

# A Requirement Engineering Approach to User Centered Design

Dirk Zimmermann<sup>1</sup>, Lennart Grötzbach<sup>2</sup>

<sup>1</sup> T-Mobile Deutschland GmbH, Landgrabenweg 151, 53227 Bonn, Germany

<sup>2</sup> Siemens IT Solutions and Services C-LAB, Fürstenallee 11, 33098 Paderborn, Germany

<sup>1</sup> Dirk.Zimmermann@t-mobile.de, <sup>2</sup> Lennart.Groetzbach@c-lab.de

**Abstract:** This paper describes an approach to integrate UCD activities into the existing Software Engineering practices and processes. The aim is to use the outcomes of UCD activities throughout the development process and to ensure that they can be utilized, traced and tested by subsequent development groups. Through this, UCD activities do become planable and manageable just like any other development activity. The authors introduce a framework of three different usability-related requirement types that reflect the results of the UCD activities performed during the software development. Each requirement type is extracted from the UCD results generated in the first three phases of the DIN EN ISO 13407 model.

**Keywords:** Usability Engineering, Requirements Engineering, Processes, Integration.

## 1 Introduction

Usability as a product property has been defined to be “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” [7]. This definition has gained acceptance in a wide area of the scientific and professional HCI community, therefore a lot of Usability Engineering (UE) and Testing approaches revolve around these three criteria.

However, during an UE process, other facets of Usability that are equally important become relevant. For example, the seven principles of dialog design from DIN EN ISO 9241 Parts 10 to 12 focus on various workflow-oriented factors that ultimately influence Effectiveness, Efficiency and Satisfaction, but give more specific guidance on the way in which they can be achieved. Also, the guidelines for information display emphasize the way in which data should be conceptualized and rendered, thus highlighting the concept and design side of UE.

Looking at these three facets of Usability, i.e. general Usability Principles, Dialog Principles and Information Display Principles, the question arises how all of them are interrelated, how they can be represented in a UE process sufficiently and how they can be assured e.g. through evaluation.

The approach taken in this paper is to create a Requirement Engineering (RE) framework that distinguishes three different types of Usability-related requirements that correlate to the three facets: Usability, Workflow and Design. Knowing that most UE processes are embedded into a more complex Software Engineering (SE) process, one of the goals was to align the framework both with current best practices in Usability Engineering and with existing RE approaches in the Software Engineering discipline.

## 1.1 Requirement Engineering

Current software development practices use software requirements to specify the functional and non-functional aspects of software systems. While functional requirements specify the services and functions the system should provide and how it should react and behave, non-functional requirements define constraints to the offered services and functions. Thus non-functional requirements define the product quality. In the ISO/IEC 9126 six non-functional requirements dimensions are differentiated – Usability is one of them [13].

The process of requirement elicitation, as well as the results of the requirements engineering activities, is well described in today's literature. The IEEE STD 830-1998 [12] describes the resulting artifact, the *Software Requirement Specification (SRS)* as a document that “correctly define[s] all of the software requirements” of the system to be developed. Each of these specifies a “software capability needed by a user to solve a problem to achieve an objective” [1].

For the SRS the IEEE STD 830-1998 also defines quality attributes such as correctness, unambiguousness, completeness, consistency and verifiability. Thus these attributes apply to each individual requirement contained in the SRS. This is a huge benefit for the development process since a SRS provides verifiable and testable demands towards the system. But to specify these testable and precise requirements is difficult, sometimes requirements represent architecture/design constraints or lofty goals that can neither be met by the system nor sufficiently tested.

