Aus dem
Institut für Multimediale und Interaktive Systeme
der Universität zu Lübeck

Direktor: Prof. Dr. rer. nat. Michael Herczeg

Systemgestützte Integration des Usability-Engineerings in den Software-Entwicklungsprozess

Inauguraldissertation

zur Erlangung der Doktorwürde der

Universität zu Lübeck

- Aus der Sektion Informatik/ Technik -

vorgelegt von

Dipl.-Inf. Marc Oliver Paul (geb. Kammler)

aus Frankfurt am Main

Lübeck 2014

1. Berichterstatter: Prof. Dr. rer. nat. Michael Herczeg

2. Berichterstatter: Prof. Dr.-Ing. habil. Peter Forbrig

Tag der mündlichen Prüfung: 29.04.2015

Zum Druck genehmigt.

Lübeck, den 30.04.2015

„Wissenschaft ist der Versuch, der chaotischen Mannigfaltigkeit der Sinneserlebnisse ein logisch einheitliches, gedankliches System zuzuordnen.“

(Albert Einstein, 1879-1955)

Kurzfassung

Die Zusammenfassung der beiden Bereiche Usability-Engineering (UE) und Software-Engineering (SE), mit dem Ziel einen systematisch planbaren Entwicklungsprozess gewährleisten zu können, stellt eine besondere Herausforderung für Softwareentwickler dar. Die vorliegende Arbeit stellt sich dieser Aufgabe und beschreibt die Analyse und Entwicklung eines Systems, welches Verfahren und Methoden aus dem UE unterstützt und sich dabei in bestehende SE-Prozesse integrieren lässt.

Dieses im Rahmen der Arbeit entstandene System „Usability-Engineering-Repository“ (UsER) wurde in einem dreijährigen Kooperationsprojekt zwischen dem Institut für Multimediale und Interaktive Systeme (IMIS) der Universität zu Lübeck und einem Lübecker Software- und Beratungshaus entwickelt. Finanziell gefördert wurde das Projekt über Mittel der Europäischen Union (EU) und des Landes Schleswig Holstein mit dem Ziel, Methoden des UEs auf die Belange von Softwareunternehmen zuzuschneiden und sie mit variabel praktizierbaren SE-Prozessen zu verknüpfen. Es sollten kundenorientierte Konzeptions-, Entwicklungs-, Test- und Einführungsprozesse sowie Systemarchitekturen definiert und in dem Unternehmen etabliert werden. Gemeinsam mit dem Software- und Beratungshaus als Kooperationspartner wurden Ansätze aus dem Bereich des UE eruiert, erprobt, bewertet und anschließend in eine Werkzeugunterstützung überführt. Die bisher identifizierten und umgesetzten Werkzeugunterstützungen wurden in Form von beliebig kombinierbaren Einzelmodulen realisiert. Die Ergebnisse dieser Arbeiten sind als Werkzeugmodule mit in das entwickelte System UsER eingeflossen. Die Module ermöglichen die Erfassung strukturierter, entwicklungsrelevanter Informationen und deren semantischer Verknüpfungen. Dies ermöglicht eine flexible Anpassbarkeit an die jeweiligen Projektgegebenheiten einer Softwareentwicklung, und das System ist somit für unterschiedliche Projekte einsetzbar. Es wurde so konzipiert, dass sich weitere methodische Werkzeugmodule integrieren lassen. Dadurch ergibt sich eine Vielzahl unterschiedlicher Verfahren, die eine flexible und problemgerechte Integration des UEs in einen SE-Prozess ermöglichen.

Die aus der vorliegenden Arbeit gewonnenen Erkenntnisse und das entwickelte System UsER werden inzwischen sowohl im universitären Umfeld als auch von dem Industriepartner eingesetzt und weiter verfeinert.

Inhaltsverzeichnis

Kurzfassung ... III
Inhaltsverzeichnis ... IV
1 Einleitung ... 1
1.1 Motivation ... 1
1.2 Ziel der Arbeit ... 2
1.3 Struktur der Arbeit ... 2
2 Usability-Engineering ... 4
2.1 Einordnung des Gebiets ... 5
2.2 UE-Prozesse ... 6
2.2.1 Human-centered Design ... 6
2.2.2 Usability Engineering Lifecycle ... 10
2.2.3 Contextual Design ... 13
2.2.4 Scenario-based Design ... 19
2.2.5 Activity-centered Design ... 21
2.2.6 Usage-centered Design ... 22
2.2.7 Usability Engineering Lifecycle (Mayhew) ... 25
2.3 Methoden des UE ... 27
2.3.1 Analyse ... 27
2.3.2 Spezifikation ... 43
2.3.3 Entwicklung ... 44
2.3.4 Evaluation ... 51
2.4 Ansätze und Werkzeuge des UE ... 55
2.4.1 CREWS ... 55
2.4.2 REUSE ... 56
2.4.3 Usability Planner ... 57
2.4.4 CTTE ... 58
2.4.5 UsabilityTools ... 59
2.5 Fazit ... 60
3 Software-Engineering ... 63
3.1 Prozesse ... 63
3.1.1 Wasserfall ... 63
3.1.2 V-Modell ... 64
3.1.3 Spiralmodell ... 64
3.1.4 Rational Unified Process (RUP) ... 65
3.1.5 Agile Prozesse und Scrum ... 67

3.2 Methoden des SE ... 70
3.2.1 Analyse ... 71
3.2.2 Design ... 74
3.2.3 Implementierung ... 76
3.2.4 Test und Inbetriebnahme ... 77
3.3 Integrationsansätze von UE in das SE ... 77
3.3.1 Goal-Scenario-Coupling ... 77
3.3.2 Agile User Centered Design Process ... 79
3.4 Fazit ... 81
4 Eigener Ansatz zur Integration von UE und SE ... 82
4.1 Das KoSSE-Verbundprojekt ... 82
4.2 Der Kooperationspartner ... 83
4.3 Die Ist-Analyse: Der SE-Prozess des Kooperationspartners ... 84
4.3.1 Anforderungsaufnahme ... 85
4.3.2 Anforderungsanalyse ... 86
4.3.3 Grobkonzeption ... 87
4.3.4 Feinkonzeption ... 87
4.3.5 Entwicklung ... 87
4.3.6 Qualitätssicherung ... 88
4.3.7 Einführung ... 89
4.3.8 Erkannte Schwachstellen des bisherigen SE-Prozesses ... 89
4.4 Entwicklung eines idealisierten Entwicklungsprozesses ... 89
4.4.1 Formulierung von Usability Zielen ... 90
4.4.2 Dekomposition von Aufgaben ... 90
4.4.3 Formulieren von Aufgaben und Szenarien ... 90
4.4.4 Nutzen von Storyboards ... 92
4.4.5 Formulierung von Aktivitäten ... 92
4.4.6 Analyse der Organisation ... 95
4.4.7 Analyse der Benutzer- und Persona-Erstellung ... 96
4.4.8 Herstellen von Mockups und Prototypen ... 97
4.4.9 Durchführung von formativen Evaluationen ... 98
4.4.10 Summative Evaluation ... 98
4.4.11 Der idealisierte UE-Prozesses ... 99
4.5 Konzeption und Entwicklung des UE-Repository ... 101
4.5.1 Ableitung der Nutzungsanforderungen aus der Kontextanalyse ... 102
4.5.2 Grobkonzeption des UE-Repositories ... 102

4.5.3 Projektverwaltung ... 103
4.5.4 Projektansicht ... 104
4.5.5 Modulansicht ... 105
4.5.6 Evaluation der vorgestellten Konzepte ... 106
4.5.7 Auswahl der Frameworks ... 106
4.5.8 Architektur ... 107
4.5.9 Datenmodell ... 109
5 Realisierung des UE-Werkzeugs UsER ... 111
5.1 Planung des Human-centered Design-Prozesses ... 113
5.1.1 Projektverwaltung ... 113
5.1.2 Projektansicht ... 114
5.2 Verstehen und Dokumentieren des Anwendungskontextes ... 115
5.2.1 Benutzer-Analyse ... 115
5.2.2 Aufgaben-Analyse ... 120
5.2.3 Artefakt-Analyse ... 124
5.2.4 Organisations-Analyse ... 125
5.3 Spezifizieren der Anforderungen ... 127
5.3.1 Anforderungsmanagement ... 127
5.4 Verfeinern der Anforderungen und Entwurf von Lösungen ... 128
5.4.1 Szenario-Modul ... 129
5.4.2 Mockup-Editor ... 130
5.4.3 Verfeinern der Anforderungen ... 131
5.5 Evaluieren und Bewerten von Lösungen ... 134
5.5.1 Kommentare und Diskussion ... 134
5.5.2 Evaluationsmodul ... 136
6 Zusammenfassung und Ausblick ... 143
Abbildungsverzeichnis ... VII
Tabellenverzeichnis ... X
Literaturverzeichnis ... XI
Normen und Standards ... XIX
Publikationen ... XX
Danksagung ... XXI
Ehrenwörtliche Erklärung ... XXII

1 Einleitung

Die ansteigende Anzahl an Softwarelösungen führt über den Wettbewerbsmarkt dazu, dass neben der Qualität auch weitere Eigenschaften der Software zunehmend bedeutendere Rollen spielen. Dies erhöht den Druck auf softwareentwickelnde Unternehmen enorm. Es reicht heute nicht mehr aus, eine funktionale und zuverlässige Software bereit zu stellen, um alle Qualitätskriterien eines Softwareproduktes zu erfüllen und am Markt erfolgreich zu sein. Software muss darüber hinaus nach der ISO/IEC 25010:2011 ebenso portabel, wartungsfähig, sicher, kompatibel, performant und vor allem gebrauchstauglich sein. Für Softwarehersteller bedeutet dies, dass sie sich nicht mehr nur mit den funktionalen Eigenschaften der Software, der stetig steigenden Komplexität des Produktes und den sich stetig ändernden Technologien auseinandersetzen sollten. Sie müssen sich ebenso mit Fragen der Gebrauchstauglichkeit befassen und sollten beachten, wie spezifische Benutzer in bestimmten Kontexten zukünftig mit dem System effektiv, effizient und zufriedenstellend deren Ziele erreichen können, so wie es von der ISO 9241-230:2009 beschrieben wird. Die fachliche Disziplin, die sich mit der systematischen Erstellung gebrauchstauglicher Systeme beschäftigt, ist das *Usability-Engineering (UE)*. Das UE wiederum ist eine Teildisziplin der Mensch-Computer-Interaktion[1] aus der eine Vielzahl unterschiedlicher Richtlinien, Standards, Verfahren und Methoden entwickelt wurden. Durch sie fließen psychologische, arbeits- und kognitionswissenschaftliche als auch ergonomische, soziologische und designorientierte Aspekte in das UE ein. Jedoch ist folgender Aussage eine besondere Bedeutung beizumessen: „Das Usability-Engineering allein kann nicht ohne das Software-Engineering existieren“ (Nebe, Leuchter & Beck, 2009). Bisherige Herangehensweisen bei der Softwareentwicklung zogen immer nur eine der beiden Disziplinen heran, statt sich auf beide zu stützen.

Die vorliegende Arbeit erforscht Ansätze, die beiden Disziplinen auf effiziente Weise miteinander zu verbinden. Es werden Wege untersucht, um das UE als integralen Bestandteil des *Software-Engineering (SE)* zu etablieren. Zu diesem Zweck wurde im Rahmen dieser Arbeit gemeinsam mit einem mittelständischen Software- und Beratungshaus in der Softwareentwicklung nach Möglichkeiten geforscht, UE-Methoden und UE-Verfahren werkzeugunterstützt in den Entwicklungsprozess zu etablieren.

1.1 Motivation

Laut der Standish Group hatten trotz enormer Anstrengungen über die Hälfte aller Softwareprojekte im Jahre 2012 am Ende ihrer Laufzeit erhebliche Schwierigkeiten oder sind ganz gescheitert. Bei wasserfallähnlichen[2] Entwicklungsprozessen waren es sogar 86 % der Projekte, die nicht erfolgreich abgeschlossen werden konnten (Cohn, 2012). Als Begründung des Ergebnisses werden wiederholt unzureichende Benutzerbeteiligung, unvollständige oder sich ändernde Anforderungen und unrealistische Erwartungen genannt (Ebert, 2010; Hull, Jackson & Dick, 2005). Daraus resultierte in den vergangenen Jahrzehnten die Entwicklung einer Reihe unterschiedlicher Prozessmodelle. Sie werden von Unternehmen benötigt, um die Herstellung eines Produktes systematisch planen und koordinieren zu können. Hierbei gilt es insbesondere, die Zeitvorgabe und die sich daraus ergebenden Kosten einzuhalten (Partsch, 2010). Ebenfalls soll die gleichbleibend gute Qualität des Endproduktes durch diese Modelle gewährleistet werden können. Stringente und sukzessiv verlaufende Prozessphasen bedeuten, dass ein neuer Prozessschritt erst begonnen werden kann, wenn der vorhergehende Schritt abgeschlossen wurde. Dieses Verfahren lässt wenig Spielraum, um auf Probleme oder Fehler in vorhergehenden Schritten zu reagieren.

Erfolgsversprechender seien laut CHAOS Manifesto der Standish Group agile Entwicklungsprozesse (Cohn, 2012). Sie zeichnen sich im Gegensatz zu klassischen[3] Entwicklungsprozessen dadurch aus, schlanker zu sein und

[1] im englischen als Human-Computer-Interaction (HCI) bezeichnet

[2] einer der ersten strukturierten Prozessmodelle in der Softwareentwicklung (siehe Kapitel 3.1.1)

[3] Im Zusammenhang mit agiler Softwareentwicklung werden alle nicht agilen Prozesse als „schwergewichtige“ oder „klassische“ Prozessmodelle bezeichnet (Hanser, 2010)

flexibler auf Änderungen reagieren zu können, sowie iterativ und inkrementell vorzugehen. Dass zahlreiche namhafte Unternehmen wie Microsoft, Google oder Cisco agile Verfahren sehr erfolgreich einsetzen, spricht ebenfalls für dieses Vorgehen (Sutherland, 2010).

Nach Metzker und Offergeld (2001) haben die verschiedenen Verfahren aus dem Bereich des UE sich in zahlreichen Projekten erfolgreich beweisen können, wie beispielsweise der Usability-Engineering-Lifecycle von Mayhew (2010) oder Nielsen (1993), das Contextual Design von Beyer und Holtzblatt (1998) oder das Usage-centered Design von Constantine und Lockwood (1999). Sie verbessern sowohl nachhaltig die Gebrauchstauglichkeit eines interaktiven Systems als auch die Akzeptanz bei den Benutzern. Bis heute fehlen jedoch spezifische Kenntnisse, wie genau die Methoden und Verfahren des UE in etablierte Software-Entwicklungsprozesse integriert, bzw. miteinander kombiniert werden können. Mit dieser Frage beschäftigen sich derzeit eine Reihe von Forschungsarbeiten (Fischer, Nebe & Klompmaker, 2011; Zimmermann & Grötzbach, 2007). Ein entscheidender Schritt zur Schließung dieser Lücke sei es nach Metzker und Reiterer (2004), bestehende Entwicklungsprozesse mit Strategien und vor allem Werkzeugen aus dem Bereich des Mensch-Computer-Designs zu unterstützen.

1.2 Ziel der Arbeit

Das Ziel der vorliegenden Arbeit besteht in der Identifizierung von Verfahren und Methoden aus dem Bereich des UEs, die sich für eine Integration in das SE eignen. Ergebnisse dieser Untersuchungen sollen in ein flexibel kombinierbares neues Verfahren überführt werden, welches bisher getrennt ablaufende Prozesse aus den beiden Bereichen verbindet. Zudem soll ein System entwickelt werden, welches dieses kombinierte Verfahren durch ein an die Begebenheiten von Software entwickelnden Firmen anpassbares und kollaboratives Werkzeug unterstützen kann. Ein wesentlicher Punkt besteht dabei in der semantischen Verknüpfbarkeit der durch das UE erfassten Informationen. Eine der größten Herausforderungen besteht in der Konsolidierung der gewonnen Informationen (Benyon, 2010). Speziell in der Analysephase, in der es darum geht, zu verstehen von wem, wo und wie ein System zukünftig eingesetzt werden soll, fallen eine Reihe heterogener Informationen an. Das Zusammenführen dieser Einzelinformationen eines bestimmten Kontextes leitet die Designphase der Entwicklung ein. Bereits erhobene Informationen wieder zu verwenden und miteinander zu kombinieren ist eine der Herausforderungen, die durch die neue Werkzeugunterstützung erleichtert werden soll. Um die Teilergebnisse des UE-Prozesses in das auszuliefernde Produkt überführen zu können, sollten die Informationen in beliebige SE-Verfahren integriert werden können.

1.3 Struktur der Arbeit

Einen Überblick der verschiedenen Verfahren und Methoden aus dem Bereich UE gibt Kapitel 2. Die über die Jahre entstandenen Verfahren bzw. Prozesse, spiegeln die Vielzahl der unterschiedlichen Schwerpunkte in der Betrachtungs- und Vorgehensweise aus diesem Bereich wider, und auch die Gemeinsamkeiten werden hier verdeutlicht. Das UE greift Erkenntnisse aus verschiedenen Fachbereichen auf, um daraus systematische Verfahren abzuleiten, mit denen die Entwicklung gebrauchstauglicher Software gewährleistet werden soll. Eine Reihe dieser Prozesse werden in Kapitel 2.2 behandelt. Einige der in den UE-Prozessen verwendeten Methoden werden in Kapitel 2.3 konkretisiert. In Kapitel 2.3.4.3 werden zudem existierende Werkzeuge und Ansätze aus dem Bereich des UEs untersucht, die Erkenntnisse über das in dieser Arbeit zu entwickelnde System liefern konnten. Ebenfalls kann der Blick auf bereits existierende Werkzeugunterstützungen dazu genutzt werden, eventuelle Probleme im Vorweg zu umgehen.

Da sich die gewünschte Gebrauchstauglichkeit eines Systems am Ende auf das auszuliefernde Produkt auswirken soll, wäre es nach Sohaib und Khan (2010) nachlässig, den Prozess des UE unabhängig von der Implementierung der Software zu betrachten. Im schlimmsten Fall könnte dies dazu führen, dass Erkenntnisse aus dem UE ignoriert werden oder verloren gehen. Aus diesem Grund beleuchtet Kapitel 3.1 die unterschiedlichen Software-Entwicklungsprozesse. Neben den klassischen Vorgehensmodellen, wie dem Wasserfallmodell (siehe Kapitel 3.1.1) oder dem V-Modell (siehe Kapitel 3.1.2), wird stellvertretend für die agilen Prozesse das SCRUM-Verfahren (siehe Kapitel 3.1.5) näher beleuchtet. Insbesondere die Methoden des *Requirement-Engineerings (RE)* stellen ein geeignetes Mittel für die Kombination der beiden Bereiche UE und SE dar. Diese Teildisziplin des SE wird in Kapitel 3.2 untersucht.

Kapitel 4 beschreibt die vollständige Entwicklung der hier vorgestellten Arbeit und die Erfahrungen, die im Rahmen des von der EU geförderten Verbundprojektes in Kooperation mit dem Software- und Beratungshaus gesammelt wurden. Kapitel 4.1 gibt einen Überblick des Ablaufs und die Zielvorstellungen des Projektes. Nach der Analyse des bestehenden SE-Prozesses des Kooperationspartners in Kapitel 4.3 werden in Kapitel 4.4 die Erfahrungen bei der Integration verschiedener UE-Methoden in den bestehenden Prozess diskutiert. Daraus gewonnene Erkenntnisse führten anschließend zur Formulierung eines *idealisierten Entwicklungsprozesses*, der UE und SE miteinander verbindet. Um diesen idealisierten Entwicklungsprozess auch werkzeugseitig zu unterstützen, wurden einzelne Methoden selektiert, um sie anschließend auf die softwarebasierte Werkzeugunterstützung „Usability-Engineering-Repository“ (UsER) in Form des Idealisierten Entwicklungsprozesses abzubilden. Sowohl der Entstehungsprozess als auch Erkenntnisse aus einigen Fehlentscheidungen, die bei der iterativen Entwicklung des Systems gewonnen werden konnten, werden in Kapiteln 4.5 vorgestellt. Das System ist das Ergebnis einer Vielzahl iterativer Verfeinerungen und Erweiterungen, die in enger Kooperation zwischen dem Institut für Multimediale und Interaktive Systeme (IMIS) der Universität zu Lübeck und dem Software- und Beratungshaus und deren Tochterunternehmen entstanden ist. UsER vereint einen Großteil der in dieser Arbeit vorgestellten Prozesse und Methoden und zeigt somit ein praktisch anwendbares System, welches diese Ansätze werkzeugseitig unterstützen kann.

