginary sprint zero, this clearly is not covered by UIF. The key philosophy in agile projects is the emphasis on quick adaptability with low formality thus minimal documentation. Criticizing UIF for an insufficient provision for changing requirements, the cost of overall planning is assumed to be higher than the cost of fixing.

The debate between representatives of both characterizations is a very emotional and intense one (especially with designers and developers advocating the respective oppositions involved). This is at least surprising, given that agile methodologies also refer to activities typically related to user-centered design (*Scope, Analyse, Design, Validate* and *Deliver*, outlined in Sect. 4) and both are driven by a strong interdisciplinary collaboration of different roles. Very abstractly speaking, from an interface design perspective both approaches share striking conceptual similarities, differing only with regard to granularity (UIF: analysis-design-validate cycles for the features embodied in a user interface, preceding development; agile UIL: many short design sprints, each devoted to selected feature sets, overlapping development) and maturity of deliverables (UIF: detailed top-down specification of core functionality, extensive prototyping; agile UIL: minimal bottom-up documentation of selected functionality, focus on productive (sub-)systems).

Most importantly, agile development and user-centered design are both fuelled by being highly iterative: The creation of an artefact that can be validated and refined based on feedback gathered. The shared goal is mainly deviating in the nature of the artefact: In UIF validation is usually done with lightweight prototypes; in UIL user validation is typically done with productive code. Its reliance on productive code is also the most fundamental challenge for UIL: If not carefully monitored, the risk of implicitly adjusting the design proposal to development capabilities is very high, diminishing the chance for real innovation.

The question when to follow UIF or UIL cannot be answered dogmatically. It should rather be seen as a tendency not as an either/or decision. The *placement* of design in the UIF/L continuum depends on a project's scope and setup of contributing parties. In the situation where development is outsourced, UIF makes a strong case because its output (e.g. a fully specified user interface, potentially accompanied by an illustrative interactive prototype) provides a sound base for a bid to potential contracting developers. Yet if development is done completely in-house its detailed documentation is less needed since developers can participate from start and feasibility concerns can be raised very early.

The major factor shaping the embodiment of design in a project is the question whether an evolution or a revolution of a product is targeted. If existing functionality is to be redesigned or enhanced, design-related activities are tied to the respective development specifics much more tightly. However, when more extensive new functionality is to be introduced in a product or when even an entire new application is being planned, the creation and validation of a greenfield before development is advisable. In the context of web applications, for example, a paradigm of continuous improvement is often practiced, explaining the popularity for UIL since the traditional definition of (yearly) releases is hardly applicable in this case. In the field of productivity tools for expert users, however, comparatively long release cycles (e.g.one a year) allowing unleashing the full potentials of UIF.

4 User-Centered Design Activities

Though carrying a plethora of different labels, which greatly vary through literature and agency descriptions, the activities performed in a typical user-centered design project can be assigned to the following five categories: *Scope*, *Analyse*, *Design*, *Validate* and *Deliver*. A structured iterative process arranged by the information needs of the corresponding phases flexibly links especially the three center categories. To perform the activities in each phase, designers base their decisions on theoretical insights and have access to practical methodologies, which – contrary to (still) popular belief – do not necessarily presuppose the *ingenious creation* of subjective art. These are rooted in the contributing disciplines of human-computer interaction that were adapted and refined to serve the goal of designing an effective, efficient and satisfying interaction for achieving a user's goal.

As outlined in the previous section, both development approaches (UIF and UIL) significantly differ in the timing of a design stage as well as in the granularity and the maturity of its deliverables. We will thus explain the activities of each phase in

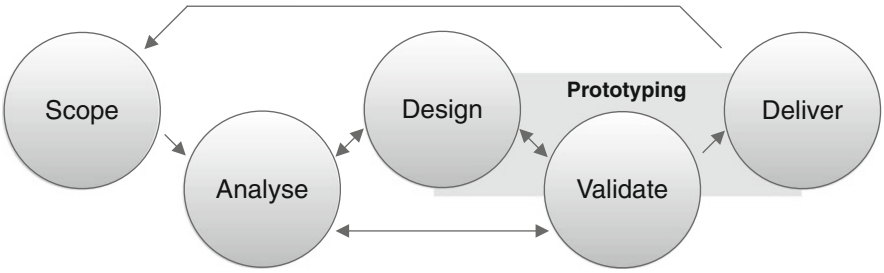

Fig. 2 User-centered design activities

their typical iterative succession – highlighting only in brief the most important supporting methods, instruments and artefacts in each phase to paint a comprehensive picture of the overall process (Fig. 2).

Deviating from common practice in usability textbooks - where methods of usability evaluation are subsumed under the umbrella of the validation phase - we are also addressing these already in our description of activities in the Analyse phase. This portrayal reflects the course of many real life projects that base their agenda on the grounds of a detailed usability inspection of the current application to arrive at a specific working goal for the upcoming application. Depending on the lapse of a project and the corresponding information needs, the corresponding activities might be applied in both, the Analyse and the Validate category.

4.1 Scope

The most crucial aspect of a user-centered design project is to initially establish a common ground (Clark and Brennan 1991) between its contributing stakeholders. This is even more challenging in the dominant situation for small/medium enterprises, where major parts or even the complete design process is often outsourced to a consulting agency with a binding offer to be delivered upfront.

Combining the product vision and the research results of a product owner's input (comprising the request for proposals, potential demos, available manuals, illustrative sketches, documented user feedback, etc.) and the output of an interdisciplinary scoping workshop (including board, product managers, developers, management, etc.), a concrete mission statement of a project should be derived during scoping that addresses two major aspects: the goals and constraints of the project. For most projects, relevant goals and constraints can typically not be established in comprehensive detail at this stage but need to be identified and discussed at a qualitative level to set the agenda for the analysis stage.

4.1.1 Goals

Since any meaningful interface exists to serve a specific purpose, the functional goals for a design project are to be addressed upfront in the light of sufficient

domain knowledge. These goals usually shift between two extremes on a continuum: from a purely visual redesign solely of existing functionality over a design for new functionality (extending or replacing current functionality) to a creation of a completely new application in a greenfield approach. Any tendency in between has to be put into perspective during scoping with the business goals of the project: Shall the new interface attract and convert new customers (in yet unclaimed markets or contexts) or will its unique selling proposition promise to yield a higher return on invest for existing users due to increased productivity, lower learning effort, higher joy of use or other goals related to product usability? The outcome of this discussion heavily plays into planning the focus of each phase in the process, especially during analysis and design.