Recent efforts in the Usability Community aim to produce similar guidelines tailored to Usability-related requirements. One result of these efforts is the *Common Industry Specification for Usability-Requirements (CISU-R)* [5] that provides guidelines for “defining usability requirements in sufficient detail to make an effective contribution to design and development and [create] usability criteria that can be empirically validated subsequently if needed.” It is closely related to the *Common Industry Format for Usability Test Reports (CIF)* [14] that offers guidance and a specification format for performing and describing the results of summative Usability Tests based on Usability requirements. These tests following the CIF can be used to validate the specified requirements written in the CISU-R notation. Both standards aim to specify the level of Usability based on its three dimensions: Effectiveness, Efficiency and Satisfaction: „The CIF and CISU-R take a broad approach to Usability based on DIN EN ISO 9241 Part 11 [...] The value of specifying these high level requirements is that they relate closely to business requirements for successful use of a product and increased productivity” [5].

From an RE point of view, it is therefore important that the quality attributes defined in the IEEE 830-1998 are applicable to any Usability-related requirement – in order to guarantee that they can be verified and tested throughout the development process.

## 1.2 User Centered Design Process

In current literature several User Centered Design (UCD) processes are described (e.g. [6], [16]). They focus on different aspects and address different needs of software development – but all of them share a set of common properties. These properties are described in the DIN EN ISO 13407, where a generalized human-centered design process for interactive systems is described [8]. The process is not a replacement to established software development processes but an addition to them: “This International Standard does not assume any one standard design process, nor does it cover all the different activities necessary to ensure effective system design. It is complementary to existing design methods and provides a human-centered perspective that can be integrated on different forms of design process in a way that is appropriate to the particular context.”

The process consists of four base activities common to UCD process models:

- To understand and specify the context of use,
- To specify the user and organizational requirements,
- To produce design solutions and
- To evaluate the designs against requirements.

These four activities are performed iteratively during the development process. The process is complete when the resulting system meets its specified requirements. Iterations to close in on the requirements and testing the results throughout the process are also common traits of this process model.

Even though the DIN EN ISO 13407 requests, that evaluation of mock-ups and prototypes take place throughout the process, few details are given on how this relates to the basic evaluation phase of the process. Because the results are iteratively improved, it seems reasonable to suggest that the intermediate results as well as the requirements need to increase in detail. Not only that new requirements will be discovered during the process, more specific requirements are needed to evaluate the intermediate process results while the system design becomes more concrete and precise.

This is in line with the activities described in the DIN IEC TR 18529, a standard that is an addition to the DIN EN ISO 13407, providing process descriptions to the human-centered lifecycle [9]. In the *Evaluate designs against requirements* sub-phase evaluation activities are proposed to improve the design, to define and to validate requirements and to check whether the defined practices are being followed.

Thus, results of the UCD phases serve as input for subsequent phases as well as validation criteria for intermediate and final UCD results.

### 1.3 Evaluation

Since the DIN EN ISO 13407 describes an iterative process model, all of its phases are performed several times during one development cycle. As Woletz [17] pointed out, evaluation activities focus on different aspects during this cycle – in early phases intermediate results are evaluated to identify weak points, gaps or errors for further improvement, while at the end a final assessment of the system is being performed.

Therefore evaluation activities should not only be performed once at the end of the cycle but several times during the development cycle. “In the general software industry it is increasingly recognized that continued evaluation is needed throughout the system development lifecycle, from early design to summative testing, in order to ensure final products meet expectations of designers, users, and organizations” [15].

Two types of goals and corresponding evaluation methods can be differentiated for validation activities: First, it can be evaluated whether the resulting documents of the development phases are correct in terms of content and style and capture the appropriate information needed for the later phases of the cycle. This can be done via interviews where the captured information are discussed with user representatives, stakeholders or other domain experts. In addition to this, “upward validations” are needed to check whether the results of a phase correspond to the results of previous phases. As an example, all generated requirements of the *Specify Requirement* phase are checked against the results of the *Context of Use* phase to find open loops, conflicts or mismatches between them.

Second it can be evaluated whether the system (in development) matches the requirements that were specified in advance. For UCD evaluations this can be done through Usability Tests or Expert Reviews. To be effectively able to do this, the specified requirements need to be measurable and precise, as stated above.