Zur Verdeutlichung der entstandenen Funktionen und Module des UsER-Systems, werden diese in Kapitel 5 anhand eines ebenfalls im Rahmen dieser Arbeit entstandenen „idealisierten Usability-Engineering-Nutzungsszenarien-Prozesses“ verdeutlicht. Das System unterstützt dabei einen Großteil der vorgestellten UE-Prozesse und kann darüber hinaus von allen an einer Software-Entwicklung beteiligten Stakeholdern[4] genutzt werden. Darüber hinaus werden neue Lösungswege aufgezeigt, wie die Ergebnisse aus den UE-Prozessen in bestehende SE-Prozesse überführt werden können.

Kapitel 6 beinhaltet ein zusammenfassendes Fazit und führt Punkte auf, die in zukünftigen wissenschaftlichen Arbeiten Gegenstand weiterer Untersuchungen werden könnten.

[4] Ein Stakeholder ist eine „Einzelperson oder Organisation, die ein Anrecht, einen Anteil, einen Anspruch oder ein Interesse auf ein bzw. an einem System oder an dessen Merkmalen hat, die ihren Erfordernissen und Erwartungen entsprechen.“ (ISO 9241-210:2011)

2 Usability-Engineering

Um herauszufinden, welche Prozesse und Methoden aus dem UE sich für den realen Einsatz von Entwicklungsprojekten für den Kooperationspartner eignen, mussten diese zunächst einmal identifiziert, verstanden und auf ihre Praxistauglichkeit hin untersucht werden. In den folgenden Beschreibungen der Prozesse und Methoden wurde darauf geachtet, die den Autoren wichtig erscheinende Aspekte sinngemäß widerzugeben.

Das UE ist keine einmalig anzuwendende Methode, die vor Fertigstellung einer Software auszuführen ist (Nielsen, 1993). UE ist vielmehr eine Menge aus Aktivitäten, die in den gesamten Entwicklungsprozess der Softwareherstellung eingebaut werden sollte. Vor allem die Phase vor den ersten Entwürfen der Mensch-Computer-Schnittstellen ist ausschlaggebend für den finalen Erfolg einer gebrauchstauglichen Anwendungssoftware. Dieser Erfolg definiert sich durch ein umfassendes Verständnis der Bedürfnisse eines Benutzers, seiner Arbeitsbedingungen, seiner angestrebten Ziele und der dabei auftretenden Probleme. Erst dadurch wird das Entwicklungsteam in die Lage versetzt, eine entsprechende Softwareunterstützung zu erstellen. Die Gebrauchstauglichkeit eines computerbasierten Systems kann als höchstes summarisches Kriterium für ein ergonomisches interaktives System angesehen werden (Herczeg, 2009). Diese definiert sich nach der ISO 9241-11 als das Ausmaß, in welchem ein Benutzer ein System in einem bestimmten Nutzungskontext zur Erfüllung seiner Ziele einsetzt (siehe Abbildung 1).

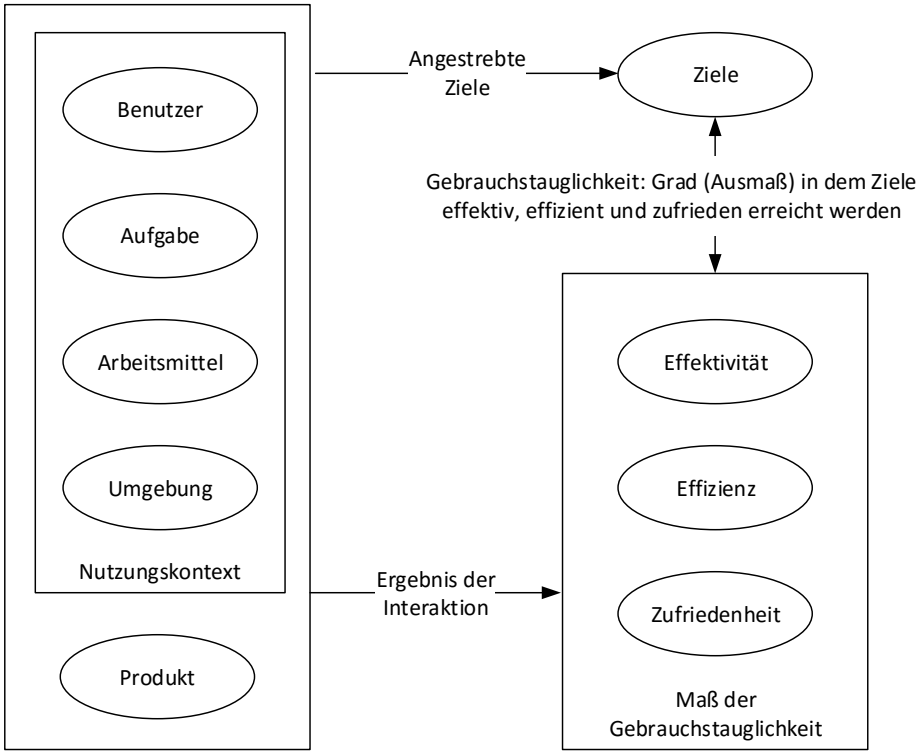


Abbildung 1: Elemente zur Bestimmung des Gebrauchstauglichkeitsgrades (nach ISO 9241-11)

Die Gebrauchstauglichkeit kann dadurch gemessen werden, wie effektiv, effizient und zufriedenstellend das System den Benutzer bei der Erreichung seiner Ziele unterstützt. Durch software-ergonomische Kriterien (siehe Kapitel 2.3.4) kann der Grad der Gebrauchstauglichkeit gemessen werden.

2.1 Einordnung des Gebiets

Das UE versteht sich als ein interdisziplinäres Gebiet, das mit einer Reihe weiterer Disziplinen in Beziehung steht und die Entwicklung von gebrauchstauglichen Systemen gewährleisten soll. Nachfolgende Grafik (siehe Abbildung 2) veranschaulicht die Querbezüge und Schnittmengen der verschiedenen Teildisziplinen des UEs.

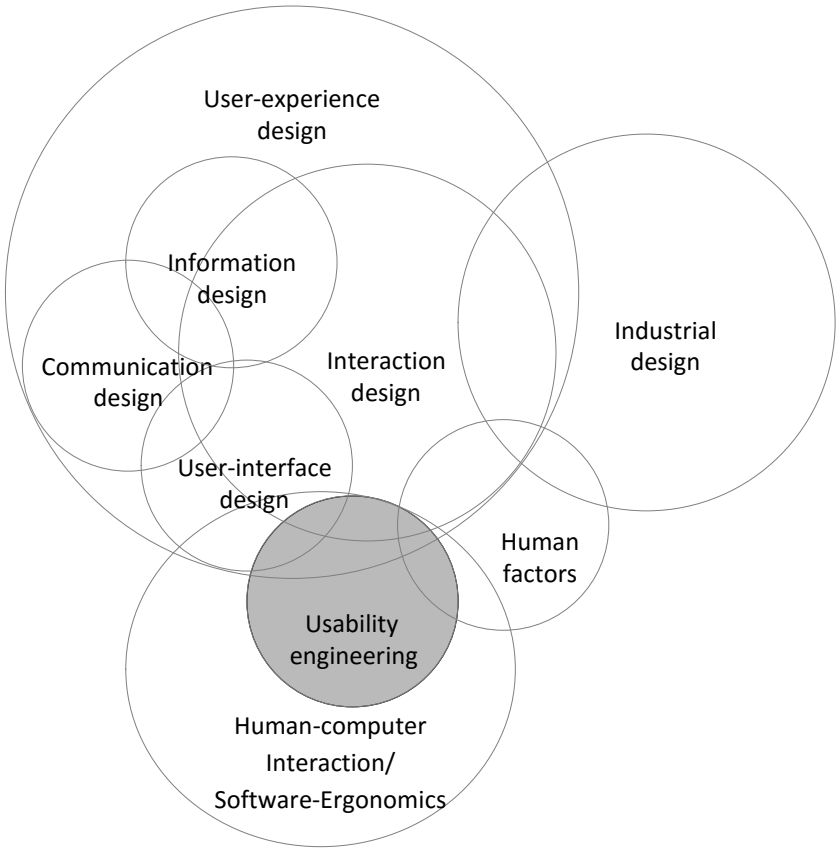


Abbildung 2: Überlappung von Disziplinen im UE (nach Saffer, 2007)

Saffer sieht das UE als eine Teilmenge der Human-Computer-Interaction (HCI) an, die im deutschsprachigen Raum auch als Mensch-Maschine-Interaktion (Saffer, 2007) oder als Software-Ergonomie (Herczeg, 2009) bezeichnet wird. Nach Sutcliffe sei es die Kernaufgabe von HCI, Erkenntnisse unterschiedlicher Wissenschaften, wie beispielsweise aus der Psychologie und der Soziologie aufzugreifen, um sie effektiv in Entwicklungsprozessen von Systemen anwenden zu können (Sutcliffe, 2000, S. 198). Dadurch solle erreicht werden, Designentscheidungen nicht mehr nur der Kreativität von Entwicklern zu überlassen. Vielmehr solle das Design des Systems auf einem tiefgehenden Verständnis des Benutzers basieren. Vor allem das Verständnis der mentalen Modelle von Benutzern sei ein wichtiger Aspekt. Der bisherige Einfluss der HCI auf die Softwareentwicklung sei nach Sutcliffe (2000) jedoch zu gering ausgeprägt.

Nach Herczeg (2009) wurden in der HCI bzw. der Software-Ergonomie Theorien und Methoden „[…] für Analyse, Modellierung, Gestaltung und Evaluation erarbeitet, die dabei unterstützen sollen, computerbasierte Werkzeuge in benutzer- und anwendungsgerechte Weise zu konzipieren, zu realisieren und zu testen.“ (Herczeg, 2009). Sie ermöglicht, die Entwicklung gebrauchstauglicher computerbasierter Werkzeuge.

Als Erweiterung der HCI könne laut IBM das *User-Experience Design* gesehen werden. In der klassischen HCI steht primär die Interaktion zwischen Mensch und Computer im Fokus. Beim User-Experience Design müssen alle Erfahrungen eines Benutzers mit dem Produkt berücksichtigt werden. Dies erstreckt sich von der ersten Wahrnehmung über die Anschaffung bis hin zur Entsorgung des Gegenstandes, quasi über den Zeitraum der gesamten Lebensdauer eines Produktes. (IBM Corporation, 2013)

Eine weitere Disziplin, die sehr eng mit der HCI und dem UE in Beziehung steht, ist das *Interaktionsdesign*. Sie beschäftigt sich vor allem „[…] mit den konzeptionellen, konstruktiven und gestalterischen Fragen der Realisierung benutzer- und anwendungsgerechter interaktiver Computersysteme“ (Herczeg, 2006). Nach Saffer (2007) unterscheide sich HCI speziell in einem Punkt zum Interaction Design. Dieses setze sich, im Gegensatz zur HCI, vor allem mit der zwischenmenschlichen Interaktion auseinander. Nach Preece gehe es vor allem um die Erstellung einer positiven User-Experience, die Arbeit, Kommunikation und Interaktion von Menschen verbessert und erweitert (Preece, Rogers & Sharp, 2002).

Das *User-Interface Design* befasst sich im Besonderen mit der Gestaltung der Benutzungsoberfläche, die bei der Verwendung wahrgenommen wird. Neben Gestaltungsrichtlinien, wie beispielsweise den „Acht Goldenen Regeln“ (Shneiderman, 1998), werden ebenfalls ästhetische Fragen behandelt. Aktuelle Forschungsfragen aus diesem Bereich beschäftigen sich mit der Verwendung von „Pattern“. Pattern, im deutschen auch „Entwurfsmuster“ genannt, sind etablierte, gut dokumentierte und kontextualisierte Designentscheidungen für häufig auftretende Probleme (Janeiro, Barbosa, Springer & Schill, 2009).

Gegenstand des Wissenschaftsbereichs „*Human Factors*“ ist es, das Verhältnis zwischen Menschen und ihren Aktivitäten im Umgang mit soziotechnischen Systemen zu optimieren. Menschliche Faktoren sind *„[…] alle physischen, psychischen und sozialen Charakteristika des Menschen, insofern sie das Handeln in und mit soziotechnischen Systemen beeinflussen oder von diesen beeinflusst werden“* (Badke-Schaub, 2008, S. 4). Viele Erkenntnisse dieser Disziplin sind in Gestaltungsprinzipien und Richtlinien eingeflossen. Durch die Beachtung dieser Prinzipien bei der Entwicklung von Systemen soll erreicht werden, die Effizienz und Sicherheit des Systems zu verbessern und das Wohlergehen des Menschen zu verbessern. Wichtig könnten diese Faktoren beispielsweise bei sicherheitskritischen Systemen wie der Luft- und Raumfahrt sein (Badke-Schaub, 2008). Bei der Optimierung solcher Systeme sei somit immer auf die Interaktion zwischen dem sozialen und technischen Teilsystem eines Arbeitssystems zu achten.

2.2 UE-Prozesse

Ziel des UE ist die *„[…] Sicherstellung der Umsetzung gebrauchstauglicher Lösungen bei der Entwicklung von Software […]“* (Nebe et al., 2009, S. 342), wobei dies am besten durch einen definierten Entwicklungsprozess gewährleistet wird, der dieses Ergebnis reproduzierbar sicherstellt. Aus diesem Grund sollen für ein eingehendes Verständnis von UE die unterschiedlichen Erstellungsprozesse, die in den letzten Jahren entwickelt wurden, kurz diskutiert werden. Details zu der in den Prozessen verwendeten Methodik werden im darauf folgenden Kapitel erörtert. Bei den vorgestellten Erstellungsprozessen handelt es sich um eine breite Auswahl unterschiedlicher Verfahren. Manche sind als theoretische Ansätze zu verstehen und lassen sich nur bedingt praktisch anwenden, führten jedoch im Rahmen verschiedener Praxistests mit dem Software- und Beratungshaus zu wichtigen Erkenntnissen. Andere Verfahren stammen aus der Praxis und konnten ihre Verwendbarkeit bereits unter Beweis stellen. Alle verfolgen den Anspruch, die Entwicklung von „gebrauchstauglicher“ Software zu unterstützen.

In den vergangenen beiden Jahrzehnten wurden eine Reihe internationaler Standards aus dem Bereich der Human-Computer-Interaction (HCI) entwickelt. Viele dieser Standards vermitteln eher generelle Prinzipien, als auf Details bei der Verwendung einzugehen (Bevan, 2001). Dies soll eine möglichst flexible Integration in bereits bestehende Verfahren ermöglichen, erschwert jedoch auch die praktische Umsetzung der jeweiligen Verfahren. Einer der aktuell populärsten Verfahren ist dabei das „Human-centered Design“.

2.2.1 Human-centered Design

Der Human-centered Design-Prozess wird in der DIN EN ISO 9241-210:2011 zusammengefasst und detailliert beschrieben. Die Norm trägt in der deutschen Fassung den Titel „Prozess zur Gestaltung gebrauchstauglicher interaktiver Systeme“ und stellt den Menschen in das Zentrum des Entwicklungsprozesses. In vielen Fällen werden Softwarelösungen projektorientiert entwickelt. Das Projekt und nicht die Benutzer stehen im Mittelpunkt der Entwicklung. In der Ursprungsnorm ISO 13407, aus der die Überarbeitung ISO 9241-210 entstand, wurde noch der Begriff „User-centered Design“ verwendet. Nach Breuer und Maier (2009) geht der Begriff des *User-centered Design* auf Donald Norman (1998) zurück. In der Überarbeitung wurde die Bezeichnung durch „Human-centered

Design“ (HCD) ersetzt, um auszudrücken, dass durch die Verwendung dieses Prozesses nicht nur die späteren Benutzer einbezogen sind, sondern auch andere Stakeholder.

Der Begriff *„menschzentriert“* wird in der Überarbeitung weitestgehend synonym zu dem Begriff „benutzerzentriert“ verstanden. Auf zwei wesentliche Unterschiede in der Überarbeitung der Norm ISO 9241-210:2011 und der abgelösten Ursprungsnorm DIN EN ISO 13407:2000-11 soll an dieser Stelle hingewiesen werden. Die bereits erwähnte Begriffsänderung der „menschenzentrierten Gestaltung“ im Gegensatz zur „benutzerzentrierten Gestaltung“ soll verdeutlichen, dass auch Benutzer bzw. Stakeholder in den Entwicklungsprozess einbezogen werden sollten, die nicht unmittelbar als Anwender des Systems gelten, jedoch ein begründetes Interesse an dem System haben. Der zweite Unterschied in der Überarbeitung betrifft die Linearität des Prozesses (siehe Abbildung 3). In der Überarbeitung wurde der ursprünglich linear iterative Prozess durch bedingte Querverweise nach dem Prozessschritt „Lösung aus Benutzerperspektive evaluieren“ ergänzt. Durch diese Veränderung ermöglicht der Prozess eine höhere Flexibilität.

Die Norm richtet sich primär an Manager von Gestaltungsprozessen interaktiver Systeme. Neben einer Reihe verschiedener Begriffsdefinitionen beschreibt die Norm den in Abbildung 3 dargestellten „menschenzentrierten Gestaltungsprozess“.

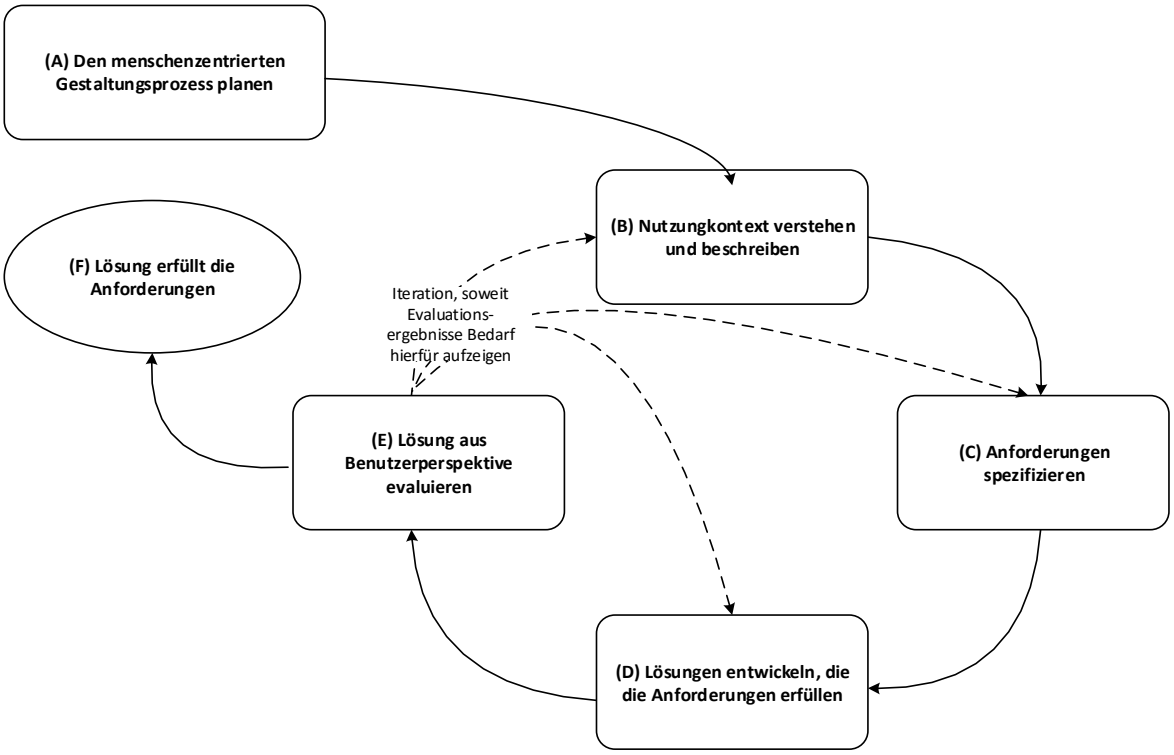


Abbildung 3: Prozess zur Gestaltung gebrauchstauglicher interaktiver Systeme aus der ISO 9241-210:2011

Für ein tiefergehendes Verständnis werden im Folgenden die einzelnen Prozessschritte näher erläutert. Dabei ist darauf zu achten, dass die einzelnen Schritte im Verlauf eines Projektes mehrfach durchlaufen werden.