Often the need for setting up specific interface design projects is extrinsically motivated through inquiries raised by a product's current users or by insights gathered from competing applications. Hence it is not uncommon that specific expectations regarding the usability of an artefact's new interface are already raised at this state, often in a less than organized way. It is recommendable to explicitly clarify and match these goals to defined usability metrics such as performance, learnability, error avoidance, self descriptiveness, memorability or consistency (for a full discussion of usability metrics see Tullis and Albert 2008). A stable framework has to be derived for setting up target and acceptance values for these predominantly business goals in the analysis phase and to evaluate the successful achievement of these non-functional requirements in the validation phase.

4.1.2 Constraints

Art lives from constraints and dies from freedom da Vinci once stated, which in its essence directly applies to interface design as well: Without establishing clear borders of the solution spectrum in the design space (Simon 1969), there is a latent risk of overdesigning solutions for which feasibility cannot be guaranteed. The strongest constraints are generally of technical nature; therefore it is mandatory to include development representatives right from the start in the project team. Seemingly smart design suggestions that cannot be transformed to real products within technical constraints in given time and budget can thus be quickly identified and solution spaces adjusted.

Besides technological constraints, the contextual environment of a user working with a (future) application defines another set of constraints that needs to be analysed and understood early. By referring to the context of use, we take a broad view consolidating attributes of the physical environment, involved additional tools and equipment as well as communication and workflow patterns during cooperation with other employees.

For instance if usage behaviour is characterized by mobility with customizable handheld devices utilized in exposure of multiple sources of distraction, different emphasizes in design decisions have to be considered than if the context is defined by shared large touchscreen-devices embedded in industrial manufacturing plants.

The consideration of different facets of goals and constraints generated from different perspectives (i.e. functional: product management, business: board,

usability: customer service, technical: development, contextual: setup engineers) enables a shared set of expectations, a common ground for enabling an understanding between stakeholders, paving the path for planning and execution of following activities. Finally this mutual understanding also helps to generate consensus on whether the project is to be classified as a redesign or if a green field approach (generally challenging the current application) is preferred by the stakeholders.

4.2 Analyse

As the cow example showed (see section 2.1), actual users' behaviour tends to differ from the imagined behaviour assumed by product owners – especially regarding the importance of proposed features. Results from Analyse activities provide the most effective vaccination to prevent self-reference when designing an application and help to end effectively otherwise endless feature discussions. The goal of the Analyse phase is to uncover attributes of the user, her tasks and the contextual circumstances of using a future or current application. Viewed from the perspective of the project team, Analyse insights are also of central relevance when opinions of the project team drift apart (which naturally happens due to the intended interdisciplinary setup of the team).

Analyse activities take different implications for a redesign endeavour or approach to replace a current solution compared to creating an innovative solution without a direct precursor. With an existing predecessor of the application and an installed user base at hand, initial Analyse activities frequently focus on accessing its usability status quo. Inspecting the compliance of a user interface with established usability guidelines and investigating its fit to the needs and requirements of its current users provides valuable guiding information for subsequent design efforts. A thorough understanding of the actual version of an application is an especially important prerequisite for being able to appraise (and justify) the amount of relearning necessary for users when introducing significant innovations in the redesign of an application.

In order to arrive at an assessment of an application's usability status we will distinguish between methods with and without end-user involvement in the following sections.

4.2.1 Analysis Without End-Users

Sometimes time, budget or general availability of potential attendants prevents access to real end-users – especially when dealing with highly specific productivity tools for very narrow market segments. In such cases a usability inspection of an application can be approached through heuristic analysis (Nielsen and Molich 1990), often also coined expert review in practice. Heuristic analyses are carried out to identify, document and classify the usability optimization potentials of an interface. In a formative inspection these findings are generally used as input for a subsequent redesign of the system under consideration, while summative reviews are targeted at providing a précis of a system's usability.

During a heuristic analysis, reviewers, guided by a set of heuristic guidelines and established usability principles, inspect an interface to uncover usability obstacles and perceived violations to these heuristics. While this inspection step is typically carried out by two to five reviewers working individually from each other, a consolidation of their findings and an assignment of severity levels to these takes place in a joint session involving all reviewers (Nielsen 1994). In formative evaluations, solutions approaches to identified findings are finally outlined to direct subsequent design activities. Often heuristic analyses are used in comparative inspections of competitors' interfaces as well in order to generate an initial comparative benchmarking.

Heuristic analyses generally offer the advantages of being applicable even to early stages of prototypes, do not presuppose the need of specific equipment or scheduling of real users and inexpensively deliver fast results. They are, however, restricted in their scalability to more complex interfaces that presuppose broad domain knowledge and have been found to be prone to false alarms (erroneously insinuating a usability problem) and missed hits (i.e. failing to identify a usability issue). Independently of their limitations, usability reviews have the positive side effect of familiarizing the participating designers with the current functional scope (Cooper et al. 2007).

4.2.2 Analysis with End-Users

As mentioned before, people who use a product often differ significantly from the ones envisioning, designing, developing, selling and supporting applications. To gain an empathic understanding of the user of a current or future application and to acquire a comprehension of her working goals, researching actual end users in context marks the via regia in a user-centered design approach. A broad diversity of different methods have been suggested to support user research (Kuniavsky 2003), to aggregate results gathered (Holtzblatt et al. 2005) and to relate derived insights to the design stage (Meth 2012). In this section we will sketch user observation by means of a method called *job shadowing* and will exemplify *contextual interviews*, both often used in combination.

Job Shadowing

Job Shadowing provides an opportunity for the designer to observe the user performing her daily work in the actual work setting for a representative period of time, depending on the complexity of the job. Important contextual parameters of the work situation can be experienced during job shadowing including a user's technical equipment that is used (or refrained to use), physical aspects of the environment (e.g. is there enough space available for using a mouse?; to how much dirt will a touch-panel be exposed during usage?; what are the spatial relations of a control center's subsystems?) or social interactions and workflow dependencies (what communication with co-workers is required to complete the task?; what if errors are happening?). Observers receive direct insights into the importance of tasks related to a work place, their frequency and the respective steps involved, the terminology used, the appearance of incidents that are classified as

critical, sources of distraction or the occurrence and repetition of assignments – factors that are of immediate influence on design decisions.

While different models of participation exist, the relationship of shadowed user and observer can adequately be characterized as a master-apprentice setting where an observer's goal is to learn as much as possible about a user's task. Dense notes are taken about the experiences, often enriched by insightful quotes, photos or even video sequences if allowed.