For both evaluation types the activities and the granularity of the resulting artifacts of the phases vary greatly. Therefore different evaluation methods need to be applied to the results. For example, analysis results from the *Context of Use* phase require different evaluation methods than software prototypes from the *Design Solution* phase.

Results of the *Context of Use* phase, such as persona descriptions or other models describing the work, the context and the users in focus can be validated with Usability Tests or through Expert Reviews. To check how well the system is accepted by the target audience, questionnaires or surveys can be conducted. The workflow descriptions resulting from the *Specify Requirement* phase can best be evaluated with real or potential users by comparing their real workflows with the proposed ones and gathering their feedback on the modifications. The same applies to early sketches showing the envisioned interaction steps the user will have to take. Design solutions, such as screens showing the user interface or detailed system interactions showing the flow of information, can a) be evaluated with users to see if the solutions supports their workflows and can b) be tested against styleguides or measurable criteria to see if the solution meets the previously specified expectations and requirements.

As mentioned before, not just final result but also intermediate results from the different phases from the development cycle need to be evaluated. To effectively evaluate the system against the results of the previous phases, they need to contain measurable criteria. Thus, in order to be able to evaluate UCD results there should be

requirements generated from each of the first three phases described in the DIN EN ISO 13407. This is what we do in the following sections.

## 2 A User Centered Requirement Framework

Given that on one side each of the first three phases described in the DIN EN ISO 13407 generates some type of result, and on the other the fact that the DIN EN ISO 9241 Parts 10 to 12 also describes three facets: Usability, Workflow and Design, the RE framework presented in this paper focuses on these three types of results as starting points.

Within the *Context of Use* phase, the analysis revolves mainly around the anticipated user, their jobs and tasks, their mental models, conceptions of the usage of the system, physical environment, organizational constraints and determinants and the like. While a lot of this information is descriptive (i.e. helps to create concepts and models rather than being a testable requirement), the Context of Use phase also yields users' expectation regarding the fundamental Usability Dimension, namely effectiveness, efficiency and satisfaction. CISU-R reflects these overarching metrics pertaining to the use and perception of the system by the user in the form of requirements, whereas CIF presents a standard format for presenting results of tests based on Usability requirements. These requirements can be used as evaluation criteria for the system and any intermediate prototypes, as well as intermediate UCD outcomes of subsequent phases.

The *User Requirements* phase focuses on individual workflows and tasks to be performed by one of the target users. Taking into consideration some of the *Context of Use* data, specific task performance models are elicited from users, workflow optimization happens and an improved task performance model is generated. The outcome of this phase could be described as a set of requirements pertaining to a user's interaction with the system in the context of a specific workflow or task, e.g. as use case scenarios. The requirements describe the discrete sub-steps of a user's interaction flow and the expected system behavior for each of these steps.

These requirements themselves can already be evaluated against the Usability requirements elicited from the context of use, e.g. by comparing an optimized workflow to the current state of workflow performance with regard to effectiveness, efficiency and user satisfaction. However, they also serve as input and evaluation criteria for the subsequent process step: *Produce Design Solution*.

During the *Produce Design Solution* phase, properties of the intended system are defined, e.g. information architecture, interaction flow, screen layout, component design, etc. Some of these properties are more conceptual, i.e. they serve as underlying models, but others can be experienced by the user during the use of the system. In order to translate these designs into solutions, different facets have to be considered.

- Conceptual/Structural/Framework type requirements that describe a model that is more underlying and less visible.
- Visual requirements (or other perceptive modality) that describe the physical properties of the solution (e.g. size, color, spacing, contrast, alignment, etc.)

- Interaction requirements that describe the behavior of the system (e.g. interaction flow, messaging, etc.)