(A) Den menschenzentrierten Gestaltungsprozess planen

Jedes Projekt beginnt demnach mit der Projektplanung, in der neben den üblichen Projektaktivitäten, folgende Aktivitäten des „menschenzentrierten Gestaltungsprozesses“ ergänzt werden sollten. In dieser Planungsphase ist es wichtig, den Aktivitäten feste Zeiten und Ressourcen zuzuweisen, um sie in der Projektplanung entsprechend berücksichtigen zu können. Insbesondere die Integration der Iterationen und der zusätzliche Informationsaustausch zwischen den beteiligten Stakeholdern sind wichtige Punkte, die es zu berücksichtigen gilt.

(B) Nutzungskontext verstehen und beschreiben

Der Iterationsprozess beginnt mit dem Verständnis und der Dokumentation des Nutzungskontextes des zu entwickelnden Systems. In der Norm wird bereits an dieser Stelle darauf hingewiesen, dass die in dieser Phase erfassten

Informationen nicht erschöpfend sein können. Die Beschreibung des Nutzungskontextes sollte folgende Punkte beinhalten:

- *Benutzer* und andere *Interessensgruppen* müssen identifiziert werden, um anschließend deren *Ziele* und *Einschränkungen* beschreiben zu können. Details dieser Beschreibungen fehlen in der Norm, werden jedoch in Kapitel 2.3.1.1 näher diskutiert.
- Merkmale der Benutzer oder Benutzergruppen wie *Kenntnisse*, *Fertigkeiten*, *Erfahrung*, *Ausbildung*, *physische Merkmale*, *Gewohnheiten*, *Vorlieben* und *Fähigkeiten* sind zu erfassen.
- *Ziele* und *Arbeitsaufgaben* der Benutzer sind zu beschreiben. Bei der Beschreibung der Arbeitsaufgabe sind Merkmale wie z.B. den *typischen Ablauf*, *Häufigkeiten*, *Dauer* und *Abhängigkeiten* der auszuführenden Aufgabe zu identifizieren.
- In der Kontextanalyse wird die Umgebung beschrieben, in der das System eingesetzt werden soll. Hierzu gehören unter anderem Hardware, Software und weitere Materialien, die zur Erfüllung der Arbeitsaufgabe benötigt werden.

Der Nutzungskontext sollte in Form eines *Arbeitsdokumentes*[5] erfasst werden. Dieses darf zu Anfang noch sehr grob sein, sollte jedoch kontinuierlich erweitert, überarbeitet, fortgeschrieben und aktualisiert werden. Zudem ist gefordert, einen klaren Bezug des Nutzungskontextes zu den im Folgenden Schritt erhobenen Nutzungsanforderungen herzustellen.

(C) Festlegen der Nutzungsanforderungen

In vielen Vorgehensmodellen ist die Erhebung der Anforderungen, häufig als „Anforderungsaufnahme“ oder „Anforderungsanalyse“ bezeichnet, die Hauptaktivität der Analysephase. In der DIN EN ISO 9241-210:2011 wird darauf hingewiesen, diese Aktivität „ganzheitlicher“ zu verstehen. In dieser Phase sollten nicht nur funktionale und nichtfunktionale Anforderungen an das System aufgenommen werden. Vielmehr sollte darauf geachtet werden Anforderungen im *„[…] Zusammenhang mit der Beschreibung des vorgesehenen Nutzungskontext und der wirtschaftlichen Ziele des Systems zu liefern“* (ISO 9241-210:2011). So können Anforderungen auch organisatorische Änderungen oder abgeänderte Arbeitsweisen berücksichtigen. Auch die Verwendung und Kombination fremder Systeme wäre hier wünschenswert. Ausgehend vom Nutzungskontext sollten die Bedürfnisse der Benutzer und weiterer Stakeholder identifiziert werden. Es sollte dabei dokumentiert werden, „was“ die Benutzer erreichen wollen und nicht „wie“.

(D) Lösungen entwickeln, die die Anforderungen erfüllen

In dieser Phase sollen „Gestaltungslösungen“ auf Basis der erhobenen Anforderungen, des beschriebenen Nutzungskontextes, technischer Umsetzungsmöglichkeiten, Richtlinien, Normen und weiterer gewonnener Erkenntnisse entwickelt werden. Die Norm weist darauf hin, dass diese in *interdisziplinären Teams* geschehen solle. Neben der Gestaltung der Benutzungsschnittstelle und der damit zusammenhängenden Benutzer-System-Interaktion müssen die sich durch das System neu ergebenden „Benutzeraufgaben“ gestaltet werden. Für die Umsetzung der Gestaltungslösungen wird der Einsatz von *Szenarien*, *Prototypen*, *Simulationen* und anderer *originalgetreuen Nachbildungen* empfohlen (siehe hierzu Kapitel 2.3).

(E) Lösung aus Benutzerperspektive evaluieren

In dieser Phase gilt es, die Entwurfslösungen im Hinblick auf ihre Gebrauchstauglichkeit und den Erfüllungsgrad der Nutzungsanforderungen hin zu evaluieren. In der Norm wird mehrfach darauf hingewiesen, dass diese benutzerzentrierte Evaluation in sämtlichen Phasen der Entwicklung sinnvoll sei. Je länger die benutzerzentrierte Evaluation in einem Entwicklungsprozess aufgeschoben werde, desto größer können die Kosten infolge von Änderungen werden. Es werden zwei verschiedene Verfahren für diesen Prozess der benutzerzentrierten Evaluation vorgeschlagen.

[5] Der Begriff „Arbeitsdokument“ soll in der Norm unterstreichen, dass das erstellte Dokument, mit den Informationen des Nutzungskontextes, während des gesamten Prozesses weiterentwickelt und verfeinert wird. Anfangs kann das Dokument nur sehr grob verfasst sein. Zu jeder Iteration des Prozesses wird das Dokument weiter detailliert.

1. *Prüfung durch den Benutzer*: Abhängig von der jeweiligen Iteration und damit dem Projektfortschritt kann hier mit verschiedenen Mitteln der Benutzer in die Konzeption involviert werden. In frühen Phasen werden beispielsweise Modelle, Szenarien oder einfache Skizzen von Gestaltungslösungen empfohlen. Wichtig sei es dabei, immer ein Modell der Interaktion zur Lösung der jeweiligen Arbeitsaufgabe zu haben. Dieses sollte einen angemessenen und nachvollziehbaren Bezug zum jeweiligen Anwendungskontext besitzen. In späteren Iterationen sollten bisherige Prototypen zur Erledigung von Arbeitsaufgaben herangezogen werden, um frühzeitige Verbesserungsvorschläge berücksichtigen zu können. Den Benutzer Arbeitsaufgaben mit realem Kontextbezug erledigen zu lassen, sei aussagekräftiger, als ihm die Gestaltungskonzepte nur zu demonstrieren. Bei Erreichung eines Beta-Stadiums[6] des zu entwickelnden Systems empfiehlt die Norm die Durchführung von Feldvalidierungen. Hierbei sollte der Benutzer das System in seiner realen Arbeitsumgebung testen können. Informationen zu diesen Versuchen könnten über Fehlermeldungen, Hotlines, Berichte und Umfragen dem Entwicklungsprozess rückgekoppelt werden.

2. *Inspektionsbasierte Evaluierung* (ohne Benutzer): Diese Form der Evaluation sei kostengünstiger und könnte auch ein größeres Spektrum von Benutzern und ihren Arbeitsaufgaben abdecken. Idealerweise sollte diese Form der Inspektion von Fachleuten aus dem Bereich der Gebrauchstauglichkeit durchgeführt werden. Diese könnten sich in die Rolle des Benutzers versetzen und Gestaltungslösungen bzgl. der formulierten Nutzungsanforderungen überprüfen. Darüber hinaus würden sich auch Checklisten, bewährte Verfahren der Industrie, Richtlinien, Normen und andere allgemeine Richtlinien (siehe hierzu Kapitel 2.3.4) zur Gebrauchstauglichkeit für diese Form der Evaluierung eignen. Die Norm weist ausdrücklich darauf hin, dass die inspektionsbasierte Evaluierung primär dafür geeignet sei, offensichtliche Probleme aufzudecken. Sie empfiehlt diese Form der Evaluierung vor einer Prüfung durch reale Benutzer durchzuführen, um offensichtliche Probleme beseitigen zu können und damit die Kosten zu reduzieren.

(F) Lösung erfüllt die Anforderungen

Wenn die Lösung nun den Anforderungen entspricht, kann sie entwickelt werden. An dieser Stelle beginnt somit die eigentliche Implementierung. Wie jedoch die Informationen an die Softwareentwicklung weitergetragen werden könnten wird in der Norm nicht beantwortet.

[6] „Noch nicht kommerzielle, aber zu Test- und Feedback-Zwecken schon meist publizierte Version eines Programms.“ Fischer und Hofer (2011)

2.2.2 Usability Engineering Lifecycle

Nielsen stellte im Jahre 1993 in seinem Buch mit dem Titel *„Usability Engineering"* die für ihn wichtig erscheinenden Aspekte zum Thema UE vor (Nielsen, 1993). Nach seiner Definition sei es wichtig, „Usability" nicht als eindimensionale Eigenschaft einer Benutzungsschnittstelle zu sehen. Vielmehr bestünde Usability aus einer Vielzahl von Komponenten und müsse sich an folgenden fünf Attributen messen lassen (frei übersetzt nach Nielsen, 1993, S. 26):

- Erlernbarkeit: Das System sollte leicht zu erlernen sein, so dass der Benutzer schnell erste Arbeiten mit dem System erledigen kann.
- Effizienz: Das System sollte effizient zu nutzen sein, so dass sobald der Benutzer das System kennt, ein hohes Niveau an Produktivität möglich ist.
- Merkfähigkeit: Das System sollte einfach zu merken sein, so dass ein Gelegenheitsnutzer auch nach einer ganzen Weile das System sofort wiederverwenden kann, ohne alles neu zu erlernen.
- Fehler: Das System sollte eine geringe Fehlerrate haben, so dass der Benutzer wenig Fehler während der Bedienung macht. Sollte er Fehler erzeugen, so müssen diese schnell zu beheben sein. Darüber hinaus sollten auch keine katastrophalen Fehler erscheinen.
- Zufriedenheit: Das System sollte angenehm zu nutzen sein, so dass der Benutzer subjektiv zufrieden ist, wenn er es nutzt.

Erst durch die Festlegung solcher „Usability Attribute" kann Usability als präzisier- und messbare Ingenieursdisziplin angesehen werden. Nur durch sie kann in einem „Usability-Engineering-Prozess" ein Produkt systematisch evaluiert und verbessert werden. In ähnlicher Weise wird diese Ansicht auch in der Normreihe „ISO 9241 – Ergonomie der Mensch-System-Interaktion" vertreten. In dieser wird der Fortschritt anhand sogenannter Dialogkriterien überprüft (siehe hierzu Kapitel 2.3.4). Auch Shneiderman (1998) propagiert in Form der „Acht goldenen Regeln des Interface Design" die Einhaltung bzw. Überprüfung von fest definierten Eigenschaften. Den Usability Entwicklungsprozess bezeichnet Nielsen als „Usability Engineering Lifecycle", der durch eine Reihe verschiedener Aktivitäten beschrieben wird. Diese sollten im besten Fall über den ganzen Lebenszyklus eines Produktes durchgeführt werden. In der Regel dienen sie jedoch der Phase vor der Entwicklung einer neuen Benutzungsschnittstelle. Dies lässt sich gut auf die Entwicklung aktueller Softwareprodukte übertragen, da viele in verschiedenen Iterationen und neuen Versionen auf den Markt gebracht werden. Die Überprüfung eines Produktes sollte immer, relativ zu bestimmten Benutzern und der zu erledigenden Aufgabe, gemessen werden. So könnte beispielsweise ein Schreibprogramm gut für die Erstellung eines Briefes geeignet sein, nicht jedoch für die Erstellung von Serienbriefen.

Der UE Lifecycle von Jacob Nielsen unterteilt sich in eine Reihe verschiedener Prozessschritte, die überwiegend darauf abzielen, noch vor einem ersten Entwurf wichtige Fragestellungen zu klären. Damit könne verhindert werden, dass unnötige Funktionen implementiert würden, die am Ende nicht verwendet werden und somit unnötige Kosten verursachen. Das Vorgehensmodell des UE Lifecycle unterteilt sich in folgende Schritte (Nielsen, 1993, S. 72):

1. Verstehen des Benutzers
   1.1. Individuelle Eigenschaften
   1.2. Derzeitige und zukünftige Arbeitsweise
   1.3. Funktionale Analyse
   1.4. Entwicklung des Benutzers und die seiner Arbeit
2. Marktanalyse
3. Festlegen von Usability Zielen
   3.1. Analyse finanzieller Auswirkungen
4. Paralleles Design
5. Partizipatorisches Design
6. Koordiniertes Design der gesamten Benutzungsschnittstelle
7. Anwendung von Richtlinien und heuristischen Analysen
8. Entwerfen eines Prototypen
9. Empirische Testungen

10. Iteratives Design
    10.1. Festhalten an Designentscheidungen
11. Feedbackeinholung von Endanwendern

Im ersten Schritt ist es wichtig, dass die Produktentwickler die Seite des Anwenders kennenlernen und verstehen. Wichtige Faktoren sind dabei die individuellen Benutzereigenschaften und die Variabilität in der Aufgabenerfüllung. *Individuelle Benutzereigenschaften* sind beispielsweise die Arbeitserfahrung, der Ausbildungsstand, das Alter und die Vorerfahrung mit Computersystemen. Jedoch auch die Arbeitsumgebung und der soziale Kontext sind wichtige Faktoren, die in den Prozess der Produktentwicklung einfließen sollten. Je mehr Anwender ein Produkt verwenden sollen, desto schwieriger wird es, repräsentative Anwender auszuwählen. Ein Lösungsansatz zu diesem Problem wird in Kapitel 2.3.1.1 unter dem Begriff der „Verhaltensvariablen“ diskutiert. Für ein umfassendes Verständnis des Benutzers führt Nielsen die bereits seit den 1960er-Jahren etablierte Aufgabenanalyse auf (Annett & Duncan, 1967). Das ausführliche Verfahren und die Ursprünge der Aufgabenanalyse werden ausführlich in Kapitel 2.3.1.2 vorgestellt. Sie sei häufig die Grundlage für Metaphern zur Gestaltung der Benutzungsschnittstelle. Es sollte herausgefunden werden, welches die übergeordneten Ziele seien und wie diese derzeit erreicht werden könnten. Nielsen schlägt für die Analyse von Aufgaben eine Dekomposition von abstrakten Aufgaben in ihre jeweiligen Teilaufgaben vor (siehe Kapitel 2.3.1.2). Um eine Optimierung bei der zukünftigen Arbeitsweise mit dem System zu erreichen und nicht nur eine genaue Nachbildung der derzeitigen Arbeitsweise zu entwickeln, schlägt Nielsen eine *Funktionale Analyse* vor. In dieser geht es darum, bessere Alternativen für bestehende Arbeitsweisen zu finden. Diese Form der Analyse sollte sich jedoch immer eng an der derzeitigen Arbeitsweise orientieren, um dem Benutzer nicht zu viel Veränderung auf einmal zumuten zu müssen. Da Benutzer mit der Zeit immer vertrauter mit dem System werden, ist es wichtig ihnen auch „Expertenfunktionen“ anbieten zu können. Beispielsweise die Verwendung von „Shortcuts“, um gewisse Funktionen schneller ausführen zu können.

Ein erheblicher Aufwand beim Usability-Engineering kann durch eine *Marktanalyse* eingespart werden. Konkurrenzprodukte sind die besten Prototypen, die bei der Entwicklung verwendet werden können. Sie sind bereits vollständig implementiert und können somit leicht für empirische Erhebungen verwendet werden (Nielsen, 1993, S. 80). Um den Fortschritt der Entwicklung besser abschätzen zu können, schlägt Nielsen eine Priorisierung der Usability Ziele vor. Da es meist unmöglich ist, alle im vollen Umfang erfüllen zu können, sollen *Usability Ziele* festgelegt werden, die als Mindestanforderung für das Release des Produktes genügen. So könnte beispielsweise eine maximale Fehlerrate der Anwendung als Usability Ziel vereinbart werden. Zeitgleich soll errechnet werden, welchen finanziellen Vorteil die jeweiligen Usability Verbesserungen bewirken können. In der frühen Phase des Design-Prozesses ist es wichtig, verschiedene Lösungsansätze zu entwickeln (Nielsen, 1993). Designerteams sollten in dieser Zeit unabhängig voneinander und ohne sich auszutauschen, möglichst viele Lösungsalternativen erstellen. Diese sollten dann bewertet werden und die besten Ansätze iterativ immer weiter verfeinert werden. Nielsen bezeichnet dies als *paralleles Design*. Alternativansätze können dabei ebenfalls miteinander kombiniert werden. Die Firma Apple beispielsweise wendet dieses Verfahren unter dem Namen „10 to 3 to 1“ in ihrem Designprozess an (Walters, 2008).

Für eine funktionierende *Benutzerpartizipation* ist es wichtig, den Benutzern möglichst anschauliche Systemrepräsentationen zur Verfügung zu stellen. Eine Studie zeigte, dass Benutzer, die den Nutzen einer Systemfunktionalität anhand einer textuellen Beschreibung bewerten sollten, zu einem unterschiedlichen Ergebnis kamen als diejenigen, die Funktionen an dem System selbst ausprobieren konnten (Root & Draper, 1983). Dabei ist es wichtig mit den Anwendern und nicht deren Vorgesetzten zu diskutieren, denn häufig kennen Vorgesetzte nur bedingt die genaue Arbeitspraxis ihrer Angestellten und wenden das System nur selten an. Dabei ist zu berücksichtigen, dass die Partizipation nicht darauf ausgerichtet sein sollte, den Benutzer zu fragen, was er sich wünscht, da er häufig nicht dazu in der Lage ist, technische Möglichkeiten abwägen zu können. Damit ein System während seines gesamten Lebenszyklus nicht inkonsistent wird, ist es wichtig für ein Produkt generell gültige Standards festzulegen. Konsistenz ist eine der wichtigsten *Usability Eigenschaften*. Weitere Software-ergonomischen Kriterien, die zur Überprüfung des Maßes der Gebrauchstauglichkeit von Software eingesetzt werden können, sind in Kapitel 2.3.4 aufgeführt. Je nach Größe des Entwicklungsvorhabens sollte die Einhaltung dieser Standards von einer Person oder von einem Komitee überprüft werden. Durch sogenanntes Code Sharing, bei dem gewisse Codekomponenten in einem Programm immer wieder verwendet werden, kann Konsistenz auf ressourcenschonende Weise gewährleistet werden. Ebenfalls zur Wahrung der Konsistenz beitragend, sind *Richtlinien*. Sie listen etablierte und

meist generell geltende Prinzipien auf, die bei der Entwicklung eingehalten werden sollten. Näheres zu diesem Thema wird in Kapitel 2.3.4 behandelt. Die Überprüfung dieser Prinzipien durch Experten wird als „Heuristische Evaluation“ bezeichnet. Diese Art der Evaluation kann in allen Phasen der verschiedenen Reifegrade des Interface Designs geschehen. Für diese frühe Form der Überprüfung bestimmter Merkmale und Funktionalitäten der Software empfiehlt Nielsen den Einsatz von *Prototypen*. Die genaue Definition und die unterschiedlichen Arten von Prototypen werden in Kapitel 2.3.3 behandelt.