Contextual Interviews

To fill in gaps in the understanding of the job situation, job shadowing is often combined with contextual interviews, with different levels of structuring. Fully structured interviews, guided by a prepared list of questions come with the benefit of a high comparability across interviewees – but leave little flexibility for individually addressing specific questions that result from particular work situations encountered. Unstructured interviews, which follow the flow resulting from the interplay of questions and answers, deliver rich data but carry the danger of getting lost in the idiosyncrasies of a situation and/or a user. In practice, semi-structured interviews often turn out to be an advantageous compromise of richness, flexibility and comparability of results across different interviewees. Contextual interviews are in fact carried out at the workplace where the context serves as probes to trigger questions, and aiding a participant's memory when answering. Nielsen (2001) argues that paying attention to what users do is more important than what they say since self-reported claims are often unreliable. It is for this reason why job shadowing typically precede contextual interviews, with the later rarely being used in isolation.

Not to be neglected at this stage is the possibility to conduct an empirical usability test for getting a comprehensive picture of the usability of an application currently in use. However, since empirical testing is not applied too often as part of a status quo analysis at the beginning of a redesign project, we will cover usability testing in the section describing the Validate phase.

4.2.3 Synthesis

The outcome of Analyse activities leads to a broad stream of data that needs to be explored, classified and pre-processed to provide the required input for Design. Even the most thoroughly performed analysis can be worthless if the gained impressions cannot be consolidated into communicable objects. Specifically during job shadowing and contextual interviews it was learned which tasks are performed frequently with which functions (or even other workarounds) used in which sequence to reach which goals – either alone or in collaboration with other users. Dealing with critical incidents has also been a central topic as was the identification of exceptional situations. Potentially, findings from a heuristic analysis have contributed to understanding how well features and workflows of an existing solution are supporting the user to adequately achieve her working goals. Given these oodles of data a suitable aggregation approach is needed for synthesis.

Practitioning synthesis is a vast field (Kolko et al. 2011) so we want to concentrate on four elements in this section: affinity diagrams, Personas, Mental Models and performed scenarios.

Affinity Diagrams

Using affinity diagrams has proven to be a very effective method that can easily be applied to consolidating and structuring the various impressions gathered. Affinity diagramming is a participatory integration technique where observations, quotes, pictures and other outcomes from the analysis stage are simply written on Post-its that are then clustered and labelled to highlight signature insights. Analyse results can be integrated and relationships identified through the creation and structuring of an affinity diagram. Including different roles in the process allows not only involving those persons who conducted research data in the process, but also to inform other stakeholders and ask for their contribution.

Personas

Cooper's concept of a persona (1999) has turned out to be of eminent help in aggregating data from research to envision and substantiate the user of an interactive system. Being hypothetical archetypes of real users, Personas are defined by distinct goals and individual characteristics, helping to literally put a face to otherwise abstract target demographics. The availability of a conceivable persona with a concrete name, a credible background, documented working goals, responsibilities, challenges and pain points avoids one of the most severe risks in user-centered design project: The design for an elastic "user" who can be bent and reshaped in favour of arbitrary arguments. Instead, a persona represents a tangible incarnation of a concrete person that – based on the partial information provided in a persona description – triggers an emphatic immersion into the needs of the person to be supported by an application. With a persona in mind, a screen in question can be assessed through different lenses, fostering informed decision-making during design.

It might turn out that a single persona description is not sufficient to encompass the variety in the user information gathered, but that different Personas are necessary to adequately capture the research findings in a project. In the case study reported in a later section of this chapter we outline a situation where two very different Personas where identified as archetypes of an application's key user: a highly trained expert who is strictly focused on the efficiency and broad functional coverage of the system in daily use for complex configuration tasks, and an occasional user challenged by mastering a very restricted subset of the available functionality to fulfil comparatively simple configuration set-ups. Approaching the creation of a single user interface to find a compromise by trying to meet the requirements of both primary Personas simultaneously would necessarily result in disenthralling both. While thwarting the performance of the expert by failing to integrate all specific custom functions for immediate access, the occasional user would be overwhelmed by endless options to enhance the complexity of the user interface without being of beneficial use for her. The solution followed in this situation was to provide two distinct user interfaces, acknowledging the individual

requirements of the primary Personas and reflecting different priorities in the use of functions. The example of sacrificing efficiency for learnability and memorability (and vice versa) in the two interfaces also demonstrates that Personas provide information to advise for the appraisal of usability goals that are then reflected in design decisions.

Mental Models

Persona descriptions are often also a point of origin for a discussion of the Mental Models of a user. Without the claim of being a valid or complete representation of its functioning, the concept of a Mental Model describes the sum of a user's subjective mental conceptualizations and believes of how a system works based on experience (Lidwell et al. 2010). Mental Models guide the actions of what to do next in order to arrive at specific goals and guides a user's reasoning and beliefs of what to do in unexpected situations. If there is a significant mismatch between a user's Mental Model and the conceptual model underlying the operating of a system, interaction is likely to break down. Norman (1983) refers to conceptual models as tools for the understanding and teaching of physical models. Picking up the differentiation between conceptual and Mental Models, an important prerequisite for successful user interfaces is to base the design on a coherent and graspable (but not necessarily physically correct) conceptual model to induce a stable Mental Model in congruence with a user's understanding of the task.

In the project reported in the case study later in this chapter, results from the Analyse phase revealed clear evidence that the Mental Models of users conflicted with the conceptual model underlying an application. As a result of this mismatch, using this application, determining what to do in exceptional circumstances and especially learning how to operate it was significantly impaired.

Scenarios

A missing ingredient for paving the ground for the design of a sound interface concept is to encapsulate what tasks have to be supported by an application through scenarios. These are simple narrative stories (the *what* of interaction) articulating a persona through an environmental context to achieve a specific goal (the *why* of interaction). Scenarios provide concrete descriptions but are open to how a persona is achieving a particular task goal – leaving the details of interaction to be filled in by design solutions. Scenarios do not focus on technology but are focusing on the user's view. Written in plain language and employing the terminology of users they come in different flavours ranging from simple goal statements (describing simply what the user wants to do) to elaborated user stories providing enriched contextual details. In contrast to use cases, scenario descriptions refrain from using formal notation, which simplifies their validation in discussions with users. Although we discuss the use of scenarios with reference to an analysis involving end-users, it is not uncommon to also adopt scenarios as a narrative tool without distilling actual user research.

By fleshing out the variety of scenarios that are to be supported, the underlying structure of an application begins to take shape before any design happens at all. In the context of an outsourced design process scenarios prove to be a very economic

instrument for communication. Rather than enumerating feature by feature of a planned application in abstract, an analysis of scenarios allows the derivation of common features de facto needed for their realization. Depending on the project, not all scenarios are explicitly worked out, but key scenarios are identified to represent associated instances, leaving their extension to related ones to the design stage.