These User Interface (UI) requirements can be evaluated against the Usability requirements generated in the *Context of Use* phase, i.e. to evaluate if the layout and interaction model fit the mental model that the users have of the task and associated information and if the general usage would be effective, efficient and satisfactory. They can also be evaluated against the Workflow Requirements from the *Specify Requirements*, in order to assess whether all specific workflows can be performed easily with the given concept and design. They serve as criteria for the actual system that has been developed, i.e. does it follow the defined model for layout and interaction

In summary, the user related requirements, which can be generated in the first three phases of the DIN EN ISO 13407 model, are depicted in Figure 1.

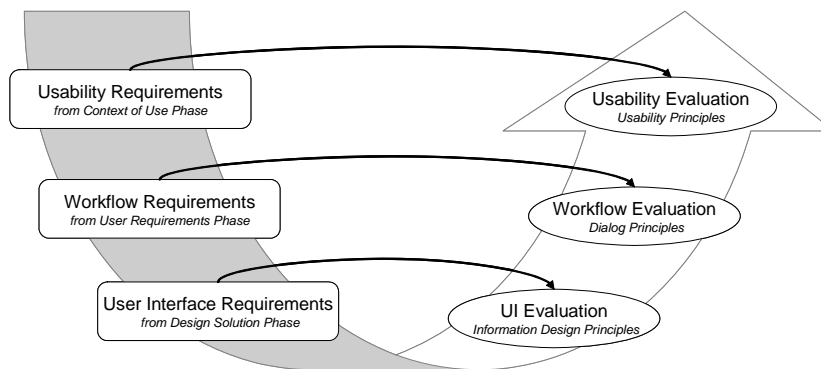


Figure 1: Usability-related requirements, their origin during the DIN EN ISO 13407 phases and their evaluation activities based on DIN EN ISO 9241 principles.

For the overall system design, it shall be noted however, that these three types of Usability-related requirements are not exhaustive in the description of a system. There is a variety of requirements stemming from different stakeholder groups that need to be viewed in conjunction with the Usability-related requirements described in this paper. But from a UCD perspective it is important to include user needs in the form of requirements into the overall process for scoping, implementation or testing purposes.

## 2. 1 Usability Requirements

This type of requirement contains general Usability criteria that the system must meet. It is based on the three Usability dimensions effectiveness, efficiency and satisfaction described in DIN EN ISO 9241 Part 11 and specifies the requirements of the user population towards the holistic usage of the system in the specified context of use. Thus, they are the result of the *Context of Use* phase in DIN EN ISO 13407 and provide measures that the system *should* comply to.

Usability requirements are derived from the results of the context analysis and from competitive analyses, previously identified areas of improvement, and from

general expertise about the domain, human-computer interaction practices, etc. For example, UCD artifacts to be used as input can be the Contextual Design Models as specified by Beyer and Holtzblatt [2] (Flow of Work, Sequence of Work, Work Artifact, Work Culture and Physical model) or Persona Descriptions as introduced by Cooper [6], since they describe user archetypes capturing user characteristics and describing overarching contextual and task information.

As usability requirements are one of many non-functional requirement types (as mentioned above), they have strong impact on functional requirements, since the usability goals address how the system shall be used and perceived by the user. They can be documented using established SRS formats or using the best practices defined in the CISU-R (see [12], [5]).

Usability Requirements can be verified through interviews and reviews during their elicitation, and tested via Usability Tests, Expert Reviews or with surveys and questionnaires towards the end of the development cycle. They can be used to generate unique selling points or other arguments used by marketing or sales divisions. The requirements can be tested by the testing department in conjunction with the other non-functional requirements.

## 2.2 Workflow Requirements

During the *Specify Requirement* phase of DIN EN ISO 13407 the current state captured during the *Context of Use* phase is considered and iteratively modified towards an optimized set of workflows and requirements that reflect the planned changes and improvements. The new workflow descriptions capture user goals and the associated workflows. They specify how the system *should* support the user to complete his tasks/goals with the system. They encapsulate essential interaction steps, needed information and options for the user and thus specify system behavior without being too specific about the concrete details (UI). Entire Workflow Scenarios are considered as one requirement, meaning that the improved workflow to accomplish a task needs to be implemented in the system to allow the users to work successfully.