Empirische Evaluationen dienen der Aufdeckung von Problemen. Sie lassen sich in zwei verschiedene Dimensionen unterteilen. Je nachdem ob für die Evaluation Benutzer zur Verfügung stehen oder nicht. Das Ergebnis dieser Untersuchung ist eine Liste von Usability Problemen. Da eine solche Liste schnell sehr umfangreich werden kann, sollten die Probleme priorisiert werden. Nielsen schlägt hierfür fünf verschiedene Schweregrade vor (Nielsen, 1993):

(0) Kein Usability Problem
(1) Kosmetisches Problem, welches nur dann gelöst wird, wenn genügend Zeit sein sollte
(2) Geringes Problem
(3) Großes Problem, das möglichst schnell gelöst werden sollte
(4) Usability Katastrophe, führt dazu, dass ein Produkt vor der Lösung nicht veröffentlicht werden kann

Zusätzlich kann die Priorität auch von der Anzahl der betroffenen Benutzer abhängig gemacht werden. Das Abarbeiten der Usability Probleme führt somit quasi automatisch zu einem „Iterativen Design“ und somit zu neuen Versionen des Produktes. Dabei ist wichtig die entsprechenden Gründe für die neuen Designentscheidungen festzuhalten. Die Hauptarbeit im UE nach dem Release eines Produktes besteht darin, Daten für die nächste Generation zu sammeln. Sie sind in diesem Fall die besten Prototypen, die für einen Test zur Verfügung stehen können. Die Erhebung der Daten kann auf klassische Weise mit Interviews, Fragebögen oder Beobachtungen geschehen.

Die besten Methoden sind jedoch überflüssig, wenn sie nicht angewendet werden. Häufig sind Projekte zeitlich sehr eng kalkuliert, und es bleibt kaum Zeit für einen umfassenden Usability Prozess wie den UE Lifecycle. Zu diesem Zweck beschreibt Nielsen einen Minimal Prozess, mit dessen Hilfe immer noch gute Ergebnisse erreicht werden können. Er bezeichnet diese Methodik als *Discount UE*. Sie besteht aus folgenden vier Methoden:

- Beobachtung von Benutzern und ihren Aufgaben
- Erstellung von Szenarien
- Durchführung von vereinfachtem „Thinking Aloud“
- Anfertigung von heuristischen Evaluationen

Verschiedene Publikationen (Abrazhevich, 2009; Behring & Petter, Andreas, Mühläuser, Max, 2009; Kane, 2003) beschäftigen sich mit möglichen Integrationsansätzen dieses reduzierten Verfahrens in die Agile Softwareentwicklung.

2.2.3 Contextual Design

Das Verfahren „Contextual Design“ (CD) von Karen Holtzblatt und Hugh Beyer bietet eine umfassende und praktisch anwendbare Auswahl an Methoden (Beyer & Holtzblatt, 1998). Es vereint die verschiedenen Ansätze und Techniken aus dem HCI-Bereich. CD unterstützt dabei den kompletten Designprozess von der initialen Datenaufnahme beim Kunden bis hin zum fertigen Lösungsansatz. Der Prozess vereint die Techniken, die dazu benötigt werden, um ein System entwickeln zu können, welches die Bedürfnisse der Kunden erfüllt. In der Regel bedeutet dies immer eine Umstrukturierung des organisatorischen Arbeitsprozesses beim Kunden. Die verwendeten Methoden sollen sowohl bei der Erfassung als auch bei der strukturierten Dokumentation der relevanten Daten helfen. Nachfolgende Auflistung veranschaulicht die zum Einsatz kommenden Methoden und Modelle:

- Kontextanalyse durch Contextual Inquiry, um Daten zu sammeln
- Interpretationssitzung und Arbeitsmodellierung des Ist-Zustands
    - Flußmodelle (Flow models)
    - Ablaufmodelle (Sequence models)
    - Artefaktmodelle (Artifact models)
    - Einflussmodelle (Cultural models)
    - Umgebungsmodelle (Physical models)
- Konsolidierung
    - Konsolidierte Arbeitsmodelle (Arbeitsstruktur der Gruppe)
    - Affinity Diagramm (Probleme, Wünsche und Hoffnungen)
- Neugestaltung der Arbeit
    - Nutzung der konsolidierten Arbeitsmodelle
    - Entwicklung der Vision
    - Konstruieren eines Storyboards
- Prototyping

Kontextanalyse durch Contextual Inquiry

Die Basis aller zu erstellenden Arbeitsmodelle ist die Erfassung der relevanten Bedürfnisse der Kunden und Anwender. Zu diesem Zweck schlagen Beyer und Holtzblatt eine spezielle Technik bei der Ermittlung dieser Informationen vor (Beyer & Holtzblatt, 1998). Die *Contextual Inquiry* ist eine Mischung aus Befragung und Beobachtung. Sie sollte immer am Arbeitsplatz des Anwenders stattfinden. Nur hier kann die detaillierte Komplexität der Arbeit erfasst werden. Da der Benutzer als Experte seiner Domäne anzusehen ist, sollte der Analyst[7] sich in die Rolle eines „Lehrlings“ versetzen. Der Benutzer nimmt die Rolle des „Meisters“ ein und erklärt seinem Lehrling die Arbeit. Der Analyst sollte seine Interpretationen der Beobachtungen mit dem Kunden zurück kommunizieren. Dies verhindert mögliche Fehlinterpretationen. Beyer und Holtzblatt schlagen als Faustregel für ein Projekt vor, zwei bis drei Benutzer pro Rolle auf diese Weise zu untersuchen. Abhängig von der Komplexität der untersuchten Arbeit sollten für ein solches Gespräch etwa zwei bis drei Stunden eingeplant werden. Produkte, die gleich von mehreren Organisationen eingesetzt werden (Standardsoftware), verlangen entsprechend mehr Untersuchungen. Hierbei sollte darauf geachtet werden, einen möglichst breiten Querschnitt an Daten zu erfassen. Für die Dokumentation der Daten werden eine Reihe verschiedener Modelle empfohlen, die im Folgenden erläutert werden.

Interpretationssitzung und Arbeitsmodellierung des Ist-Zustands

In der sogenannten *Interpretationssitzung* trifft sich ein Team aus Analysten, Modellierern, Protokollierern und weiteren interessierten Personen. Sie dient dazu, Informationen aus den Interviews mit dem Rest des Teams zu teilen. Nach Beyer und Holtzblatt sollte die Sitzung zeitnah nach den Interviews bei den Kunden geschehen. Jeder Analytiker wird in der Sitzung durch die restlichen Teilnehmer zu einem einzigen Benutzer befragt. Die Sitzung sollte nicht in Form einer Präsentation verlaufen. Vielmehr sollte jedes Interview zu einem Benutzer gemeinsam durch das ganze Team verstanden und diskutiert werden. Alle Beteiligten sollen dabei laut denken und ihre Einwände oder Verständnisprobleme klar kommunizieren. Auf diese Weise sollen im Verlauf der gesamten Sitzung

[7] Person aus dem Designerteam, die das Contextual Inquiry durchführt

eine Reihe verschiedener Arbeitsmodelle entstehen, die im Folgenden vorgestellt werden. Jedes Modell sollte dabei durch eine eindeutige „Identifikationsnummer“ dem untersuchten Benutzer zugeordnet werden können. Zusätzliche Informationen, die in dieser Sitzung eingebracht werden, wie Einwände, Wünsche, Anregungen, Anforderungen und Bedenken werden von einem Protokollierer in einer Liste festgehalten. Diese Notizen dienen im späteren Verlauf als Grundlage für das sogenannte Affinity Diagram. Auf diese Weise lernen alle Beteiligten die Perspektive des Anderen kennen. Durch einen Moderator sollte das gesamte Treffen gelenkt werden.

Arbeit visualisieren durch das Flussmodell (Flow Models)

Ein Flow Model stellt die Aufteilung einer Arbeit aus dem Betrachtungswinkel einer Person dar. Es verdeutlicht, welche Personen für die Erfüllung einer Aufgabe zusammenarbeiten. In einer Interpretationssitzung können somit eine ganze Reihe unterschiedlicher Flow Models zu den einzelnen Benutzern entstehen. Folgende Komponenten können in diesen dargestellt werden:

- *Individuen*: Alle an der Arbeit beteiligten Individuen. Jedes Individuum sollte eine eindeutige Kennung – wie „Kunde 1“ – bekommen. Zu Anfang wird empfohlen, sich einzelne Personen vorzustellen und nicht gleich mit der generalisierten Rolle zu arbeiten. Zudem sollte die Stellenbezeichnung anstatt eines persönlichen Namens verwendet werden.
- *Aufgaben* bzw. *Verantwortlichkeiten*: Jedem Individuum werden die relevanten arbeitsteiligen Aufgaben zugeordnet.
- *Gruppen*: Eine Gruppe besteht aus mehreren Individuen. Wenn eine Aufgabe von einer Gruppe erledigt wird, kann diese wie ein Individuum betrachtet werden.
- *Ablauf* (Flow): Verbindungen zwischen den einzelnen Individuen verdeutlichen die Kommunikation zwischen allen Beteiligten, die zur Erledigung der Aufgabe notwendig sind. Diese können informeller Natur sein oder formell in Form eines Arbeitsobjekts wie einer Rechnung.
- *Artefakte*: Als Artefakte werden alle Arten relevanter Arbeitsobjekte angesehen, die materiell oder auch immateriell sein können.
- *Kommunikationsthema* oder *Aktion*: Die Kommunikation zwischen den Beteiligten kann mit einem aussagekräftigem Thema oder der ausgeführten Aktion beschrieben werden.
- *Orte*: Wenn ein bestimmter Ort, wie ein Arbeitsplatz, für die Durchführung der Arbeit wichtig ist, sollte dieser ebenso mit aufgenommen werden.
- *Störungen***:** Die in der Beschreibung auftretenden Probleme in der Kommunikation oder der Koordination können durch einen (roten) Blitz symbolisiert werden.

Die entstehenden Daten könnten daran anschließend textuell erfasst werden. Beyer und Holtzblatt empfehlen jedoch die Dokumentation in Form von Diagrammen. Für jedes Flow Model können im ersten Schritt alle an einer Aufgabe direkt oder indirekt beteiligten Individuen aufgenommen werden. Im Zentrum des Models steht die interviewte Person.

In der nachfolgenden Abbildung 4 ist hierfür beispielhaft der Rezeptionist mit der eindeutigen Bezeichnung U1, Gäste und alle weiteren Personen, mit denen er zusammenarbeitet, dargestellt.

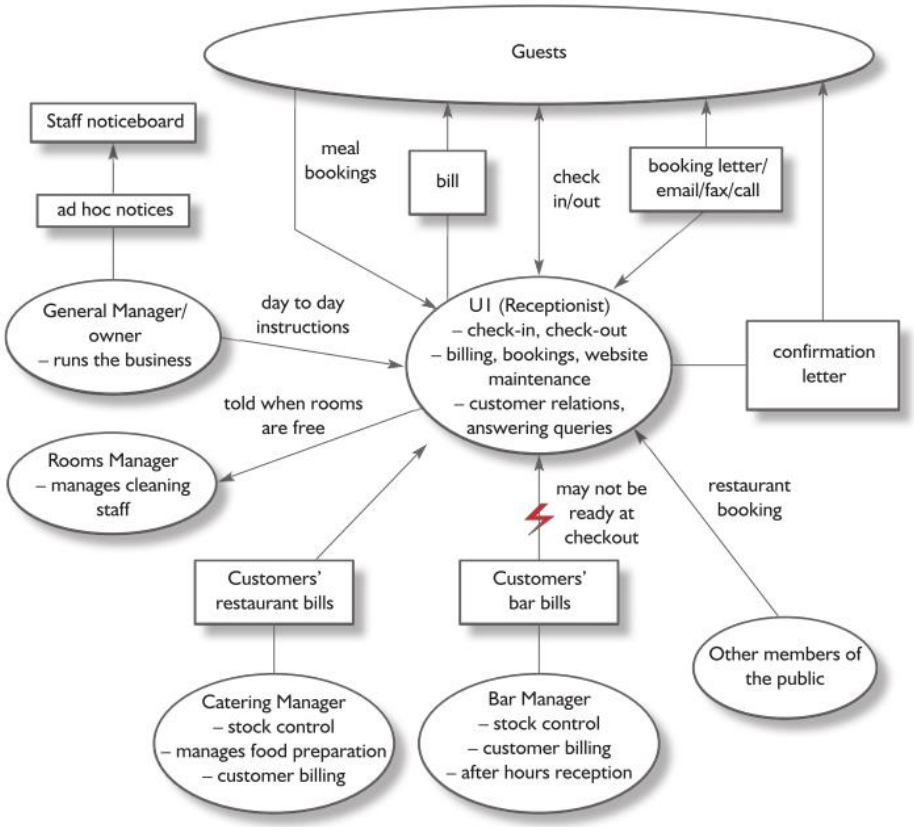


Abbildung 4: Beispiel eines Flow Models (Benyon, 2010)

Im nächsten Schritt sollte die Kommunikation in Form von gerichteten Kanten gekennzeichnet werden. Diese sollten mit dem entsprechenden Thema bezeichnet werden. Alternativ werden die ausgetauschten Artefakte dargestellt. Im oben angeführten Beispiel ist dies die Rechnung zwischen dem Rezeptionisten und dem Gast, dargestellt in einem Rechteck. Orte wie das schwarze Brett (Staff noticeboard) für die Mitarbeiter werden ebenfalls in Rechtecken aufgenommen. Als Letztes sollten Störungen durch einen roten Blitz gekennzeichnet werden. Die Störung sollte kurz jedoch detailliert beschrieben werden.

Sequence Model

Sequence Models zeigen die detaillierte Struktur einer Aufgabe auf. Zu diesem Zweck wird eine Aufgabe in ihre Folge von Aktionen aufgeteilt. Dabei entsteht eine Sequenz von Teilaktivitäten. Alternativ könne die Aufgabe auch als hierarchische Aufgabendekomposition (HTA[8]) dargestellt werden (Benyon, 2010). Wenn eine Aufgabe auf eine andere Art und Weise ausgeführt wird, sollte dies in einem eigenen Sequence Model dokumentiert werden. Eine Aufgabe erfüllt auch immer einen bestimmten Zweck. Dieser kann gut dem vorher erstellten Flow Model entnommen werden und unter dem Punkt „Intent" neben einer Aktion notiert werden. Jede Sequenz besitzt darüber hinaus ein auslösendes Ereignis, welches am Anfang des Models steht. Als Letztes sollten noch alle Störungen, wie bereits im Flow Model, durch einen Blitz identifiziert und kurz beschrieben werden.

Artifact Models

Artefakte sind Gegenstände, die von Menschen während der Arbeit erstellt, verwendet oder modifiziert werden (Beyer & Holtzblatt, 1998). Sie sollten aus zwei Gründen aufgenommen werden: Zum einen konkretisieren sie die derzeitige Arbeitsweise. Beispielsweise könnte der Rezeptionist in seinem Gästebuch mit einer selbst entwickelten Codierung zusätzliche Informationen zu einem Gast notieren oder sich die Arbeitsweise anderweitig er-

[8] Wird detailliert in Kapitel 2.3.1.2.1 behandelt

leichtern. Zum anderen beschreibt das Artefakt die wichtigsten Informationen, die später beim Design berücksichtigt werden sollten. In einigen Fällen kann ein Artefakt sogar als Ausgangslage eines Designs dienen. Beyer und Holtzblatt schlagen vor, folgende Informationen über ein Artefakt aufzunehmen (Beyer & Holtzblatt, 1998):

- *Information*: Inhalt des Objekts selbst;
- *Struktur*: Aufbau und Aufteilung des Objekts;
- *Bemerkungen*: Informell verwendete Ergänzungen wie beispielsweise Post-its;
- *Darstellung*: Verwendete Farben, Formen und Schriftarten;
- *Verwendung*: Arbeitsweise mit dem Objekt;
- *Störungen*: Probleme, die mit dem Artefakt auftreten.

Artefakte sollten während des Contextual Inquiry aufgenommen werden. Entweder wird das Artefakt selbst verwendet, oder man erstellt eine Kopie, ein Foto oder eine aussagekräftige Skizze des Objekts.

Cultural Model

Beyer und Holtzblatt sprechen von der Kultur der Arbeitsumgebung, die es im Folgenden zu erfassen gilt.

> *"Work takes place in a culture, which defines expectations, desires, policies, values, and the whole approach people take to their work […]"* (Beyer & Holtzblatt, 1998, S. 107)

Dieser Aspekt kann entscheidend für den Erfolg eines Produkts sein (Benyon, 2010). Häufig sind diese Erwartungen und Wünsche unausgesprochen oder sogar unsichtbar. Manchmal unterscheiden sich auch die jeweiligen Intentionen der Bereiche in einer Arbeitsumgebung. Wahrzunehmen sind die Aspekte durch Beobachtung wiederkehrender Verhaltensmuster oder Einstellungen. Das Modell setzt sich aus folgenden Komponenten zusammen:

- *Beeinflusser* sind Personen oder Einrichtungen, die die Art und Weise der Arbeit beeinflussen. Sie werden als Blasen in dem Model dargestellt.
- Durch die *Überlappung* von Blasen wird der beeinflussende Faktor symbolisiert. So wird in dem Beispiel in *Abbildung 5* aufgezeigt, dass der Rezeptionist durch den Besitzer des Hotels und die Kunden beeinflusst wird.
- Die Richtung des Einflusses wird durch *Pfeile* symbolisiert.
- *Störungen* können Standards, Richtlinien oder sich widersprechende Absichten sein.

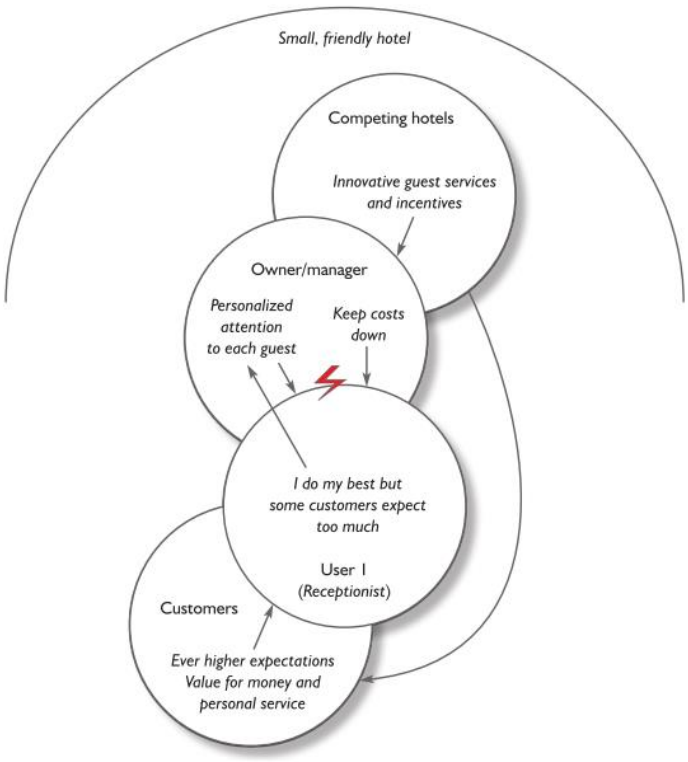


Abbildung 5: Beispiel für ein Cultural Model (Benyon, 2010)

Physical Model

Das Physical Model gibt die physikalische Arbeitsumgebung wieder. Dabei ist kein genauer Grundriss der Arbeitsstätte anzufertigen, sondern es sollten für die Arbeit relevante Stationen erfasst werden. Wichtig sind Wege, die zurückgelegt werden, um für die Arbeit wichtige Gegenstände wie den Drucker oder eintreffender Faxe zu erreichen. Häufig ist auch die Aufteilung des Arbeitsplatzes hilfreich, um beschriebene Arbeitsweisen besser verstehen zu können. Oft reicht eine einfache Skizze der Raumaufteilung oder des Arbeitsplatzes.

Konsolidierung

Bisher wurden immer nur einzelne Menschen bzw. Benutzer untersucht. Die meisten Softwareprodukte werden jedoch für mehrere Menschen entwickelt. In der Konsolidierungsphase sollen die bisher zu jedem Individuum entstanden Arbeitsmodelle zusammengeführt werden. Dabei entstehen neue Arbeitsmodelle in der zuvor vorgestellten Weise. Diesmal sollen sie jedoch ein Gesamtbild der Arbeitssituation veranschaulichen, indem jedoch die Bedürfnisse jedes Individuums berücksichtigt werden.