Having defined Personas, Mental Models and scenarios as typical outcomes from a synthesis of data, insights and potential heuristic analysis, the next section covers the anchoring of design decisions on these grounds.

4.3 Design

Designing a user interface is a process of balancing potentially conflicting requirements, considering the trade-offs of resulting solution explorations and acknowledging the constraints of the solution space. Its goal is to transform insights and findings from the Scope and Analyse phases into a tangible artefact. Having already emphasized that User Interface Design goes beyond the definition of purely visual attributes, design activities address two layers of the interface connecting users with the functionality of an application: the conceptual and the visual layer. While both layers are not independent from each other, processes for their creation ideally do not overlap too much, with conceptual design preceding its visual elaboration.

4.3.1 Conceptual Design

The conceptual layer of a user interface refers to the result of decisions regarding the layout (if graphical user interfaces are the target of the design efforts), the workflow and the underlying interaction model. Conceptual decisions comprise determinations of the spatial relationship of screen views, their approximate dimensions, the use of interface controls and the interrelation of screens. Picking up the aforementioned analogy to architecture, conceptual design renders the floor plan, the skeleton of a user interface that is further supported and enriched by appropriate visual design. Conceptual design decisions provide the answers to the *how*-questions that were deliberately left open during scenario construction when explaining *what* a persona does, concentrating on the *why* of following her task goal. Personas and scenarios thus form central ingredients for informing conceptual design about the targeted functional and non-functional requirements of the intended future application. Helpful Interface Design Guidelines (Johnson 2007; Shneiderman and Plaisant 2009; Raskin 2000) are rooted in the scientific understanding of how *people perceive, learn, reason, remember, and convert intentions into action* (Johnson 2010, p. xiii) that is provided by Cognitive Psychology. Although being abstract to allow for a broad applicability, such guidelines direct design efforts to meet elicited requirements through the creation of a user interface that takes the capabilities and limitations of the human cognitive system into account.

Scribbles

Simple scribbles are of valuable help to arrive at initial representations of conceptual design, sketching screen areas and outlining their mutual relationship without already fixing their true physical dimensions. Adding adumbrations of interface controls then provides a first coarse picture of how functionality might be wrapped by the interface. Such often hand-drawn scribbles – not although but because of their obviously unfinished and tentative nature – represent a manifest artefact for fostering communication, effectively supporting reflection to allow for the elevation of early feedback regarding the design directions taken. It is one of the insights of project experience that the more elaborate a design artefact is, the narrower will be the feedback that can be expected. E.g. when exposed to user feedback, elaborated design suggestions are mainly tweaked for incremental improvement – but rarely challenged in their essential appropriateness. It is exactly the inchoate, incomplete nature of a scribble that provokes questioning its high-level features because their typically hand-drawn effectuation spells out a disposition for revision without much effort – also evidencing that not much expense has been invested yet into their creation.

Albeit exaggerating slightly, one of the central differences in training when comparing computer scientists to designers lies in the general approach to solution finding. While computer scientists focus on selecting the most promising candidate from a (usually small) set of potential candidates for implementing an algorithmic solution, designers tend to explore and evaluate a broad range of alternatives in finally arriving at the right design. Following a process of successive elimination of candidates' branches while traveling through the solution space, the design process is described by Laseau (1980) as a symbiotic relationship between the elaboration of generated solutions and their subsequent reduction to iteratively decide on the ones worth to be further pursued (see also Greenberg et al. 2012). Scribbles are perfect tools for not committing oneself too early to a single design solution, thus avoiding the danger of getting stuck in a local maximum of the solution space. Due to their unwrought character, a set of scribbled solution candidates can be reduced to the more promising ones without the guilty conscience of having wasted valuable resources in their creation. Cutting candidate branches in the solution space can be justified, for example, by analytical considerations (for example by taking the implementation effort for a technical realization into account), or by iteratively evaluating the concepts expressed by scribbles with prospective users (see Sect. 4.4).

Wireframes and Prototypes

Pugh (1990) argues that the granularity of solution explorations is successively refined through the course of elaboration and reduction cycles, suggesting to move from the use of scribbles to so-called wireframes to reflect an increased maturity of design candidates. In contrast to scribbles, wireframes – while still being silent about the visual details of the interface – provide a consolidated view of the spatial arrangement and dimensions of an interface's layout. Wireframes communicate decisions on the selection of interface controls and navigation elements and also allow for an initial exploration of using a dedicated colour to guide a user's attention to specific interface areas. The increased concreteness of wireframes

also enhances the scope of potential evaluation questions that can be addressed to also comprise, for example, validations of the appropriateness of interface controls or investigating whether the layout reflects the logical path through a screen.

Basing design activities on the use of scribbles and wireframes grants quick iterations, working out varieties of alternative solutions while subsequently moving from the bird's eye perspective of scenarios down to the specifics of interface controls. By the necessity to abstract from decorative details through the constricted expressiveness of mainly having levels of grey for conveying design decisions, wireframes focus the attention of a designer on establishing a sound logical and consistent foundation of a screen.

Again emphasizing the iterative nature of user-centered design by referring to empirical validation of early design results, valuable feedback can be gathered when involving users to inform the respective next elaboration cycle until quiescence is reached and a convincing interface solution is found. Because of their typical existence on paper (or as static digital depictions on a computer screen), neither scribbles nor wireframes are interactive – but both incarnations of potential design solutions can nevertheless be used in pseudo-interactive workflow evaluations. With a moderating facilitator in place, wireframes presented on paper can be switched in dependence of a participant's simulated *click* on the outline of a button on paper. Alternatively, simple concatenations of wireframes (or, in a slight enhancement of the setup: the use of dynamically linked areas in these) can be arranged in presentation software for simulating "interaction". Just as scribbles and wireframes instantiate different degrees of fidelity, paper prototypes are early reflections of interactivity, coarsely illustrating the intended *behaviour* of concepts under consideration. Fully interactive prototypes, created by the use of dedicated prototyping tools like SKETCHFLOW, AXURE or ANTETYPE, Java-Script frameworks or occasionally even written in the target language of an application's development framework, then excite the experience of the consecutive interplay of action and reaction spanning over time when interacting with an application. Interactive prototypes render the tangible *feel* of a future application, vividly demonstrating *the way products behave in response to the behaviour that people behave* (Saffer 2007, p. 44). Prototypes hereby also serve highly important communication purposes by allowing all stakeholders in the design process to get their hands literally on otherwise abstract emerging ideas, encouraging reflective discussions and triggering *what-if*-questions regarding future iterations.