The information and models describing the future system's context of use serve as input as well as innovations, the anticipated changes, fixes to known bugs and new features which are molded into a description of a "better" system than the current.

The possible interaction with the system and the flow of information is described in use cases or similar artifacts (e.g. scenario descriptions). "A scenario is a concrete description of a specific flow of interaction, but one that is chosen to be typical or representative. [...] A use case is a generic scenario, describing one kind of interaction with a particular user interface." [4] Cockburn defines, that each use case is the implementation of a stakeholder's goal. Multiple stakeholders participate in the use case and their interests should also be protected by the interactions defined in it. For Cockburn, the collection of use cases is the essence of the system's requirements, even though they don't represent all the requirements [3].

Workflow Requirements can be reviewed and verified with users to assess whether the current workflow description is accurate and if the optimized workflows are regarded as an actual improvement of their tasks. Later in the development process

the requirements can be used to evaluate a prototype or system with regard to the workflow support, i.e. whether the interaction model meets the workflow requirements. A single scenario, e.g. the main scenario from a use case, could for example serve as an evaluation scenario during a usability test.

The concrete descriptions of the system and the needed functionality within workflows can be used to determine what to build into the system. The workflow descriptions are used as test cases to verify if the workflows can be realized with the prototypes and the final system created during the *Design Solutions* phase.

### 2.3 UI Requirements

During the *Produce Design Solution* phase, both the Usability Requirements and the Workflow Requirements are synthesized into a set of conceptual models and solution elements. These usually consist of Information Architecture, Navigation Models, Wireframes, Sketches, high fidelity screen designs, or prototypes. They describe the logical model and physical properties of the system, i.e. specify how the widgets, UI patterns, and interaction elements *shall* look, behave, and interact. Ideally, they also provide all states, presets, and system reactions for concrete system screens.

UI requirements are generated from prototypes, the actual system in the current stage, UI specification documents and also from general styleguides, information architecture or navigation models. They focus on different layers of the user interaction with the system and thus have sub-types:

- Information architecture and information flow requirements which define the overarching logical structure of the user interface.
- Presentation requirements, where the layout of an entire component or a single element is defined, e.g. a screen, a widget like a dropdown combo box.
- User-system-interaction requirements, where the behavior of the UI elements is defined, e.g. status changes.
- Compound requirements, where the interaction between more than one element is specified, e.g. in the case of drag and drop functionality.
- Message requirements, which define when and how system generated notifications, e.g. errors or alerts, are presented to the user.

In order to ensure that the solution is conform with the previous analysis, the UI requirements are evaluated against usability and workflow requirements to ensure that they match the mental model that the user has of the “look & feel” of the system, as well as the users expectations regarding effectiveness and efficiency on a granular level, e.g. with regard to screen flips or mouse clicks. Additionally, they should comply with general design guidelines as defined e.g. in DIN EN ISO 9241 Part 12, or styleguides. UI requirements are also used to evaluate the fulfillment of the solution design in the coded system.

Within the development process, UI requirements can be used by Development as input for their system design, as well as by testing divisions for ensuring appropriate

realization of the UI. They enhance/extend workflow requirements in a way that development can select the most appropriate implementation of a given workflow.

UI Requirements are also used by internal experts (e.g. UI design, testing) to determine if the (coded) system complies with its specification. They are mostly of internal value, i.e. supporting system specification and verification. From an end user's perspective they should almost be invisible, i.e. the users' perception of the system should not recognize individual UI features, but a holistic experience that is compliant with their expectations.

### **3 Summary and Outlook**

The authors introduced a framework of Usability-related requirements to align Usability Engineering with Software Engineering and Requirement Engineering practices. Correlated to the phases of the DIN EN ISO 13407 process model and to the three levels of usability principles laid out in DIN EN ISO 9241 Parts 10 to 12, three requirement types were differentiated: Usability-, Workflow- and UI Requirements. Through the introduction of Workflow- and UI Requirements the authors extended the established concept of usability requirements, e.g. as described in CISU-R. These two additional types offer a more precise representation of UCD demands generated in the later phases of a development cycle. The three requirement types were explained, their interconnections described and details to their application and reuse were given.