Die bisher erstellten Modelle können dazu verwendet werden, Einzelheiten zu verdeutlichen. Ziel ist es, alle relevanten Aspekte eines Benutzers dem Team mitzuteilen. Das Team hinterfragt, beleuchtet und deckt weitere wichtige Facetten auf, die bisher nicht bedacht wurden. Dieses Vorgehen wird für alle interviewten Benutzer durchlaufen. Die Aufgabe der Modellierer (Beyer und Holtzblatt empfehlen zwei Modellierer) ist es dabei, die vorgetragenen Informationen auf einem Flipchart festzuhalten. Es sollten neue Flow, Culture und Sequence Models entstehen, welche das Gesamtbild des Anwendungskontextes verdeutlichen. Erst durch die Zusammenführung aller Modelle und somit aller Benutzer kann ein solches Gesamtbild verfestigt werden.

Affinity Diagram

Mit Hilfe des sogenannten Affinity Diagrams sollen die zuvor vom Protokollanten notierten Informationen wieder zusammengeführt und geordnet werden. Alle in dem Projekt angefallenen Punkte sollen in kurzer Form, maximal einem Satz, auf eine Art Post-it© geschrieben werden, um sie anschließend thematisch gruppieren zu können. Das so entstehende Diagramm zeigt alle wesentlichen Elemente der Arbeitspraxis eines Kunden auf. In Summe können in diesem Konsolidierungsprozess hunderte oder auch tausende einzelner Punkte zusammen kommen.

Neugestaltung der Arbeit

An dieser Stelle beginnt der eigentliche Designprozess, nicht jedoch der des zu entwickelnden Systems, sondern die der Arbeit mit all ihren Facetten. Zu diesem Zweck sollten alle zur Verfügung stehenden Arbeitsmodelle und Notizen verwendet werden. So kann beispielsweise mit Hilfe des konsolidierten Flow Models die Rollenverteilung untersucht und optimiert werden. Dabei sollte darauf geachtet werden, dass Benutzer ihre Rollen während des Arbeitsalltags wechseln können und Rollen von verschiedenen Benutzern ausgeübt werden (Beyer & Holtzblatt, 1998). An dieser Stelle verweist Karen Holtzblatt auf die Verwendung von Persona, die in Kapitel 2.3.1.1 diskutiert werden (Holtzblatt, Wendell & Wood, 2005). Ziel ist es, überladene Rollen zu minimieren. Menschen, die zu viele sich gegenseitig behindernden Aufgaben ausüben, sollten beispielsweise durch Neuverteilung oder Teilautomatisierung entlastet werden. Die Entwicklung eines neuen Softwaresystems bedeutet auch immer einen direkten Eingriff in bestehende Arbeitsstrukturen.

Vision und Storyboards

Auf Grundlage des Wissens der bisherigen und der optimierten Arbeitsweisen sollte im nächsten Schritt eine Vision des neuen Systems entwickelt werden.

> *"Invention is a response to some life or work practice by a designer or technologist who, seeing the need and knowing the technology, imagines a new possibility."* (Holtzblatt, 2008, S. 957)

Der Vorschlag von Karen Holtzblatt ist die Durchführung einer „Visionssitzung“. In dieser sollten die Rollen eines Zeichners, der die Vorschläge der anderen Teilnehmer aufmalt, eines Moderators, der die Sitzung steuert und einer Reihe weiterer Teilnehmer vertreten sein, die neue Designlösungen beisteuern. Im Vorweg der Sitzung sollten alle bisherigen Informationen nochmals durchgegangen und verstanden worden sein. Während der Sitzung sollte darauf geachtet werden, dass Visionen nicht zum Zeitpunkt der Entstehung evaluiert werden, keine Idee Eigentum eines Einzelnen ist und alle zum Zeichner sprechen, da eine Idee, die von diesem nicht erfasst wurde, auch nicht mit in das Design fließt. Die Visionen sollten immer aus Sicht des Benutzers und seinen Aktivitäten formuliert werden. Nachdem einige davon zusammengetragen wurden, sollten die Vor- und Nachteile diskutiert werden, um anschließend eine konsolidierte Gesamtlösung daraus ableiten zu können. Um die Details der Gesamtlösung zu erarbeiten müssen einzelne Aufgaben eines Benutzers mit dem System durchgespielt werden. Hierzu werden im Contextual Design sogenannte Storyboards (siehe auch Kapitel 2.3.3.1) verwendet.

> *"Storyboards are like freeze-frame movies of the new work practice. Like storyboarding in film, the team draws step-by-step pictures of how people will work in their new world. Storyboards include manual steps, rough user interface components, system activity and automation, and even documentation use. Storyboards act like high-level use cases."* (Holtzblatt et al., 2005, S. 230)

Erst diese Storyboards verdeutlichen mögliche Schwierigkeiten in der Anwendung eines Systems.

Prototyping

Um die entworfenen Visionen mit echten Benutzern testen zu können, müssen Prototypen entwickelt werden (siehe auch Kapitel 2.3.3). Im Speziellen sollten bei dem Contextual Design Papierprototypen verwendet werden.

> *"A paper prototype is a paper representation of your product. Constructed out of Post-it® notes, various other pieces of paper, or any other materials you need to use, it allows you to test your design with the user interctively."* (Holtzblatt et al., 2005, S. 246)

Karen Holtzblatt schlägt vor, hierfür die bereits interviewten Benutzer zu involvieren. Die Prototypen werden dann so häufig verfeinert, bis die Lösung für alle Stakeholder akzeptabel ist.

2.2.4 Scenario-based Design

Rosson und Carroll beschreiben ein Verfahren, welches überwiegend auf der Verwendung von Szenarien basiert (Rosson & Carroll, 2002). Das sogenannte Scenario-based Design (SBD) unterteilt sich in die drei Teilaktivitäten Analyse, Design und Erstellung von Prototypen mit Evaluation (Abbildung 6). Die letzten beiden Teilaktivitäten sollen dabei so häufig wiederholt werden, bis sie bestimmten Bedingungen entsprechen. Der Prozess sieht eine sukzessive Transformation der Szenarien über den gesamten Entwicklungsprozess vor. Zunächst werden in der Analysephase durch Interviews, Informationen über Benutzer und andere Interessensvertreter sowie deren Tätigkeiten gesammelt. Diese dienen als Grundlage für die Erstellung von sogenannten Problemszenarien. Im Bereich des Requirements-Engineerings (siehe Kapitel 3.1.2) wird diese Form der Szenarien auch als „Indikative Szenarien“ bezeichnet. Sie beschreiben den aktuellen Arbeitsablauf und zeigen etwaige Probleme auf. Sie bilden die Basis für die anschließende dreigliedrige Designphase.

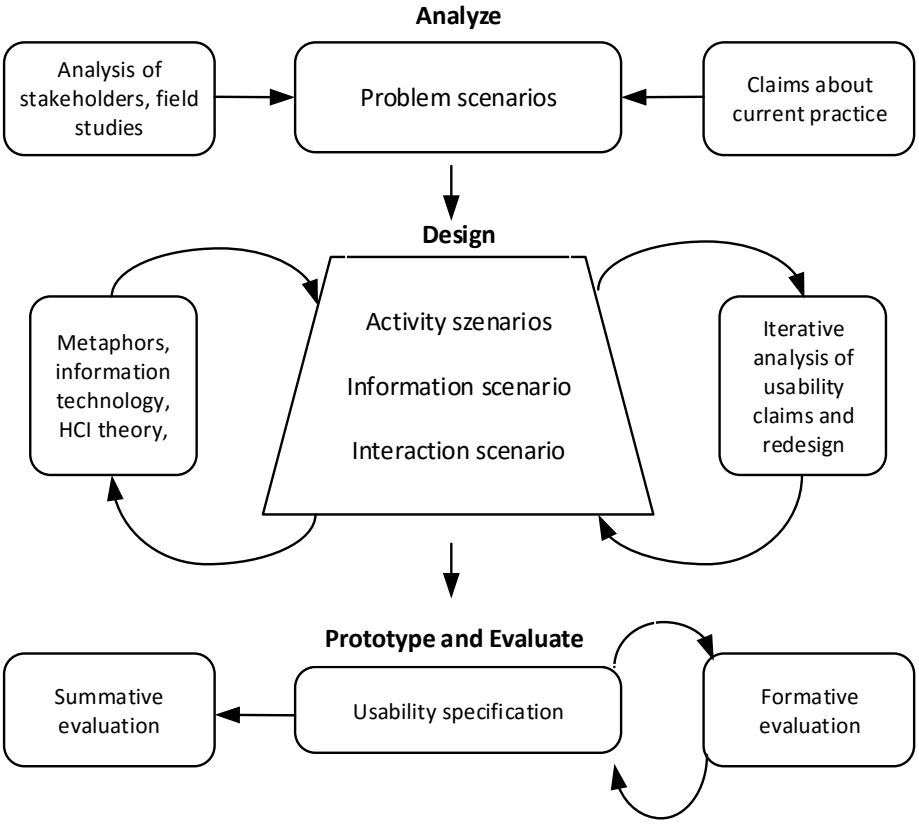


Abbildung 6: Vorgehensmodell Szenario-basierter Entwicklung (Rosson & Carroll, 2002, S. 25)

Im SBD werden Szenarien für die Analyse von Anforderungen, der Kommunikation neuer Designs, der Einleitung der Prototypenerstellung und Implementierung wie auch der Organisation der Evaluation eingesetzt (Rosson & Carroll, 2002). Ein Szenario wird im SBD wie folgt definiert:

> *"Scenarios are stories. They consist of a setting, or situation state, one or more actors with personal motivations, knowledge, and capabilities, and various tools and objects that the actors encounter and manipulate. The scenario describes a sequence of actions and events that lead to an outcome. These actions and events are related in a usage context that includes the goals, plans, and reactions of the people taking part in the episode."* (Rosson & Carroll, 2003)

Für die Erhebung der nötigen Informationen werden im SBD Interviews mit den Benutzern und Feldstudien vor Ort zur Untersuchung der aktuellen Arbeitssituation vorgenommen. Auf diese Weise können in den Problemszenarien wichtige Charaktereigenschaften der Benutzer, typische und problematische Aufgaben, verwendete Werkzeuge und Informationen über interne Betriebsstrukturen festgehalten werden.

Der sukzessive Transformationsprozess wird im SBD über *„Claims“* vorangetrieben. Sie werden sowohl in der Analyse- als auch der Designphase eingesetzt. In der Analysephase sind Claims eine Liste wichtiger Merkmale einer Situation und ihrer Auswirkung auf die Benutzer. In der Einleitungsphase zum Design beschreiben Claims mögliche Lösungsvorschläge. Erst durch die Ergänzung von positiven und negativen Aspekten jedes Merkmals, wie in Abbildung 7, entspricht ein Claim seiner endgültigen Form im SBD.

A. Science Fiction Club on a Web forum
Accessing an online meeting through an Web discussion forum
+ enables convenient browsing of an entire discussion at many points in time or place
- but when a discussion becomes long or complex, it may be difficult to browse or understand
B. Science Fiction Club in a Community MOO
Accessing an online meeting room by “walking” through a spatial model of the town
+ allows fortuitous social encounters while moving around the MOO
+ evokes application and development of real world spatial knowledge about the town
- but navigation to non-immediate sites may be tedious or awkward
- but the overall spatial model may be poorly evoked by step-by-step navigation

Abbildung 7: Beispiel zweier Claims (Rosson & Carroll, 2003)

In der Designphase durchlaufen Szenarien drei verschiedene Transformationsstufen. Zunächst werden „Aktivitätsszenarien“ erstellt. Aus diesen sollte klar ersichtlich werden, welche Aufgaben und Aktivitäten ein Benutzer zukünftig mit dem System erledigen soll. Dabei werden ausschließlich reine Funktionalitäten beschrieben, jedoch explizit nicht, wie diese aussehen und wie der Benutzer mit diesen interagieren soll. Auf diese Weise können im Sinne des „Parallel Designs“ (siehe Kapitel 2.2.2) unterschiedliche funktionale Sequenzabläufe modelliert und vor allem bewertet werden, ohne überflüssige Prototypen zu entwickeln, die am Ende verworfen werden müssen.

In der nächsten Stufe werden einige der Aktivitätsszenarien zu Informationsszenarien ausgearbeitet. Das heißt, es werden Informationen ergänzt, die dem Benutzer zur Verfügung gestellt werden müssen, um seine geplanten Aktivitäten ausüben zu können. In dieser Phase zu der es, wie in Kapitel 1.1 bereits erwähnt, den eigenen Fachbereich des „Information Designs“ gibt, soll geklärt werden, welche Objekte und welche mögliche Aktionen wie dargestellt und angeordnet werden sollen, um dem Benutzer die Wahrnehmung und das Verstehen zu erleichtern. Hierzu gehören die Erstellung von Anwendungsdialogen, Menüs, Dialogboxen und Icons.

Im letzten Schritt werden die Informationsszenarien zu Interaktionsszenarien ergänzt. Diese Form der Szenarien beschreibt alle Details der Aktionen, die der Benutzer ausführen kann, und die Reaktion des Systems. Information Design und Interaction Design versuchen dabei die Wahrnehmungs- bzw. die Ausführungskluft zu überbrücken, wie sie durch Norman und Draper beschrieben wurde (Norman & Draper, 1986).

Die letzte Phase des SBD *„Prototype and Evaluate“* ist als Überprüfung der erstellten Prototypen zu verstehen. Dabei können unterschiedliche analytische oder empirische Evaluationsmethoden (siehe Kapitel 2.3.4) angewandt werden. Ziel der Evaluation ist die Verbesserung des iterativen Vorgehens bei der Softwareentwicklung.

2.2.5 Activity-centered Design

In der evolutionären Entwicklung der verschiedenen UE-Prozesse, gab es in den letzten Jahren immer wieder Veränderungen des Betrachtungsstandpunktes. Zu Beginn der Entwicklung tendierten Designer zu einem „Computer-centered Design“, welches sich im besten Fall den Wünschen und Bedürfnissen der Kunden annäherte, im schlimmsten Fall jedoch diese ignorierte. Der technologische Fortschritt und die Machbarkeiten bestimmten das Design (Gay & Hembrooke, 2004). Diesem Missstand wurde durch das Human-centered Design begegnet, indem Ziele und Eigenschaften der Benutzer in den Mittelpunkt der Entwicklung gerückt wurden (siehe Kapitel 2.2.1). Eine aktuelle Bewegung, das Activity-centered Design (ACD), stellt im Gegensatz dazu die Aktivitäten eines Benutzers mit dem System in den Mittelpunkt der Betrachtung. Selbst Mitbegründer des Human-centered Designs wie Donald Norman geben zu bedenken, dass eine Fokussierung auf die Vorlieben, Abneigungen, Fähigkeiten und Bedürfnisse einzelner Personen oder Gruppen dazu führen könnte, dass die Lösung für andere Personen oder Gruppen ungeeignet werden kann (Norman, 2005). Aktivitäten werden demnach von Menschen ausgeführt. Sie bestimmen die Bandbreite der Aktionen, die von ihnen ausgeführt werden können. ACD unterscheide sich zum HCD in einem tiefgehenden Verständnis der Aktivitäten, die von den Benutzern ausgeführt werden können, sowie der Technologie des zu entwickelnden Systems (Norman, 2005). Das ACD wird von einigen Autoren (Gifford & Enyedy, 1999; Kaptelinin & Nardi, 1997; Williams, 2009) auf die aktivitätstheoretischen Vorarbeiten von Vygotskiĭ und Leontjew zurückgeführt (Leontyev, 1978; Vygotskiĭ, 1962). Den Autoren zufolge stammt das Modell der „Hierarchischen Struktur von Aktivitäten“ von dem sowjetischen Psychologen Alexei Nikolajewitsch Leontjew. Dieses besagt, dass sich die Aktivitäten eines Menschen in eine Reihe von Aktionen unterteilen lassen. Diese wiederum werden durch Operationen ausgeführt. In Kapitel 2.3.1.2 werden einige der für das UE relevanten Aspekte dieser Aktivitätstheorien noch eingehender betrachtet.

In der Literatur lassen sich kaum detaillierte Prozessbeschreibungen oder Methoden finden, ACD in der Praxis anzuwenden (Williams, 2009). Der Grund dafür könnte sein, dass ACD nicht als eigenständiger und durchgehender Prozess zu verstehen ist, sondern als „leichte Modifikation des HCD“.

> *"I consider activity structure to be a refinement of HCD."* (Norman, 2006, S. 63)

Lediglich die Erweiterung der „hierarchischen Struktur von Aktivitäten“ nach Leontjew um die Komponente der Aufgabe (task) durch Norman sei an dieser Stelle erwähnt (Norman, 2005). Nach dieser unterteilt sich eine Aktivität wie folgt (Beispiel frei übersetzt nach (Martin, 2008)):

- *Activity*: Musik während der Autofahrt hören
- *Tasks*: Wähle einen Radiosender, spiele eine CD, höre Musik von der Playlist des iPods
- *Actions*: (Wähle einen Radiosender) Schalte das Radio ein, suche nach Sendern
- *Operations*: (suche nach Sendern) Drücke einmal den Suchen-Knopf des Radios

ACD muss also zum derzeitigen Stand als theoretisches Konstrukt angesehen werden, welches sich in der Praxis noch beweisen muss. Für eine umfassende Dokumentation aktueller Verfahren aus dem Bereich des UEs sollte das ACD jedoch nicht fehlen.

2.2.6 Usage-centered Design

1999 stellten Constantine und Lockwood ein Vorgehen vor, welches sich in einem wesentlichen Punkt von dem des „Human-centered Designs“ unterscheidet. Das „Usage-centered Design“ geht davon aus, dass nicht primär der Benutzer im Fokus einer Entwicklung stehen sollte, sondern die Aufgaben und Aktivitäten, die dieser zu erledigen beabsichtigt (Constantine & Lockwood, 1999). Für das Verständnis des menschlichen Nutzens eines Produktes bedient sich das usage-centered Design der Ansätze und des Vokabulars aus dem Bereich der Tätigkeitstheorie (Constantine, 2011). Constantine beschreibt das grundlegende Prinzip der Tätigkeitstheorie wie folgt:

> *“The essentials of activity theory can be summarized by a couple of simple diagrams supported by brief explanations.”* (Constantine, 2009)

Eine genaue Definition zur Tätigkeitstheorie wird in Kapitel 2.3.1.2. behandelt. Aktivitäten finden immer in einem sozialen Kontext statt und werden durch unterschiedliche Verantwortlichkeiten (Rollen) und durch soziokulturelle und prozedurale Faktoren (Regeln) reglementiert.

Das Usage-centered Design kombiniert diese aktivitätstheoretischen mit modellgetriebenen Ansätzen, um daraus ein methodisches Grundgerüst für ein praktisch anwendbares Verfahren für die Softwareentwicklung abzuleiten. Designern soll so in konsistenter Weise der notwendige Kontext der Tätigkeit verdeutlichet werden, der zukünftig durch die Software unterstützt werden soll. Zu diesem Zweck werden in dem Prozess von Constantine und Lockwood in jedem Schritt verschiedene Modelle erstellt.

- Role Model
- Task Model
- Content Model

Jedes dieser Modelle besteht aus zwei Teilen, einer Übersichtskarte der zu untersuchenden Faktoren sowie deren Wechselbeziehung und einer kurzen textuellen Beschreibung dieser (Constantine & Lockwood, 1999, S. 30).

Role Model

Nach Constantine sei der einfachste Weg, eine Aktivität zu modellieren, erst einmal einen Plan des „Spielfeldes“ mit allen Teilnehmer anzufertigen (Constantine, 2006a). In diesen können sowohl Beziehungen zueinander sowie die für die Aktivitäten benötigten Artefakte aufgeführt werden. Zu diesem Zweck wurden eine Reihe unterschiedlicher Visualisierungsformen von Constantine entwickelt. Eine mögliche Darstellung des systemrelevanten Kontextes mit den Akteuren und den jeweiligen Rollen ist in Abbildung 8 zu sehen. Eine solche „Context Map“ mit allen Beteiligten, würde für den weiteren Verlauf einen guten Überblick der Anwendungsdomäne geben. Für den Designprozess sei es wichtig, zwischen direkt und indirekt mit dem System interagierenden Akteuren zu unterscheiden.