4.3.2 Visual Design

It depends on the information needs during the design process whether to work out the appearance of an interface – its *look* – before exploring interactional aspects on the grounds of an interactive prototype.

Although mostly applied after having sanded and polished the layout and interaction details, there is, however, no reason for debating on the importance of visual design. User interface *mock-ups* are representations in which visual design decisions like colours, textures, exact font definitions, icons and decorations like the use of glossy highlights are defined on the pixel-precise layout of an interface.

Funnelling the input from the *Scope* phase regarding corporate design guidelines (if applicable), market research findings on user preferences and also technical possibilities, a range of visual styles is explored in interface mock-ups. As stated before, visual design is not independent from conceptual design, but provides the expressive means to effectively support the conceptual level. Reducing visual design to a simple bedaubing of wireframes in order to match the taste of prospective users (or the aesthetic preferences of a client) falls short of acknowledging the tight relation of conceptual and visual design. Purposeful visual design decisions help to emphasize the structure written out by conceptual design, prescinding hierarchies of interface elements and directing the attention of a user.

But of course visual design is also about the overall aesthetic impression of an interface – the appearance of it, after all, being so much more immediate to grasp than interaction. It is one of the old lessons learned from Social Psychology, known as the *Attractiveness-bias*, that humans have a tendency to perceive physically attractive people as more intelligent, competent, sociable and even taller (when asked to estimate height of people presented on portrait photographs) than less attractive ones (Dion et al. 1972). This bias has a counterpart result in the so-called *Aesthetic-Usability Effect*, describing that users *perceive more aesthetic designs as easier to use than less aesthetic designs – whether they are or not* (Litwell et al. 2003, p. 18; Kurosu and Kashimura 1995). Visual design thus clearly contributes to the acceptance of a user interface, enhancing its desirability, stimulating identification and increasing its hedonic quality (Diefenbach and Hassenzahl 2011).

Visual design efforts can range from applying standard operating system *Look & Feels* with stock icons to extensively styled applications with laborious graphical assets and custom icons. Reiterating on the architecture metaphor, a visually embellished interactive prototype of an application transcends the floor plan and can be compared to the tangible outfitted three-dimensional models that aid imagination nowadays in allowing for an interactive exploration of a house's future rooms.

A leitmotif running through the design phase – that preferably will subsequently become part of a project team's DNA in user-centered design approaches – is the appreciation of iteration, frequently validating assumptions by evaluating increasingly refined artefacts. Prototypes can be thrown away after validation or be evolutionary in the sense that they are revised and extended during validation. Especially with the advent of powerful prototyping programs it is easy to move from raw scribbles to static wireframes, fine tuning these over time to arrive at prototypes enhanced with realistic interactions and transitions vested in an elaborated visual design. The potential (and temptation) for perfection is endless, explaining why prototypes are also frequently used for convincing a product owner's board members and potential key clients before a single line of functional code being written (UIF).

4.4 Validate

As carved out in the description of the previous section, the activities in the Design and the Validate stages are strongly intertwined in practice. Independent of an artefact's fidelity the assumed progress in design efforts needs to be iteratively validated against goals to appreciate its appropriateness and maturity. This can be done by a project's team checking the current design state against the requirements derived from analysis (*Does this interface adequately support the person in achieving a scenario's goal?*), by Heuristic Analysis to inspect its usability status (see Sect. 4.2.1) or through empirical usability testing, covered in the following paragraph.

4.4.1 Usability Testing

Usability Testing is often associated with the availability of a dedicated usability lab. In such a lab setting, participants are recruited to match the profile of prospective users of a current or future application. The persona description that was established in the Analyse phase and used during design to keep the user in focus now helps to define a recruiting brief for inviting participants that are regarded to represent future users. Although usability tests are empirical studies involving human participants, they are not to be confused with experiments. Experiments serve the purpose of testing scientific hypotheses about expected causal relationships. Usability tests, on the other hand, focus on identifying the obstacles to frictionless interaction, conceptual obscurities or learning barriers of an interface. Picking up terms from the previously emphasized definition of usability, usability tests do not evaluate user interfaces as such but do so in reference to significant goal-directed tasks of a user representative working in a typical context.

The recruiting brief assures the representativeness of participants while the scenarios (representing the main task goals identified in the Analyse stage) can be reused to formulate task goals that are to be attained in testing sessions using an interface prototype. In a usability test setting, participants work through a set of task scenarios with their behaviour being monitored from an adjacent observer room. A Venetian mirror, allowing to look from the observer room to the participant room but not in the inverse direction, is a typical feature of a usability lab. Specific recording software packages like MORAE are used to capture the user-system interaction as well as behavioural indicators of the participant. Digital cameras record the facial expressions of a user, while eye trackers can be used to register her visual fixations. To gather insights into the accompanying thinking processes of participants, the *thinking aloud method* (Ericsson and Simon 1980) is often used: During interaction participants are instructed to concurrently verbalize what comes into their minds, providing conclusive evidence for the reasons behind a user's action – thus helping to retrace her train of thought that is withdrawn from direct observation.

Although in some publications incorrectly coined *user testing* (Bernsen et al. 1999) it is of course not the user who is being tested – the focus of interest is to evaluate whether the interface matches the usability requirements of a task and the

expectations of a user in the work context approximated in the lab. For reasons of contextual inadequacy it would thus be a violation of representativeness to conduct a usability test of a mobile surveying application in the narrow space of the participant's room of a usability laboratory.

Whenever possible it is highly commendable to invite as many stakeholders as possible to attend at least selected testing sessions. Sitting in the observation room opens up the chance for assembling a comprehensive picture of the user experience by monitoring the user in action, proximately comprehending her successes and failures – and the reasons behind these: *Arguably, the most valuable contribution of usability testing is made when programmers are forced to sit behind the one-way mirror to view typical users struggling with their programs. The programmers are shocked and incredulous, shouting sentiments like, "You are testing mental retards!" Usability testing is a useful whack on the side of the head for recalcitrant software engineers, showing them that there is indeed a problem* (Cooper 1999, p. 207). While we endorse Cooper's statement, invitations to stakeholders for attending testing sessions are not restricted to developers – we can confirm on the empirical grounds of many projects that all stakeholders greatly benefit from the shared experience of jointly monitoring the course of interaction, rendering many otherwise vivid future discussions obsolete.

Usability testing comes in many flavours, ranging from testing in a full-fledged laboratory to informal guided walkthroughs in which participants are visited in their respective working context and questioned while being led through a (paper) prototype. Especially in the early stages of conceptual design renting a full lab often appears to be over the top while walkthroughs allow for fast and helpful feedback without much organizational effort. Worth mentioning is also the possibility to conduct remote usability tests via some screen sharing application. Although having economic and practical advantages, remote testing poses strong challenges on appropriate guiding and briefing of the participants to achieve the required quality of feedback. In addition, important capillary reactions of participants such as full body language or subtle variations in their voice might get lost in translation.