By differentiating the three types of requirements, development organizations can ensure a seamless, traceable hierarchy of user-focused requirements, starting with the high level needs pertaining to the general use of a system, going through the specific needs during the task performance towards requirements regarding the specific implementation of the system.

In addition, the approach allows selecting the most appropriate evaluation method for different types of requirements, e.g. summative usability tests for Usability Requirements, or expert based reviews for UI Requirements. Guidance in selecting the appropriate evaluation method is provided by a poster by Freymann et al. [11] that differentiates evaluation methods by their result types and by their application within the development life cycle.

Another benefit is scalability. Individual requirements are less monolithic than e.g. complete specification documents. Especially with regard to the emerging lightweight agile approaches to SE, e.g. Extreme Programming or Scrum, a less document driven approach to capture UCD outcomes could be more suitable. A detailed analysis of potential application areas for the three UCD requirement types in agile development is described by DÜchting et al. [10].

The approach and the requirement framework presented in this paper need to be applied in the field and tried out. The requirement types need to be integrated within established requirement engineering approaches to evaluate whether the proposed granularity and differentiation of the requirement types is sufficient and where areas of improvement and refinement can be identified.

## References

1. ANSI/IEEE Std. 729-1983. Standard Glossary of Software Engineering terminology, 1983.
2. H. Beyer and K. Holtzblatt: Contextual Design – Defining Customer-Centered Systems, Morgan Kaufmann, 1998.
3. A. Cockburn: Writing Effective Use Cases, Addison-Wesley, 2001.
4. L. Constantine: What do users want? Engineering Usability into Software, Windows Tech Journal 4, 12, 30–39, 1995.
5. CISU-R. Common Industry Specification for Usability-Requirements, Draft Version 0.86, National Institute of Standards and Technology, 18.03.2006.
6. A. Cooper and R. Reimann: About Face 2.0. Wiley, Indianapolis, 2003.
7. DIN EN ISO 9241. Ergonomic requirements for office work with visual display terminals (VDTs). International Organization for Standardization, 1998.
8. DIN EN ISO 13407. Human-centered design processes for interactive systems, International Organization for Standardization, 1999.
9. DIN IEC TR 18529. Software engineering - Product quality. International Organization for Standardization/International Electrotechnical Commission, 2001.
10. M. DÜchting, D. Zimmermann, K. Nebe: Incorporating User Centered Requirement Engineering in Agile Software Development. In preparation, HCII 2007, Beijing, 2007.
11. M. Freymann, L. Grötzbach, K. Nebe: Selecting Appropriate Evaluation Methods for Different UCD Outcomes. In preparation, HCII 2007, Beijing, 2007.
12. IEEE STD 830-1998. IEEE recommended practice for software requirements specifications, International Organization for Standardization, 1998.
13. ISO/IEC 9126. Software Engineering - Product Quality, International Organization for Standardization, 2001.
14. ISO/IEC 25062:2006. Software product Quality Requirements and Evaluation (SQuARE) - Common Industry Format (CIF) for usability test reports, 2006.
15. A. W. Kushniruk and V.L. Patel, Cognitive and Usability Engineering Methods for the Evaluation of Clinical Information Systems, Journal of Biomedical Informatics 37, 2004.
16. D. J. Mayhew: The Usability Engineering Lifecycle – A practitioner’s Handbook to User Interface Design, Morgan Kaufmann, 1999.
17. Woletz, N.: Evaluation eines User-Centred Design-Prozessassessments - Empirische Untersuchung der Qualität und Gebrauchstauglichkeit im praktischen Einsatz. Doctoral Thesis. University of Paderborn, Germany, 2006.