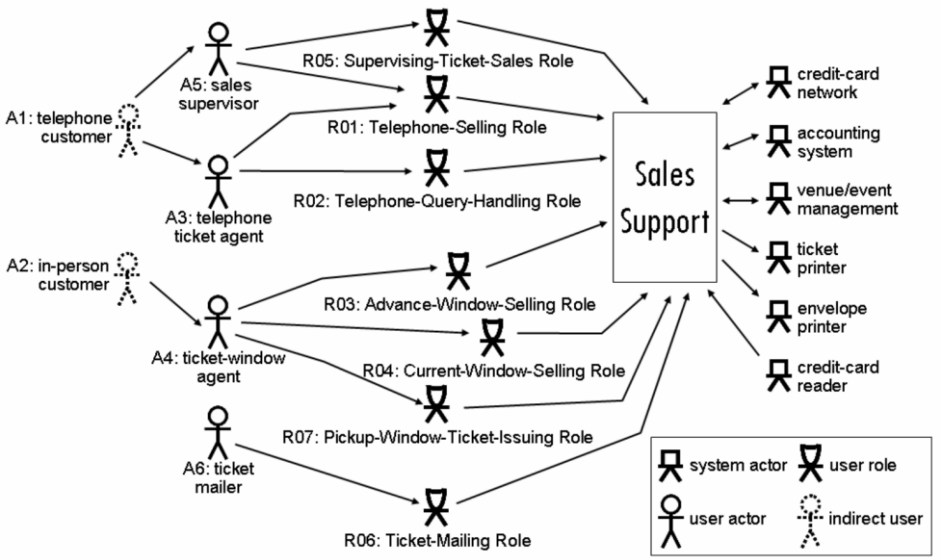


Abbildung 8: Beispiel einer "Context Map" (Constantine, 2006b)

Tätigkeitstheoretisch würden die Aktivitäten der Akteure aus Sicht einer Rolle ausgeübt werden. Anders formuliert bestimmt die Rolle eines Akteurs, die von ihm ausgeübten Aktivitäten. Die Rolle sei eine abstrakte Repräsentation der Beziehung zwischen dem Benutzer und dem System. Eine Rolle könne dabei durch folgende Punkte beschrieben werden:

- *Aktivitäten***,** die von der Rolle ausgeführt werden, sollen kurz in Bezug auf Zweck, Ort (physischen oder sozialen Kontext), Zeit sowie beteiligte Personen oder Artefakte beschrieben werden.
- *Hintergrundeigenschaften* sind beispielsweise Erfahrungen, Ausbildungsstand, Systemwissen, Fachwissen, Kompetenzen und Haltungen des Anwenders.
- *Leistungsmerkmale* wie Frequenz, Regelmäßigkeit, Intensität, Komplexität und Berechenbarkeit der Leistung.
- *[Design]* als optionale vierte Rubrik für offensichtliche Designideen, welche die Rolle wirksam unterstützen, könnten verwendet werden.

Die Beschreibung solle dabei nicht umfassend sein. Ziel des Usage-centered Designs sei es, effektiv die wichtigsten Informationen für die Entwicklung zu beschreiben.

Task Model

Eine detaillierte Beschreibung der Benutzeraktivitäten geschieht in dem sogenannten Performance-Model. Dieses besteht, erneut aus einer Übersicht wie in Abbildung 9 dargestellt und einer detaillierten Beschreibung einzelner Aufgaben in Form sogenannter Essential Use Cases.

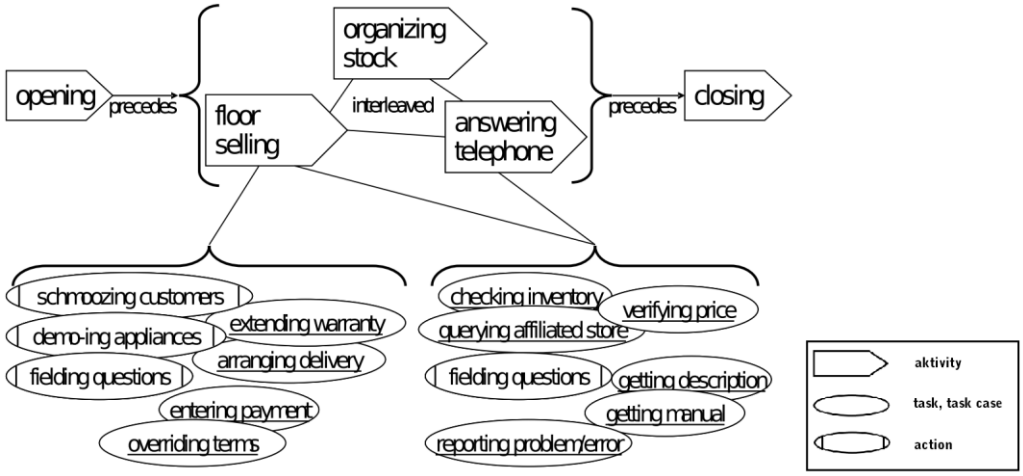


Abbildung 9: Überarbeiteter Ausschnitt einer Activity-Task Map (Constantine, 2009)

Die Übersichtskarte lässt sich in zwei Bereiche aufteilen. In der oberen Hälfte werden die Aktivitäten in einer Prozessreihenfolge dargestellt, wie sie üblicherweise durchlaufen wird. Da eine Aktivität eine Aggregation von Aufgaben beschreibt, können diese den Aktivitäten zugeordnet werden. Constantine unterscheidet zwei Arten von Aufgaben. Interaktionen zwischen System und Benutzer werden durch normale Ellipsen dargestellt. Aufgaben, die ohne Systemverwendung auskommen, werden in vergitterten Ellipsen abgebildet. Aufgaben stehen in der hierarchischen Aufteilung (Abbildung 14) auf zweiter Ebene und können somit nach Bedarf weiter verfeinert werden. Die dafür verwendeten „Essential Use Cases“ beschreiben dabei relevante Aufgaben durch ihre einzelnen Operationen detaillierter.

Task Model durch Essential Use-Cases

Mit „Essential Use Cases” oder auch „Task Cases“ bezeichnet Constantine eine generalisierte und vereinfachte Form von Interaktionsszenarien (siehe Kapitel 2.3.3.1) (Constantine, 1995, 2006b). Zur Verdeutlichung des Unterschieds konventioneller „Use Cases“ und „Essential Use Cases“ führt er das Beispiel „Bargeld an einem Geldautomaten abheben“ auf. In der konventionellen Form würde eine Folge von unterschiedlichen Interaktionen zwischen Benutzer und System beschrieben werden. Dies könnte folgendermaßen aussehen: Der Benutzer steckt seine Karte in den Geldautomaten; das System liest den Magnetstreifen; das System erfragt den Pin; das System

überprüft den Pin über ein Netzwerk; das System fordert den Benutzer auf, ein Konto zu wählen; usw. bis schließlich der Automat den gewünschten Betrag an den Benutzer auszahlt. Um aus diesem Use Case einen „Essential Use Case“ zu bilden, muss als erstes gefragt werden, warum der Benutzer den Geldautomaten verwendet. Für den Benutzer dieses Use Cases könnte beispielsweise der Hauptgrund darin liegen, Bargeld zu erhalten. Alles andere ist für den Benutzer überflüssig oder sogar störend. Wird die Frage nach dem „Warum“ für jeden Interaktionsschritt weitergeführt, würde ein „Essential Use Case“ entstehen. Für den konkreten Fall würde dies bedeuten: Warum steckt der Benutzer seine Karte in den Geldautomaten? Antwort: Es dient der Identifikation des Benutzers. Warum gibt der Benutzer eine PIN ein? Antwort: Um den Besitz der Karte zu verifizieren. Dies würde dann zu folgendem „Essential Use Case“ führen:

Tabelle 1: Essential Use Case bzw. Task Case (Constantine, 2006b)

Benutzer Aktion	System Reaktion
	1. Fordert Identifikation an
2. Liefert Identifikation	3. Verifiziert Identifikation
	4. Genehmigt Identifikation
5. Wählt Betrag	6. Zahlt Betrag aus
7. Nimmt Geld	

Der Vorteil dieser Form von Use Cases liegt in der höheren Abstraktion der Interaktionsfolge. Sie erleichtert das Aufspüren neuer Herangehensweisen. Im Unterschied zu klassischen „Use Cases“ (Jacobson, 1992) beschreiben „Essential Use Cases“ nicht eine Folge von Interaktionsschritten (siehe Definition in Kapitel 2.3.3.1), sondern die Intentionen des Benutzers. So könnte beispielsweise das „Identifizieren des Benutzers“ auch durch einen Fingerabdruck oder einen Retinascan realisiert werden.

Content Model

Im Usage-centered Design bilden die Content Models eine Brücke zwischen den Task Models und dem User-Interface-Design (Constantine, 2006b, S. 12). Das Content Model besteht aus abstrakten Prototypen, die sich aus den Task Models ableiten lassen. Diese wiederum können den Benutzerrollen zugeordnet werden. Auf diese Weise hängen im Usage-centered Design alle Informationen, angefangen bei den Benutzern über die Rollen, die sie einnehmen, bis hin zu dem Grund einer auszuführenden Aufgabe zusammen.

2.2.7 Usability Engineering Lifecycle (Mayhew)

Auch Deborah J. Mayhew (Mayhew, 1999) bezeichnet ihr Vorgehensmodell wie Jacob Nielsen als „Usability Engineering Lifecycle“. Das von ihr beschriebene Vorgehen ist stark prozessorientiert und beschreibt eine genaue Abfolge von Aktivitäten (siehe Abbildung 10), die zum Ziel der Gebrauchstauglichkeit eines Softwareprodukts führen sollen. Der von ihr beschriebene Lebenszyklus lässt sich grob in drei unterschiedliche Phase einteilen:

1. Requirements Analysis
2. Design/ Testing/ Development
3. Installation

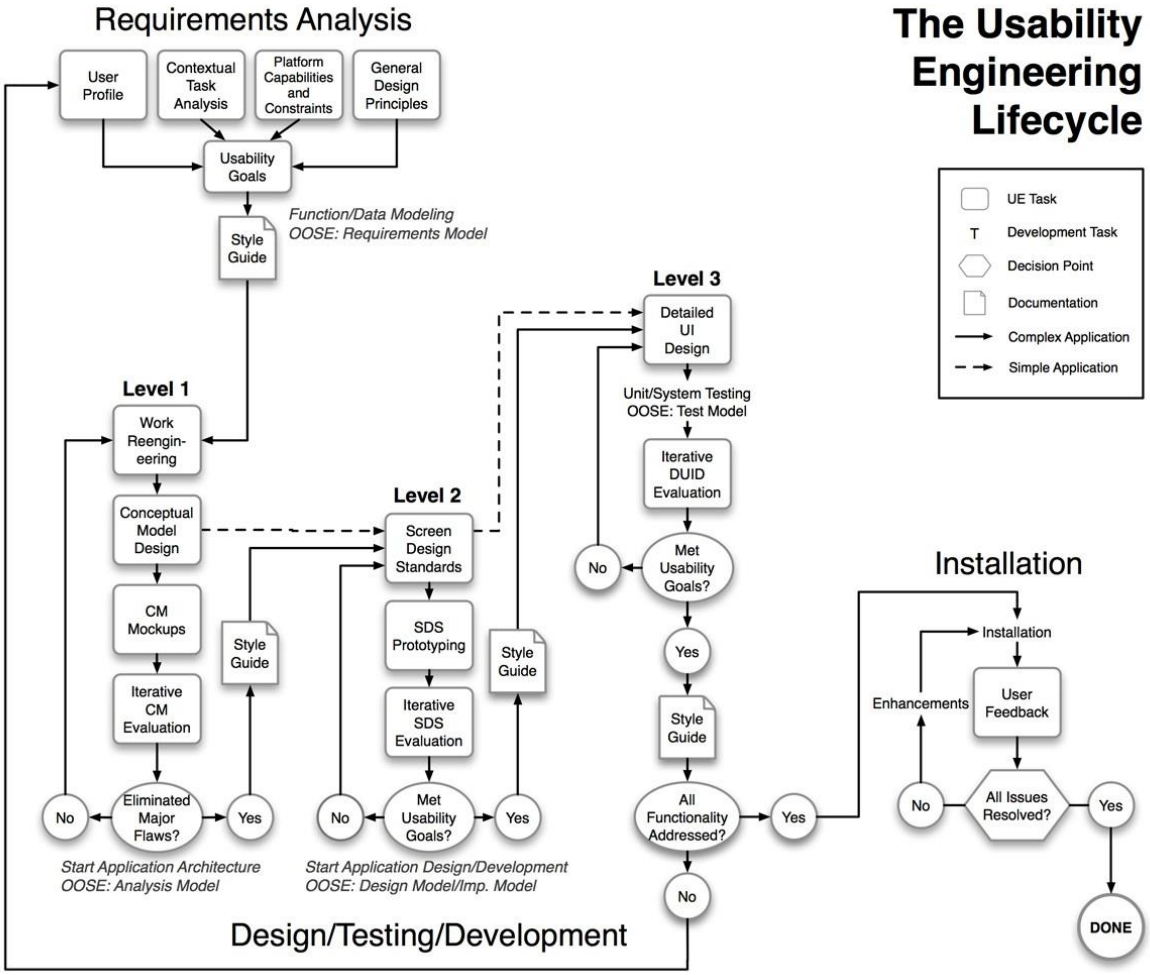


Abbildung 10: Usability Engineering Lifecycle von Deborah Mayhew (www.thinkbrownstone.com)

Requirements-Analyse

Auch Mayhew beginnt den Prozess mit einer *Benutzeranalyse* (User Profile). Der erste Schritt ist die Ermittlung aller späteren Anwender und ihrer Charakteristiken. Hierzu gehören beispielsweise Computerkenntnisse und die erwartete Häufigkeit der Nutzung. Die nach Möglichkeit breit angelegten Umfragen sollten in mehreren Rollenbeschreibungen resultieren. Die Benutzeranalyse dient im weiteren der Kategorisierung der anschließenden *kontextuellen Aufgabenanalyse* (Contextual Task Analysis). In dieser werden derzeitige Aufgaben, Arbeitsabläufe und die dahinterliegenden Benutzerziele aufgenommen. Die Ziele dienen als Grundlage der *Usability Goals* und fördern den Prozess der *Neugestaltung der Arbeit* (Work Re-engineering). Die Entscheidung für eine technische Plattform, wie das zu unterstützende Betriebssystem sowie die Möglichkeiten und Einschränkungen eines Designparadigmas, wie bei einem Menü-Masken-System oder einem direkt manipulierbarem System, sollten ermit-

telt und dokumentiert werden. Diese Arbeiten fallen in dem Prozessdiagramm unter den Punkt *Platform Capabilities and Constaints. Generelle Design Prinzipien* und Richtlinien werden gesammelt, ausgewertet und gemeinsam mit allen weiteren Informationen in einem Dokument für die Entwicklung festgehalten.

Design/ Testing/ Development

Die nächste Phase unterteilt sich in drei verschiedene Stufen. In der ersten Stufe sollte die *Arbeit Neugestalten* (Work Re-engineering) durchgeführt werden. Ausgehend von den Daten aus der ersten Phase (Requirements Analysis) werden organisatorische Bedingungen und Arbeitsabläufe optimiert. Hierbei können vor allem die technischen Automatisierungsmöglichkeiten ausgenutzt werden. Es sollen jedoch in dieser Stufe explizit noch keine Benutzungsschnittstellen entwickelt werden. Erst im nächsten Schritt werden verschiedene *konzeptuelle Designmodelle* (Conceptual Model Design) erstellt. Ausgehend von den Aufgaben werden mögliche Navigationspfade, Hauptdialoge und konsistente Präsentationsmöglichkeiten identifiziert. Diese können aus groben Skizzen und kurzen Beschreibungen bestehen. Details der Benutzungsschnittstelle sollen an dieser Stelle noch nicht entwickelt werden. Das gilt auch für den nächsten Schritt, der Erstellung von *konzeptuellen Modell-Mockups* (CM Mockups). Einige der besten zuvor erstellten Designmodelle sollen so aufbereitet werden, dass sie als Mockup oder einfacher Prototyp im nächsten Schritt formativ evaluiert werden können. Sehr grobe Mängel können sogar mit einfachen Papier-Mockups aufgedeckt werden (Mayhew, 2010, S. 220). Jedoch auch Werkzeuge für die Erstellung von Prototypen könnten hier hilfreich sein. Ziel des nächsten Schrittes ist eine auf lange Sicht kosteneinsparende Maßnahme. Durch frühzeitige *iterative Evaluation der konzeptuellen Modelle* (Iterative CM Evaluation) können kostenintensive Änderungen in späteren Phasen eingespart werden. Unter Zuhilfenahme der zuvor erstellten Konzeptmodelle und des Einbezugs repräsentativer Endanwender sollen in diesem Schritt verschiedene Usability Tests durchgeführt werden. Dabei sollen die Endanwender die zuvor identifizierten repräsentativen Aufgaben mit einem möglichst geringen Trainingsaufwand und wenigen Eingriffen ausführen können. Das Vorgehen der ersten Stufe sollte so häufig iteriert werden, bis alle Usability Fehler aufgedeckt und behoben worden sind. Erst dann kann mit der zweiten Stufe fortgefahren werden.

Die zweite Stufe beschäftigt sich überwiegend mit der Festlegung von Standards. Im ersten Schritt werden die produktspezifischen *Screen Design Standards* und Konventionen entwickelt. Sie dienen dem kohärenten und konsistenten Erscheinen der gesamten Anwendung. Diese Screen-Design-Standards werden im nächsten Schritt (*Screen Design Standards Prototyping*) an der detaillierten Benutzungsschnittstelle angewendet. Es sollte dabei ein verwendbarer Prototyp entstehen, an dem sich weitere ausgewählte Teilkomponenten des Systems und ihre Funktion überprüfen lassen. Das Vorgehen bei der *Evaluation der Screen-Design-Standards* kann mit den gleichen Methoden wie bei der Evaluation der konzeptuellen Modelle durchgeführt werden. Mayhew schlägt die Einbeziehung von drei bis zehn repräsentativen Endanwendern vor, die relevante Aufgaben mit möglichst realem Kontextbezug durchführen sollen. Dieses Vorgehen wird erneut so häufig iteriert, bis alle Mängel beseitigt wurden. Als Ergebnis der ersten beiden Stufen stehen somit sowohl ein validiertes und stabiles konzeptuelles Modell der Anwendung als auch generelle Screen-Design-Standards mit detaillierten Designvorstellungen für eine Reihe verschiedener Produktfunktionalitäten. Alle bisherigen Ergebnisse sollten in einem Dokument festgehalten werden. Mayhew bezeichnet dieses als *Styleguide*. In dieses Dokument sollten auch die Informationen aus der Requirements-Analyse stehen. Das Dokument dient in der nächsten Stufe als Grundlage für das detaillierte Benutzungsschnittstellendesign.

Das verfeinerte konzeptuelle Modell und die Screen-Design-Standards ermöglichen in der dritten Stufe das Erstellen eines *detaillierten Benutzungsschnittstellendesigns* (Detailed User Interface Design). Dieses wird, wie bei den vorherigen Stufen, so häufig verfeinert und evaluiert (*Iterative Detailed User Interface Design Evaluation*), bis alle Funktionalitäten berücksichtigt wurden. Es sollte dabei überprüft werden, ob alle zuvor aufgenommenen Usability Ziele erreicht wurden. Die Designergebnisse fließen daraufhin wieder in das Style-Guide-Dokument.

Installation

Die letzte Phase des UE Lifecycle bezieht sich auf die Zeit nach der Fertigstellung und der Installation beim Kunden. Hier ist es wichtig, sich Informationen für Verbesserungen und Wartungsarbeiten von den Anwendern einzuholen. Die Informationen können auch für eine neue Erweiterung genutzt werden, oder um für zukünftige Projekte mehr über die Reaktionen der Kunden über das UE-Verfahren zu lernen.

2.3 Methoden des UE

Im Rahmen dieser Arbeit wurde angestrebt einzelne Methoden aus dem UE zu identifizieren, die sich für die Integration der zu entwickelnden Werkzeugunterstützung UsER eignen könnten. Die im Folgenden vorgestellten Methoden können daher gewissermaßen als methodische Bausteine der am Ende vorgestellten Werkzeugunterstützung (siehe Kapitel 5) angesehen werden. Die Methoden des UE können in Analyse, Design, Entwicklung und Evaluation eingeteilt werden.

2.3.1 Analyse

Wie aus ISO 9241-11 und ISO 9241-210 hervorgeht, ist im UE für die Analyse des Nutzungskontextes ein umfassendes Verständnis der Benutzermerkmale, Arbeitsaufgaben sowie die organisatorische, technische und physische Umgebung notwendig, in der das zu entwickelnde System eingesetzt werden soll. Die nachfolgenden Kapitel beleuchten die hierzu im UE etablierten Methoden.