In conjunction with usability tests usability questionnaires and post-task interviews are often used to get an overview of the subjective impressions of a user regarding the usability of a system. The System Usability Scale (SUS, Brooke 1986; Sauro 2011) is an example for a compact questionnaire to elicit the perceived usability of a user judging a system (that might as well be embodied as an early prototype). Summarizing the result in a simple one-dimensional score allows for easy interpretation of the SUS and – when comparing scores over subsequent iterations – for quantifying the success (or failure) of an iteration.

4.4.2 Usability Goals

Any iterative approach needs to be informed by some termination criteria about when to stop and deliver the result of design activities – the final artefact, found to be sufficiently adequate for the targeted purpose. We need, however, a clear operationalization of this vague formulation of being *sufficiently adequate*. Simply

asking designers for their impression puts us in jeopardy of iterating beyond necessity. Setting up explicit usability goals provides an appropiate solution in this situation. Usability goals are quantifiable criteria that reflect target (or acceptance) indicators on usability metrics that are found to be relevant for an application. This raises the question of when to formulate these usability goals in terms of their target or acceptance values.

The correct answer to this question is related to the type of project: When redesigning the interface for an application, discussions in the Scope phase might lead to at least comparative criteria (i.e. increasing the efficiency [operationalized as the mean time on task for the core tasks done with this application] by at least 15 % (*target value*; with the *acceptance value* being at least 10%). In such case we are not only able to quantify the usability goal by reference to experiences with the current version of an application, but we also have an understanding regarding a usability metric of relevance. This knowledge is typically not available when commissioned to design the interface for a completely new product. In this situation the Analyse stage might shed light on the metric(s) being relevant. Building on this information, contextual interviews are likely to provide initial cues for the respective target and acceptance values on this metric from a user perspective.

Usability goals are of central importance in discussions with the client, but generally need to be considered for their technical and design feasibility. Model-based analyses (John 2010) grounded on the GOMS-approach (Card et al. 1983) have been shown to be an especially powerful method for quantitatively predicting the performance using a designed interface. Although a more detailed discussion of model-based analysis is beyond the scope of this section, it should be noted, however, that this approach is to a large extent restricted to performance predictions for highly practiced routine tasks with efficiency being the metric under consideration.

After successfully meeting previously defined usability goals in empirical usability tests, model-based analyses or by the use of other inspection methods, the Validate stage comes into quiescence. The next section focuses on delivering the validated result of a user-centered design endeavour.

4.5 Deliver

In the Deliver phase the result of the Design stage, typically enhanced with explaining documentation and instructions, is handed over to development. The exact form, timing and granularity of the respective deliverables depends on the concrete project situation. Deliverables may range from mock-ups of selected key screens to comprehensive interactive prototypes, documented by brief descriptions. On the other side of the spectrum they may be extensive written specifications of an interface's structural, visual and behavioural properties. The main graphical assets like icons may be provided – or the full XAML-coding of an application's interface presentation layer, ready for integration with the business logic. Last but not least, the nature of deliverables also depends on the frame (waterfall or agile) of a project.

Certain dimensions for classifying documentation, however, always apply: Documentation can either be *static* (often lengthy text files with a dedicated owner and a strict release mechanism) or *dynamic* (typically in the form of WIKIs where different roles can edit at the same time and updating is simplified).

Documentation can be embodied in the form of a comprehensive *generic* style guide for harmonizing the interfaces of different applications of a company (ensuring overall consistently with regard to colours, measures, controls, fonts, common design patterns etc. by explicit rules for their application). On the other hand, it can be very *specific* in narrowly describing the nitty-gritty details of all core screens of a single application (often then called an *interface specification* to distinguish it from style guides). In real life projects conducting reviews at certain stages of application implementation frequently ensures compliance to such documentation provided.

Especially in the past years, technological advances provided an opportunity for designers to give developers more than just documentation and icons as reusable assets. Modern frameworks allow casting (visual) design in code for selected *Look & Feel* engines. Provided that the *Look & Feel* code is sufficiently separated from functional code, developers then do not need to care about the appearance of the elements. This approach not only allows designers and developers to communicate by reference to the very same language (instead of talking at cross-purposes by speaking in PHOTOSHOP to C# et vice versa). It also relieves the documentation effort significantly by providing developers with code ready for subsequent integration. From a designer's perspective this also comes with the advantage of preserving more control over the result of the final implementation of the validated interface concept. Although not being trivial for reasons outside the scope of this chapter, the approach outlined finally leaves open the possibility of developing and iteratively validating an evolutionary interactive prototype on a code basis that can then largely be reused during implementation.

With the differentiation of Scope, Analyse, Design, Validate and Deliver stages we have summarized the core process of user-centered design in the pervious sections. In the next section we follow the goal of briefly illustrating the course of these stages in a real life case study.

5 A Brief Case Study

Referring to the distinction between UIF and UIL introduced in the second section of this chapter, the case study sketched here exemplifies a prototypical example for a *user interface first* approach. In fact, the major deliverable of the project on which the study is based was a truly comprehensive and detailed 300 + pages user interface specification that formed the core of an invitation of tenders. The winning bidder of this tender was finally commissioned to implement the full application with the interface specification comprising the requirements stage of the waterfall model. The client mandating our company for redesigning the user interface is the owner of a worldwide standard for home and building control allowing building management components to communicate via a common bus system. The

engineering tool (*ET*) is the respective configuration application to design and configure sophisticated home and building control installations. To avoid confusion, its redesigned interface will subsequently be coined *ET**.

5.1 *ET**: Scope

A comprehensive scoping workshop was conducted in which the applicant informed about already identified functional and non-functional shortcomings of *ET* that were based on an analysis of user feedback collected by the application hotline. In addition to fixing the technical framework for implementing the future *ET** solution, different stakeholders reported extensively on perceived learnability problems that they mainly associated with usability problems of the *ET* user interface, suggesting learnability as an important usability metric. A timeline for the project was fixed and a sequence of workshops to present intermediate results at selected stages was agreed upon.

5.2 *ET*: Analyse

The domain of home and building control systems covered by the *ET* application is complex. More than 250 leading manufacturers in this field provide components that can be integrated using *ET* with more than 30,000 installer companies in 110+ countries being active users of the configuration software. Over 215 partner training-centers have been established to instruct installers in courses of several days' duration on using *ET* for configuring residential and commercial buildings.

Before participating in one of these courses to gain deeper knowledge about *ET*, interface designers of the project team had to prepare themselves by studying introductory textbooks and white papers to get acquainted with the domain. By education most users of *ET* are either electrical engineers or electricians, so the designers had to master a challenging wealth of materials to come sufficiently up to par. While a project team might be split into separate roles for researchers and designers, with researchers covering the Analyse phase and then handing over (written documentations of their) insights to designers for Design, we usually involve designers directly in the analysis phase. Having the advantage of fostering unfiltered empathic knowledge about users, their task and the domain, this also eliminates the difficulty for researchers to anticipate the design importance of information and insights during documentation.

Equipped with the knowledge from having participated in the trainings, the designers were prepared for job shadowing and contextual interviews to understand how to work with the *ET* application. To consider aspects resulting from the international use of *ET*, these activities were carried out in Germany, Spain, France, the UK and Belgium.

A heuristic analysis of *ET* completed the picture to identify current usability shortcomings allowing finding an optimal trade-off between innovation and

re-learning when designing *ET**. To acknowledge the space limitation of this chapter, we like to present two selected findings from the Analyse phase:

5.2.1 *ET*: Personas

When establishing Personas for ET through affinity diagramming it became clear that two primary Personas were necessary to conform the research results. One persona covered the professional user, typically trained as an electric engineer and ready to configure ET for commercial buildings as large as London Heathrow airport. The other primary persona represented an occasional user, often trained as an electrician and mostly using ET for configuring small, one-family residential buildings. Job shadowing also revealed significant differences in the technical equipment of the respective Personas: While the professional user typically had access to cutting edge laptops to run ET, the occasional user was fighting with ET's system requirements on technically rather out-dated machines.

5.2.2 *ET*: Mental Models

The Mental Models of the respective Personas were assumed to be significantly different, with the conceptual model underlying ET being up to now only in adequate correspondence with the Mental Model of the professional user. ET required users to think in terms of networks during configuration tasks. Contextual interviews with electricians revealed, however, that their Mental Model was grounded on their education of thinking in terms of switches being connected to appliances in electric circuits. This mismatch of the Mental Model of occasional users and the conceptual model of ET resulted not only in massive difficulties during learning the application, but was also hampering the user during failure detection in real life configurations.

These two findings related to Personas and Mental Models, although being more or less arbitrarily picked-out from a wealth of insights gathered in this phase, already provided very important food for thought for the design phase.

5.3 *ET**: Design

Although reported in sequence for reasons of linearity when reading from paper, the Analyse and the Design phase were substantially intertwined during actual project execution. Illustrating this overlap, scribbles and wireframes were used in selected contextual interviews mentioned above to elicit concrete feedback on early design visions. Exemplifying the descriptions in Sect. 4.3 of this paper, the design process moved through the creation and subsequent refinement of scribbles, wireframes, paper prototypes and visual design variants to the development of an interactive prototype. Starting from the key scenarios identified during the Analyse phase, the provision of two primary Personas called for the introduction of two distinct interface types to be integrated into *ET**, reflecting the differing requirements of the professional and the novice user.

While the *professional* mode allowed full and flexible access to the complex functionality provided by *ET**, the *novice* mode concentrated on offering a guided configuration process based on separate steps that matched the Mental Model of occasional users by the automatic allocation of network addresses. Albeit being restricted in the number of appliances for integration during configuration, the demonstrative visualizations offered in the *ET** novice mode not only fostered learning how to operate the interface, but also facilitated a smooth transition to the *ET** professional mode. After arriving at a mature state of the interface concept, a comprehensive interactive prototype for *ET** was developed allowing to work through the key scenarios of a representative future *ET** user.

To continuously inform all stakeholders – including the client's board – about the progress in the project, the interactive prototype (which was implemented using a JavaScript framework) was made available for inspection via password-protected web access. This procedure not only kept management up to date, but also resulted in unexpectedly vivid, helpful and often very positive feedback.

5.4 *ET**: Validate

As mentioned in the previous section, initial explorations were already conducted with early paper scribbles to consolidate the interface concept behind *ET**. In the Validate stage, a broad empirical user interface evaluation was arranged, recruiting prospective user representing professional and novice participants. Inspection of behavioural and thinking aloud data, as well as subsequent post-task interviews collected, revealed minor usability stumbling stones that were subsequently revised in an updated version of the *ET** prototype. The first half of the participants was invited to the first part of the testing sessions taking place at ERGOSIGN's usability lab. Based on their feedback and performance the *ET** prototype was refined before being exposed to the other half of participants in a second set of test sessions. Given the comprehensive input from the Analyse phase and the experiences gathered with early paper prototypes it was no surprise that the second set of testing sessions provided evidence for a quiescence of the design process, resulting in a sound and validated user interface concept.

5.5 *ET**: Deliver

As stated in the introduction to this section, the main deliverable in this project consisted of a comprehensive user interface specification used as the basis for an invitation of tenders for implementing *ET**. During the implementation process we were reviewing the state of the application with regard to the interface specification, picking up the chance for minor adjustments or – in case of emerging technical challenges – eventually discussing and providing alternative solutions for aspects of the user interface. Interestingly, the interactive user interface prototype turned out to play an unexpected role – besides providing an illustrative companion to the specification – after having fulfilled its part in empirical evaluation.

5.6 A Perspective for eLearning

Analysis brought many insights that shaped the *ET** interface in its current form. These insights also uncovered learning difficulties regarding the understanding of domain concepts as well as specific steps in the configuration process. Most of these findings were considered during Design, some of them had to remain challenges even in the final version of the user interface, mostly because they were part of the intrinsic cognitive load of understanding the complex domain of configuration in home automation. On the grounds of our experiences with making the interactive *ET** prototype available to the project's stakeholders via web access, the idea of re-using it as part of an eLearning-system to support the introduction and training of *ET** rose.

After presenting the idea to the board we were commissioned with the development of an eLearning platform. In this comprehensive eLearning-tool learners are exposed to didactically founded declarative lessons on the basics of home automation and configuration, to interactive quizzes for self-testing their learning progress, to illustrative multimedia screen-casts of user-system interaction based on the scenarios established in the analysis phase – and finally to challenging exercises with the *ET** web simulation to actively explore the actual interface without the need of installing the full application. Taking the full *ET** eLearning course requires about seven to eight hours of learning time and may – if all lessons and their corresponding tests are achieved – result in the display of a certificate to demonstrate the learning success.

This approach not only made heavy reuse of the artefacts created during the user-centered design process: Personas for focusing the eLearning content on occasional users and scenarios for setting up the storyboards of the learning lessons and the interactive prototype as the core of the *ET** web simulation. It also resulted in a highly successful multi-language web-based learning platform with almost 3,000 active users and a mere 1,300+ achieved *ET** certificates within the first 3 months of its release.