2.3.1.1 Benutzeranalyse

Bereits bei der Planung eines Systems stellt sich die Frage, an wen sich das System richtet bzw. für wen es einen Sinn erfüllen soll. Es gilt die benutzerspezifischen Eigenschaften der späteren Benutzer zu erkennen und diese bei der Entwicklung des Systems zu berücksichtigen. Diese Eigenschaften umfassen: Absichten, Bedürfnisse, Fähigkeiten und Motivationen und werden häufig als Benutzercharakteristik bezeichnet (Herczeg, 2009). Als *Benutzer* verstehen wir im Folgenden die

> *„Person, die mit dem Produkt arbeitet" (ISO 9241-210:2011)*

Den Benutzer zu verstehen, wird immer wieder als das wichtigste Prinzip des UEs angesehen (Nielsen, 1993; Shneiderman, 1998). Die ISO-9241-210:2010 „Prozess zur Gestaltung gebrauchstauglicher interaktiver Systeme" weist darauf hin, dass die Hauptursache für den Misserfolg bei der Entwicklung von Systemen, auf einem unangemessenen und unvollständigen Verständnis der Erfordernisse der Benutzer beruht.

Ein vor allem von Cooper, Reimann und Cronin diskutierter Punkt bei der Benutzeranalyse sind die Benutzerziele. So beschreibt beispielsweise Cooper gleich zu Anfang seines Buches „About Face 3.0".

> *„If we design and construct products in such a way that the people who use them achieve their goals, these people will be satisfied, effective and happy, and will gladly pay for the products and recommend that others do the same."* (Cooper, Reimann & Cronin, 2007)

Das Ziel der Benutzeranalyse ist es, Software zu entwickeln, welche die Ziele, Wünsche und Fähigkeiten der Benutzer berücksichtigt. Darüber hinaus muss eine Form gefunden werden, die erhobenen Informationen über die Benutzer an alle am Entwicklungsprozess beteiligten Personen weiter zu kommunizieren, damit diese bei der Entwicklung der Software berücksichtigt werden können. Eine weitere Schwierigkeit ergibt sich durch den Umstand, dass Software in der Regel von einer Vielzahl unterschiedlicher Benutzer angewendet wird. Das kann dazu führen, dass bei jeder Designentscheidung die Entwickler sich mehr oder weniger bewusst einen Benutzer vorstellen, für den die gewählte Lösung angemessen wäre. So entstehen Gesamtlösungen, die nach vielen Designentscheidungen zu keinem realen Menschen mehr passen. Cooper spricht bei diesem Phänomen vom sogenannten *Elastischen Benutzer*.

> *"When it comes time to make product decisions, this "user" becomes elastic, conveniently bending and stretching to fit the opinions and presuppositions of whoever's talking."* (Cooper et al., 2007)

Bei Designentscheidungen stellen Entwickler eigene Vorlieben mit denen des Benutzers gleich. Cooper bezeichnet dies als *Selbsreferenzielles Design*. Dieses tritt auf, wenn Designer oder Entwickler ihre eigenen Ziele, Motivationen, Fähigkeiten und mentalen Modelle auf das Produktdesign projizieren (Cooper et al., 2007). Cooper veranschaulicht dies in Form der Vorlieben unterschiedlicher Autofahrer. Jeder Benutzer oder in diesem Beispiel die Autofahrer haben völlig unterschiedliche Ziele. Die Junganwältin „Alessandra" wünscht sich ein schnelles und spaßbringendes Auto, wohingegen der Familienvater „Stefan" für die Beförderung seiner Kinder eher eines

sicheren und komfortablen Autos bedarf. Der Landwirt „Hendrik“ hingegen benötigt ein zuverlässiges Auto, in welchem große Ladungen transportiert werden können.

Würden die Designer und Entwickler nun versuchen, diese Ziele und Wünsche der einzelnen Personen zusammenzufassen, könnte ein uneinheitliches Auto wie in Abbildung 11 dargestellt entstehen.

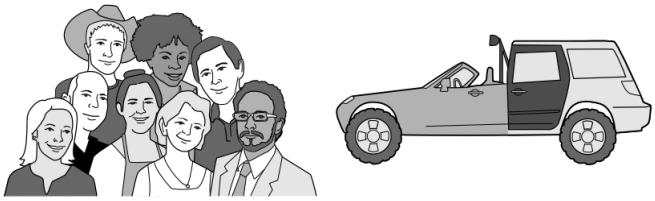


Abbildung 11: Zusammenführung aller Wünsche und Ziele (Quelle: Cooper et al., 2007)

Dieses Problem lässt sich auch auf die Softwareentwicklung übertragen. Auch hier haben die Benutzer unterschiedliche Zielvorstellungen des zu entwickelnden Systems. Bei der Softwareentwicklung sind darüber hinaus jedoch auch die Fähigkeiten, wie beispielsweise der Umgang mit anderen Systemen oder Programmen, sowie die zu erledigenden Aufgaben, zu berücksichtigen. Um die Vielzahl an Informationen der verschiedenen Benutzerindividuen den Designern und Entwicklern zur Verfügung stellen zu können, müssen die Informationen zusammengefasst und zugänglich gemacht werden, damit nicht das Phänomen des „Elastischen Benutzers“ entsteht. Es gilt, die Informationen zu klassifizieren und daraus ein einheitliches Bild des Benutzers zu entwickeln, welches von allen am Entwicklungsprozess beteiligten Personen verwendet wird. Zu diesem Zweck wurden im Laufe der Zeit eine Reihe unterschiedlicher Modelle entwickelt.

John Pruitt und Tamara Adlin haben hierzu in ihrem Buch – The Persona Lifecycle – eine übersichtliche Grafik der verschiedenen Benutzerrepräsentationen aufgestellt (Abbildung 12).

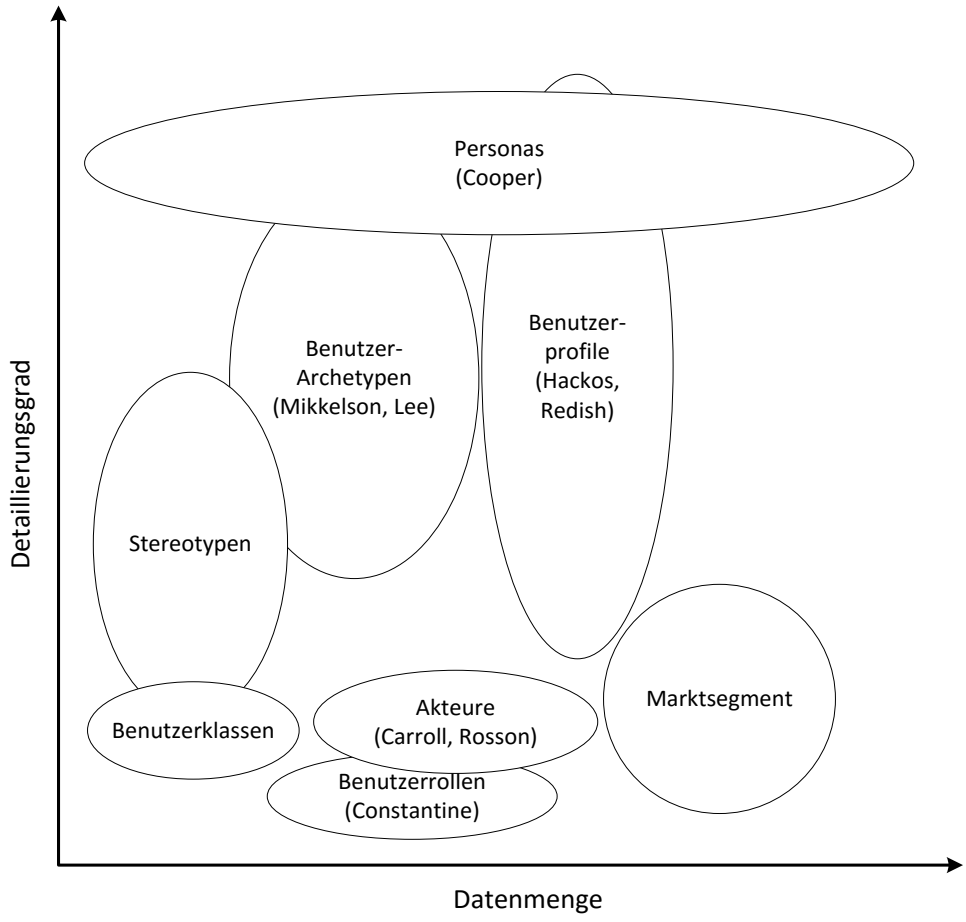


Abbildung 12: erweiterte Benutzerklassifikationen (Pruitt & Adlin, 2006)

Die Beschreibungen der Klassen können unterschiedlich stark ausgeprägt und damit unterschiedlich aufwändig in der Erstellung und Wartung der Informationen sein. Es sollte somit gut überlegt sein, welche Form der Beschreibung für ein Entwicklungsprojekt genutzt werden sollte oder kann. Auch die Etablierung der Benutzerbeschreibung im Designer- oder Entwicklerteam sollte geklärt werden.

Eine sehr einfache Form der Kategorisierung sind *Benutzerklassen*. Diese beschreibt eine Menge „ähnlicher" Benutzer. Ähnlichkeiten bestehen beispielsweise:

- in Aufgaben und Rollen
- im Erfahrungsstand
- in der Herkunft
- im Marktverhalten
- in der Altersgruppe

Eine Benutzerklasse liefert zwar eine Vorstellung von den Benutzern, diese ist jedoch sehr abstrakt. Sie beinhaltet nur wenig Details und beruht häufig nur auf Annahmen bzw. logisch definierten Charaktereigenschaften (Pruitt & Adlin, 2006). Bei konkreten Fragen zu Konzeption und Gestaltung ist es schwierig, mit ihnen zu arbeiten.

Anstatt unterschiedliche Benutzer in einer Benutzerklasse zusammenzufassen, können diese auch in Form eines *Stereotypen* beschrieben werden. Ein Stereotyp ist ein „fiktiver" Stellvertreter einer Benutzerklasse (Herczeg, 2009) und kann somit dem Problem des „Elastischen Benutzers" entgegenwirken. Da Stereotypen jedoch auf Annahmen und Vorurteilen der Designer und Entwickler beruhen, laufen sie Gefahr, mit der Zeit zu Karikaturen zu verkommen (Cooper et al., 2007). Um dies zu verhindern ist es wichtig, Benutzerbeschreibungen sorgfältig und zweckgerecht zu erstellen, damit Designer und Entwickler ein Interesse für die Personen entwickeln, für die das Produkt bestimmt ist. Detailliertere und somit auch greifbarere Beschreibungen können durch Benutzer-Archetypen (Mikkelson & Lee, 2000) oder Benutzerprofile (Hackos & Redish, 1998) erstellt werden. Sie unterscheiden sich hauptsächlich in der verwendeten Datenmenge. Mikkelson schlägt vor, aus den Informationen der Benutzerklasse einen konkreten jedoch fiktiven *Benutzer-Archetyp* mit nachfolgend aufgelisteten Eigenschaften abzuleiten:

- Beschreibung: Name, Bild und eine einzeilige Beschreibung
- Attribute: Alter, Familienstand, Lebensstil, Rollen und Interessen
- Erfahrungen: Umgang mit dem Computer
- Ziele und Bedürfnisse
- Marktgröße: Dominanz und Relevanz des Benutzer-Archetyps
- Aktivitäten: Aufgaben und kontextbezogenes Anwendungswissen

Hackos und Redish (1998) empfehlen die Erstellung von *Benutzerprofilen*. Dazu werden einige Eigenschaften der real befragten Personen in eine Tabelle eingetragen und anschließend eine oder mehrere als Repräsentanten für die Systementwicklung ausgewählt.

Eine in der Softwareentwicklung häufig verwendete Klassifizierung von Benutzerinformationen ist die Beschreibung eines Benutzers durch seine *Rolle*.

> *„Eine Rolle stellt eine abstrakte Qualifikation eines Akteurs dar."* (Rosemann & Mühlen, 1996)

Rollen „[…] und die damit verbundenen Funktions- oder Aufgabenbeschreibungen von Mitarbeitern […]" (Herczeg, 2009, S. 82) eignen sich insbesondere zur Klassifizierung von Benutzereigenschaften in einem betrieblichen Kontext. Benutzerrollen können auf sehr umfangreichen Informationen aufbauen. Nach Constantine sei eine Benutzerrolle eine Kollektion von charakteristischen Bedürfnissen, Interessen, Erwartungen und dem Verhalten zu einem bestimmten System (Constantine, 2006b). Dennoch ist die Beschreibung sehr abstrakt.

Die Bezeichnung einer Benutzerbeschreibung als *Persona* führte 1999 erstmals Alan Cooper ein (Cooper, 1999). Sie beschreiben einen Benutzer in einer narrativen Form, welche auf vorhergehenden umfangreichen Studien realer Menschen beruhen. Persona unterscheiden sich zu anderen Formen der Benutzerbeschreibungen dadurch,

dass sie für den Betrachter „echt“ wirken. Dabei können sie – wie es sich auch häufig im realen Leben widerspiegelt – in verschiedene Typen von Stakeholdern kategorisiert werden. Folgende Kategorien können nach Cooper unterschieden werden:

- *Primary Persona*: Entspricht dem tatsächlichen späterem Anwender des zu entwickelnden Systems. Sie sind die Benutzer, nach deren Bedürfnissen ein System entwickelt werden sollte. Bei mehreren Primary Persona sollte überlegt werden, ob nicht für jede ein einzelnes Interface erstellt werden sollte.
- *Secondary Persona*: Solang die Belange dieser Persona nicht die der Primary Persona negativ beeinflussen, sollte diese Anforderungen bei der Entwicklung auch berücksichtigt werden.
- *Supplemental Persona*: Können für ein besseres Verständnis des Kontextes mit aufgenommen werden, steuern jedoch keine neuen Anforderungen mit bei.
- *Customer Persona*: Durch sie werden die Kunden beschrieben, die jedoch häufig die Anwendung nicht selbst nutzen werden.
- *Served Persona*: Sind die Personen, die mithilfe des Systems bedient werden.
- *Negativ Persona*: Beschreibt den Personenkreis, für den das System explizit nicht entwickelt wird.

Personas bestimmen, was ein Produkt können muss und wie es sich dabei verhalten soll. Ein nach Cooper wichtiger Bestandteil einer Persona sind die *Benutzerziele* und deren Aufgaben. Erst durch die Benutzerziele würde einer Benutzerbeschreibung – bei Cooper auf die Erstellung von Persona bezogen – wahres Leben eingehaucht werden können (Cooper et al., 2007). Wichtig sei dabei die Unterscheidung von Zielen, Aufgaben und Aktivitäten. Ein Ziel sei die Erwartung eines Endzustandes, wohingegen Aufgaben und Aktivitäten Zwischenschritte auf dem Weg zur Erreichung dieses Endzustandes bzw. Zieles seien. Benutzerziele würden sich nur selten ändern. Aufgaben und Aktivitäten hingegen hängen von unterschiedlichen Faktoren ab. Um diesen Sachverhalt zu verdeutlichen, bedient sich Cooper des Beispiels zweier Reisender, die beide von einem Ort zu einem anderen reisen möchten. Das Ziel beider Reisenden sei es, schnell, komfortabel und sicher zu reisen. Bei einem Siedler aus dem Jahre 1850 würde schnell und komfortabel vermutlich bedeuten, mit einem überdachten Wagen zu reisen, und für die Sicherheit würde er noch einen Revolver mit sich führen. Für einen Geschäftsmann aus der heutigen Zeit würde es bedeuten, in einem Business-Jet zu fliegen und keine Waffe zu benötigen.

> *“If Persona provides the context for sets of observed behaviors, goals are the drivers behind those behaviors. A persona without goals can still serve as a useful communication tool, but it lacks utility as a design tool. User goals serve as a lens through which designers must consider the functions of a product. The function and behavior of the product must address goals via tasks—typically, as few tasks as absolutely necessary. Remember, tasks are only a means to an end; goals are that end.”* (Cooper et al., 2007, S. 88)

Eine Persona sollte nach Cooper auch immer auf real basierenden Daten aufbauen, um von der Entwicklung – also den Personen, die die Persona verwenden sollen – akzeptiert zu werden. Zu diesem Zweck wurden von Cooper die sogenannten *Verhaltensvariablen* eingeführt. Im Bereich der Entwicklung von Geschäftsanwendungen hängen diese Variablen häufig eng mit den vorkommenden Geschäftsrollen zusammen. Pro Rolle sollten etwa 15 bis 30 projektrelevante Verhaltensvariablen untersucht werden. Die meisten der Variablen können aus den Untersuchungen vor Ort bzw. den Interviews entwickelt werden. Mittels dieser Informationen kann dann die Verteilung und Tendenz verschiedener Variablen und der realen Benutzer auf einer Skala wie in Abbildung 13 dargestellt werden.

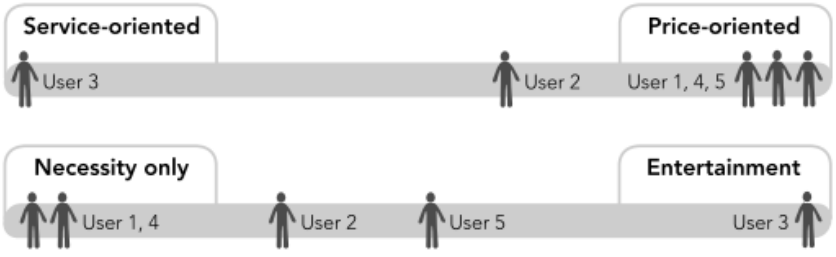


Abbildung 13: Verteilung der benutzerspezifischen Verhaltensvariablen (Cooper et al., 2007, S. 99)

Nach der Zuordnung werden signifikante Cluster jeder Verhaltensvariablen identifiziert. Sie und die jeweiligen Benutzerziele dienen dann als Grundlage der Entwicklung einer Persona-Beschreibung. Cooper weist in diesem Zusammenhang zudem darauf hin, dass die Ausformulierung nicht zu „blumig" ausfallen sollte, da sie als Design-Werkzeug dienen soll und nicht für eine Erzählung (Cooper et al., 2007, S. 100).

2.3.1.2 Aufgabenanalyse

Eines der bedeutungsvollsten Themen in der Mensch-Maschine-Interaktion ist das Verständnis über die mit dem System zu erledigenden Aufgaben. Eine *Aufgabe* kann als Folge von Handlungen verstanden werden, welche durchzuführen sind, um ein bestimmtes Ziel zu erfüllen.

> *„A task is a goal together with some ordered set of actions."* (Benyon, 2010, S. 252)

Dabei kann diese Folge fremdbestimmt sein, beispielsweise durch einen Arbeitgeber oder von dem Handelnden selbst (Paech, 2000, S. 21). Eine ganz ähnliche Definition des Begriffs der Aufgabe stammt aus der EN ISO 9241-11:

> *„Die zur Zielerreichung erforderlichen Aktivitäten." (ISO 9241-11:1998)*

Wobei in der letzten Definition als Ziel das „angestrebte Arbeitsergebnis" verstanden wird. Ein Ziel kann in der Regel auf unterschiedliche Weise erreicht werden. Bezogen auf die Definitionen kann ein bestimmtes Ergebnis durch unterschiedliche Folgen von Aktivitäten erreicht werden.

Für die Umsetzung eines gebrauchstauglichen Werkzeugs ist es für die Produktdesigner oder Entwickler notwendig die Ziele sowie die dafür nötige Handlung des Benutzers genau zu kennen. Dies gilt insbesondere für die Entwicklung komplexer Softwarelösungen, bei denen meist eine ganze Reihe verschiedener Aufgaben zu dem angestrebten Ziel führen. Eine weitere Definition verdeutlicht den wichtigen Unterschied zwischen „Aufgabe" und „Aktivitäten".