Currently conducted accompanying research is focusing on exploring and quantifying the exact learning benefits associated with the use of the eLearning platform to support training the *ET** application. Preliminary results clearly point to especially fruitful synergies of user-centered design and eLearning with the interactive prototyping being the aorta of both.

Conclusion

In this chapter we presented the methodological cornerstones of user-centered design. Using a case study for illustration, we discussed the application of these cornerstones in a real-life project and pointed to the relationship of UCD and eLearning. To put it in a nutshell: It is safe to say that Gould and Lewis' key principles from the IT stone age (1985) have survived the dynamics of time and still prove to be valid in these days. The necessity of a user-centered approach in

software development is, in our opinion, beyond dispute and due to the abundance of digital interfaces in our daily lives even more crucial for sustainable differentiation than ever. User-centered design offers a great set of different tools that can easily be adjusted to fit any combination of project type, scope, timeline, budget and team setup – even without having someone like Steve Jobs on board.

References

Bernsen, N.O., & Dybkjær, H. & Dybkjær, L. (1998). Designing Interactive Speech Systems. From First Ideas to User Testing. London: Springer Verlag.

Biddle, R., Ferreira, J., & Noble, J. (2007). Agile development iterations and UI design. *Proceedings of the agile*, Washington, DC.

Brooke, J. (1986). SUS: A quick and dirty usability scale. In P. W. Jordan, B. Thomas, B. A. Weerd-meester, & A. L. McClelland (Eds.), *Usability evaluation in industry* (pp. 189–194). London, UK: Taylor & Francis.

Card, S., Moran, T., & Newell, A. (1983). *The psychology of human-computer interaction*. Hillsdale: Lawrence Erlbaum Associates.

Clark, H. H., & Brennan, S. E. (1991). Grounding in communication. In L. B. Resnick, J. M. Levine, & S. D. Teasley (Eds.), *Perspectives on socially shared cognition* (pp. 127–149). Washington, DC: American Psychological Association.

Cooper, A. (1999). *The inmates are running the asylum; Why high-tech products drive us crazy*. Indianapolis: Macmillan.

Cooper, A., Cronin, D., & Reimann, R. (2007). *About face 3: The essentials of interaction design*. Indianapolis: Wiley.

Diefenbach, S., & Hassenzahl, M. (2011). The dilemma of the hedonic – Appreciated, but hard to justify. *Interacting with Computers, 23*(5), 461–472.

Dion, K., Berscheid, E., & Walster, E. (1972). What is beautiful is good. *Journal of Personality and Social Psychology, 24*(3), 285–290.

Ericsson, K. A., & Simon, H. A. (1980). Verbal reports as data. *Psychological Review, 87*, 215–251.

Gould, J. D., & Lewis, C. (1985). Designing for usability: Key principles and what designers think. *Communications of the ACM, 28*(3), 300–311.

Greenberg, S., Carpendale, S., Marquardt, N., & Buxton, B. (2012). *Sketching user experiences: The workbook*. San Francisco: Morgan Kaufmann.

Holtzblatt, K., Wendell, J. B., & Wood, S. (2005). *Rapid contextual design: A how-to guide to key techniques for user-centered design*. San Francisco: Morgan Kaufmann.

Jobs, S. (2007). http://www.european-rhetoric.com/analyses/ikeynote-analysis-iphone/transcript-2007/. Accessed 29 April 2012.

John, B. E. (2010). Reducing the variability between novice modelers: Results of a tool for human performance modeling produced through human-centered design. *Proceedings of the 19th annual conference on behavior representation in modeling and simulation*, Charleston.

Johnson, J. (2007). *GUI bloopers 2.0: Common user interface design don'ts*. San Francisco: Morgan Kaufmann.

Johnson, J. (2010). *Designing with the mind in mind: Simple guide to understanding user interface design rules*. Burlington: Morgan Kaufmann.

Kolko, J., et al. (2011). *Exploring the magic of design: A practitioner's guide to the methods and theory of synthesis* (pp. 59–61). Oxford, UK: Oxford University Press.

Kuniavsky, M. (2003). *Observing the user experience: A practitioner's guide to user research*. San Francisco: Morgan Kaufmann.

Kurosu, M., & Kashimura, K. (1995). Apparent usability vs. inherent usability: Experimental analysis on the determinants of the apparent usability. *Conference companion on human factors in computing systems* (pp. 292–293). New York: ACM Press.

Laseau, P. (1980). *Graphical thinking for architects and designers*. Indiana: Wiley.

Lidwell, W., Holden, K., & Butler, J. (2010). *Universal principles of design, revised and updated: 125 ways to enhance usability, influence perception, increase appeal, make better design decisions and teach through design* (2nd ed.). Beverly: Rockport.

Mayhew, D. (1999). The Usability Engineering Lifecycle. San Francisco, CA: Morgan Kaufmann Publishers.

Meth, H. (2012). Reminer. http://www.youtube.com/watch?v=oavu6wX2bJc. Accessed 29 April 2012.

Nielsen, J. (1994). Heuristic evaluation. In J. Nielsen & R. L. Mack (Eds.), *Usability inspection methods*. New York: Wiley.

Nielsen, J. (2001). First rule of usability? Don't listen to users. http://www.useit.com/alertbox/20010805.html. Accessed 29 April 2012.

Nielsen, J., & Molich, R. (1990). Heuristic evaluation of user interfaces. *Proceedings of the SIGCHI conference on human factors in computing systems: Empowering people*, Seattle (pp. 249–256).

Norman, D. A. (1983). Some observations on Mental Models. In D. Gentner & A. L. Stevens (Eds.), *Mental Models*. Hillsdale: Lawrence Erlbaum Associates.

Pugh, S. (1990). *Total design: Integrated methods for successful products engineering*. Essex: Pearson.

Raskin, J. (2000). *The humane interface: New directions for designing interactive systems*. Amsterdam: Addison Wesley.

Saffer, D. (2007). *Designing for interaction: Creating innovative applications and devices*. Berkley: New Riders.

Sauro, J. (2011). *A practical guide to the system usability scale*. Denver: CreateSpace.

Shneiderman, B., & Plaisant, C. (2009). *Designing the user interface: Strategies for effective human–computer interaction*. Boston: Addison Wesley.

Simon, H.A. (1969). The Sciences of the Artifical. Cambridge, MA: The MIT Press.

Tullis, T., & Albert, B. (2008). *Measuring the user experience*. San Francisco: Morgan Kaufmann.