> *„Eine Aufgabe ist eine Vorgabe zum zielorientierten Handeln. Diese Vorgabe kann von außen oder von den Handelnden selbst kommen. Die Erfüllung einer Aufgabe erfolgt in Arbeitsabläufen, die in einzelnen Aktivitäten strukturiert sind. Die Ausübung von Aktivitäten erfolgt unter gewissen Voraussetzungen und liefert Ergebnisse. Die Arbeitsabläufe sind geprägt von organisatorischen Randbedingungen."* (Paech, 2000)

Aus aktivitätstheoretischer Sicht können Aktivitäten in einer hierarchischen Form verstanden werden. Aktivitäten bestehen aus einer Kollektion von Handlungen (action), die einen Zweck (Motiv) erfüllen sollen. Durch Operationen werden diese Handlungen, die der Erfüllung eines Ziels dienen, ausgeführt. Außenfaktoren (Bedingungen) können Operationen beeinflussen (Engeström, 1999).

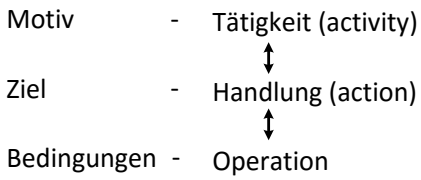


Abbildung 14: Hierarchische Struktur von Aktivitäten (Specker, 1998)

Norman stellt ein sehr übersichtliches und verständliches Schema von Handlungen des Menschen vor (Norman, 1998). Dabei unterteilt er Handlungen in sieben verschiedene Phasen - Seven Stages of Action (siehe Abbildung 15). Ausgehend von einem Ziel, das erreicht werden soll, muss dieses in die Intention zu Handeln übersetzt werden. Die Intention wird im nächsten Schritt in einen Satz von internen Aktionen umgesetzt. Zu diesem Zeitpunkt ist diese Folge von Aktionen rein mentaler Natur. Erst wenn diese Aktionen ausgeführt werden, bewirken sie eine Veränderung der Umwelt. Komplettiert wird das Modell durch die Wahrnehmung, Interpretation und Bewertung der Veränderung (Norman, 1998, S. 46). Sollte die Veränderung noch nicht der angestrebten Zielvorstellung entsprechen, werden die einzelnen Phasen erneut durchlaufen.

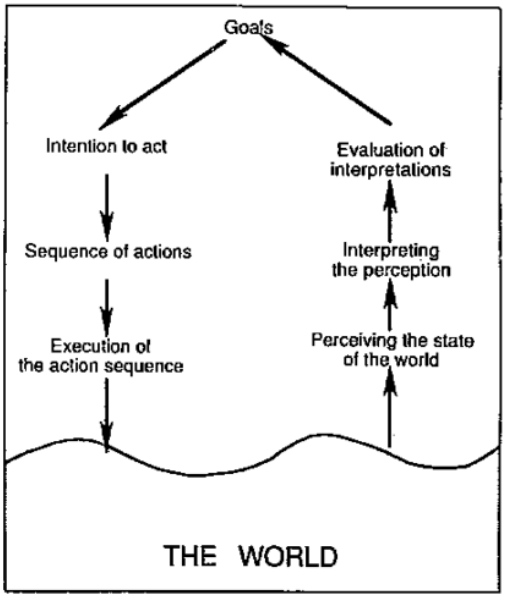


Abbildung 15: Seven Stages of Action (Norman, 1998, S. 46)

Normen weist jedoch auch auf die Möglichkeit hin, dass der Prozess in jeder Phase unterbrochen werden kann. So könne man sich das Beispiel vorstellen, jemand würde beim Lesen eines Buches merken, dass es zu dunkel sei. Das Ziel – mehr Licht zu haben – wäre entstanden. Dieses Ziel könne dann in die Intention mit einem Lichtschalter eine Lampe anzuschalten, überführt werden. Dazu müsse sich die Person ein entsprechendes Vorgehen (Sequenz von Aktionen) überlegen, um den Lichtschalter zu betätigen. An diesem Punkt (Intention to act) sei es nach Hackos und Redish (Hackos & Redish, 1998, S. 57) wichtig zu erkennen, dass sich der Benutzer an dieser Stelle für eine Aufgabe entscheidet. In den meisten Fällen gibt es eine ganze Reihe an Verfahren, das Ziel zu erreichen. Je nachdem, wo sich der Lichtschalter befindet, wird diese Folge von Aktivitäten unterschiedlich ausfallen können. So könnte sich der Lichtschalter direkt neben der Person befinden, wodurch es der Person erspart bliebe, diesen erst noch erreichen zu müssen. An dieser Stelle wird auch die Ortsbezogenheit bei der Bestimmung eines bestimmten Vorgehens deutlich, welches in Kapitel 2.3.1.3 näher beleuchtet wird. Genauso gut könnte jedoch plötzlich eine Person den Raum betreten und von sich aus den Lichtschalter betätigen. Dadurch würden sich zwei der nachfolgenden Phasen erübrigen. Für die Person des Ausgangsbeispiels würde dieses Ereignis bedeuten, dass sie direkt auf die rechte Seite des Handlungsmodells (in Abbildung 15) springen könnte und die neuen Lichtverhältnisse wahrscheinlich zum Lesen ausreichen würden. Durch den jetzt neuen Zustand, dass eine Person den Raum betreten hat, könnte die Situation eingetreten sein, dass das ursprüngliche Ziel, mehr Licht um Lesen zu können, hinfällig geworden wäre. Dieses Beispiel verdeutlicht mehrere wichtige Aspekte. Das ursprünglich formulierte Ziel – mehr Licht haben zu wollen – könnte einem eigentlich höheren Ziel gedient haben. Möglicherweise war das eigentliche Ziel der Person, sich in einer geeigneten Form zu entspannen, und die den Raum betretende Person kann diesem Ziel viel dienlicher sein. Sämtliche Umweltereignisse haben einen Einfluss auf aktuelle Zielvorstellungen und das aufgezeigte Modell von Norman wird hochfrequent durchlaufen, und Ziele müssen gelegentlich neu priorisiert werden. Das Verständnis über sich häufig ändernde Zielformulierungen von Personen sollte speziell in der Phase der Anforderungsaufnahme bei der Softwareentwicklung berücksichtigt werden. Nach Benyon können Ziele auch als Externe Aufgaben bezeichnet werden (Benyon, 2010).

In einem weiteren Modell, dem 6-Ebenen-Modell (Herczeg, 2009, S. 119), wird die Interaktion des Menschen mit einem Computersystem verdeutlicht. In diesem wird, ausgehend von einer Aufgabe eines Benutzers, schrittweise über 6 Ebenen seine Handlung detailliert (siehe Abbildung 16). Aus Sicht eines Benutzers wird in diesem eine Aufgabe mit einem Computersystem in folgende Ebenen unterteilt (Herczeg, 1994, 2009):

Intentionale Ebene: Auf oberster Ebene entwickelt ein Benutzer eine Intention bzw. Motiv für die Bewältigung einer Aufgabe. Dafür müssen verschiedene Tätigkeiten geplant werden. Entspricht die Bewertung nach Durchführung aller benötigten Tätigkeiten der Intention, so ist die Aufgabe erledigt. Andernfalls müssen die Tätigkeiten modifiziert und erneut durchlaufen werden.

Pragmatische Ebene: Für die Erfüllung der Intentionen durch verschiedene Tätigkeiten muss sich ein Benutzer verschiedene konkrete Verfahren überlegen, die ihm seine Zielerfüllung ermöglichen. Die Ergebnisse seiner Verfahren muss der Benutzer durch die Interpretation der neuen Zustände deuten.

Semantische Ebene: Die geplanten Verfahren des Benutzers werden durch Operationen bzw. durch Änderungen des Zustandes an Objekten umgesetzt. Hat ein Objekt den gewünschten Zustand eingenommen, muss keine weitere Operation auf das Objekt ausgeübt werden. Dabei ist darauf zu achten, dass die Operationen semantisch korrekt verwendet werden und gewisse Regeln einzuhalten sind.

Syntaktische Ebene: Für die Ausführung einer Operation des Benutzers an dem System muss eine bestimmte Eingabesyntax verwendet werden. Entspricht diese Eingabesyntax, beispielsweise der Doppelklick zum Öffnen einer Datei, einer entsprechenden Struktur, führt das System die entsprechende Operation aus. Der Zustand eines Objektes wird geändert.

Lexikalische Ebene: Für die Umsetzung der Operationen wird bei einem Computersystem einen Satz von Eingabezeichen verwendet. Mittels der Wahrnehmung kann ein Benutzer registrieren, ob ein Zeichen von dem Computersystem erkannt wurde.

Sensomotorische Ebene: Die verwendeten Zeichen werden über die Motorik in Form von Signalen an das Computersystem übertragen. Dies geschieht beispielsweise über Eingabegeräte wie die Maus oder Tastatur. Über Sensoren werden die Eingaben des Benutzers registriert.

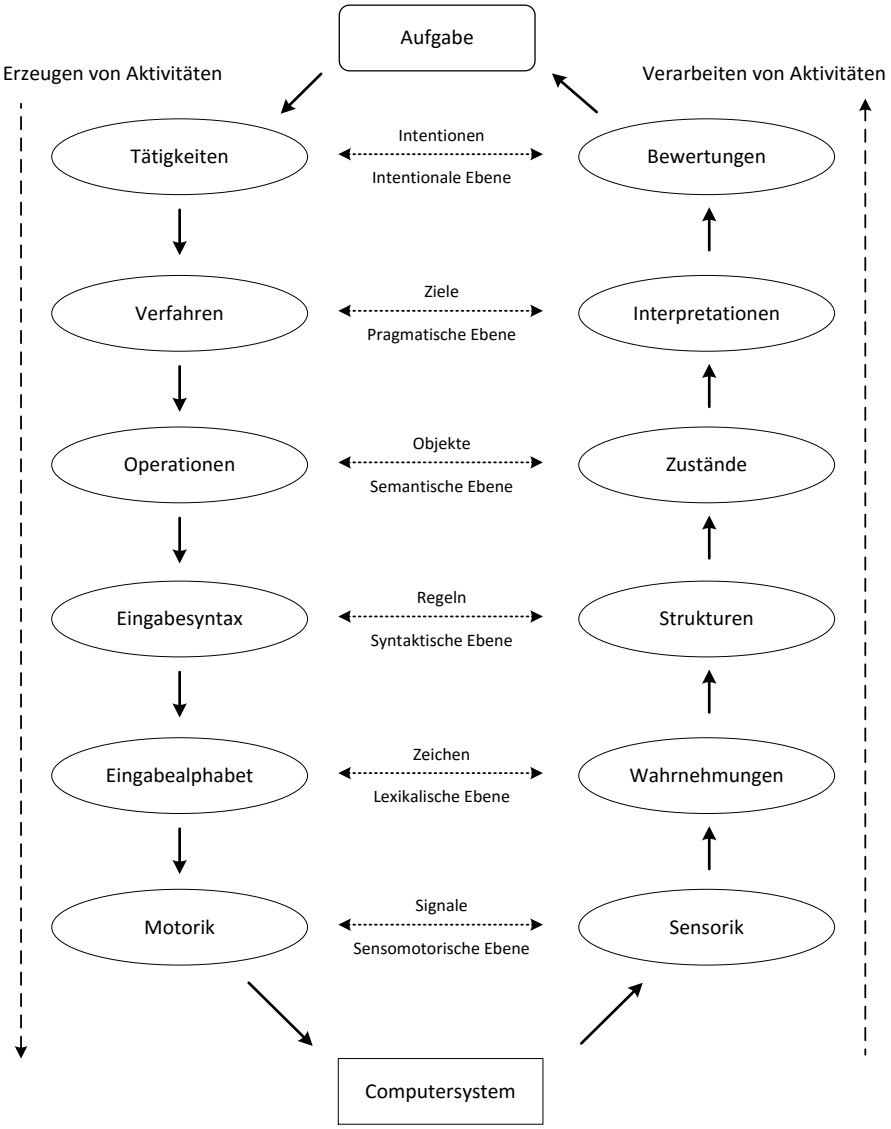


Abbildung 16: Modell der Mensch-Computer-Kommunikation (Herczeg, 1994)

Die bisherigen Modelle geben einen Überblick der verschiedenen Aspekte von Arbeit. In den meisten Verfahren aus dem Bereich des UEs (siehe Kapitel 2.2) ist ein benutzerzentriertes Verständnis der Arbeit, wie sie derzeit verrichtet wird, zwingende Voraussetzung für das weitere Vorgehen. Zu diesem Zweck muss die zu unterstützende Tätigkeit soweit verstanden sein, dass Teilschritte, Entscheidungspunkte und Zusammenhänge von allen Stakeholdern nachvollzogen werden können. Darauf aufbauend können dann Variationen oder Kombinationen untersucht werden, die diese Tätigkeit zukünftig möglichst optimal durch entsprechende Funktionalität des Computersystems unterstützt (siehe hierzu auch Kapitel 2.3.4).

2.3.1.2.1 Hierarchische Aufgabenmodelle

Neben dem Verständnis von Arbeit und deren Struktur sollte bei der Entwicklung von Software dieses Verständnis auch dokumentiert werden können. Zu diesem Zweck wurde speziell im Bereich der HCI eine Reihe unterschiedlich komplexer Modelle entwickelt. Die meisten basieren dabei auf einer hierarchischen Dekomposition von Aufgaben. Abbildung 17 zeigt eine grobe Übersicht dieser Modelle. Das Diagramm veranschaulicht darüber hinaus auch die Komplexität der einzelnen Modelle, die sich jedoch direkt auf die Aussagekraft auswirkt.

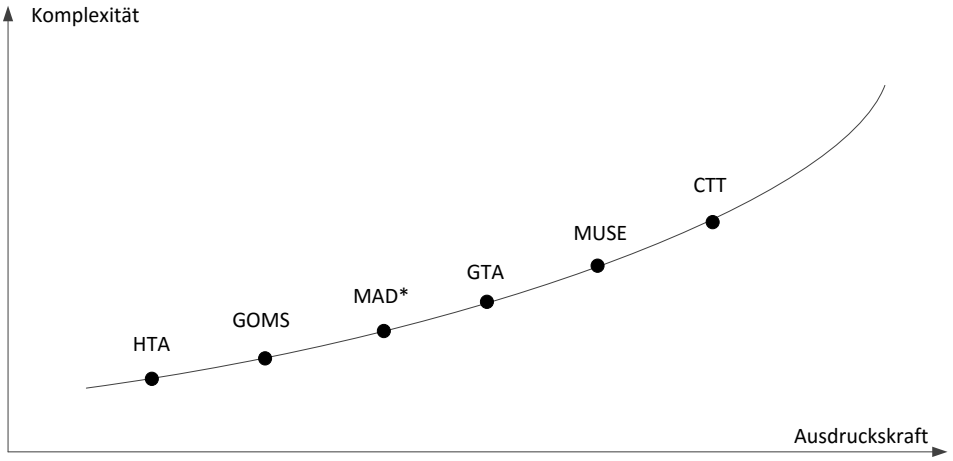


Abbildung 17: Ausdruckskraft versus Komplexität bei Aufgabenmodellen (Limbourg & Vanderdonckt, 2004)

Hierarchical Task Analysis (HTA)

Eine der ersten Tätigkeitsanalyseverfahren wird durch die Hierarchical Task Analysis (HTA) (Annett & Duncan, 1967; Diaper & Stanton, 2004; Kirwan & Ainsworth, 1992) beschrieben. In diesem wird davon ausgegangen, dass sich Handlungen in eine Hierarchie von Aufgaben aufteilen lassen. Eine HTA ist die Dekomposition einer Aufgabe in seine Teilaufgaben. Eine HTA besteht aus der Aufgabe selbst, der Hierarchie ihrer Teilaufgaben und einem Plan, der die Reihenfolge der Abarbeitung beschreibt (siehe Abbildung 18). Aufgaben werden dabei so häufig zerlegt, bis sie entweder dem Benutzer oder dem System zugeordnet werden können. Jede Aufgabe kann unabhängig von ihrem Lösungsansatz einem Ziel zugeordnet werden. Ziele selbst können auch in hierarchischer Form aufgenommen werden.

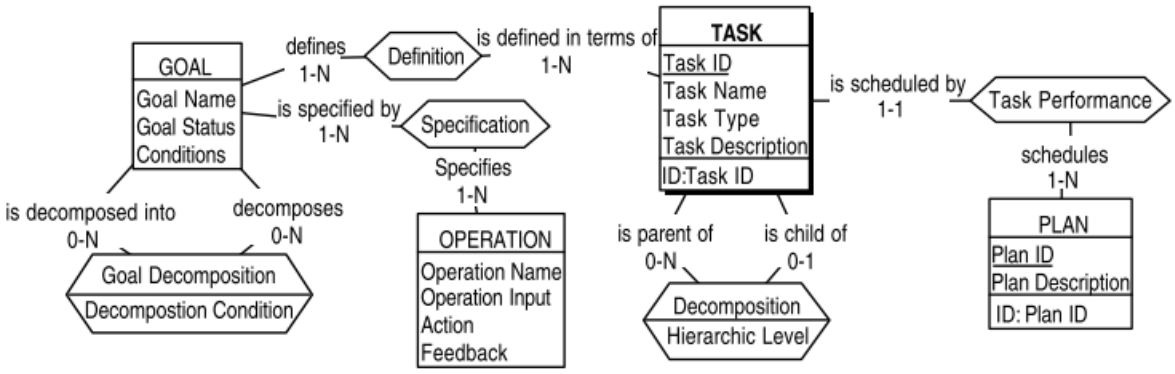


Abbildung 18: Datenmodell der Hierarchischen Aufgaben Analyse (Limbourg & Vanderdonckt, 2004)

Nachfolgend soll diese Struktur an einem Beispiel verdeutlicht werden (siehe Abbildung 19). Ziel dieser Tätigkeit ist es, ein Buch aus einer Bibliothek zu leihen. Dafür muss als erstes eine Bibliothek besucht werden, dann wird das Buch gesucht, um es dann aus dem entsprechenden Regal nehmen zu können. Für die Suche des Buches beschreibt die unterste Ebene weiter Teilaufgaben, die für die Tätigkeit 2 benötigt werden. Wie in Abbildung 19 zu sehen beschreibt die HTA neben der Zielhierarchie in jeder Stufe auch entsprechende Pläne. Sie dienen der Herstellung von Beziehungen der Unterziele einer Tätigkeit.

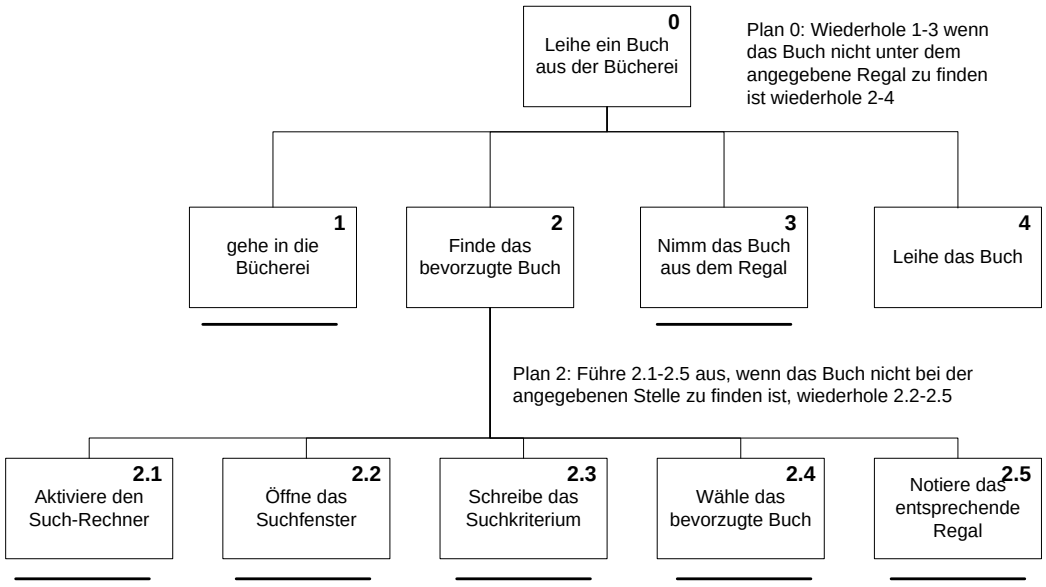


Abbildung 19: HTA-Beispiel für das Ausleihen eines Buches (Johansson, 2004)

Die HTA stellt ein sehr einfach nutzbares Verfahren zur Beschreibung von Handlungstheorien dar. Bei der HTA werden jedoch keine Objekte untersucht, die von der ausführenden Person verwendet oder manipuliert werden. Nach Specker kann dies für ein umfassendes Verständnis einer Aufgabe von Nachteil sein (Specker, 1998, S. 84).

Goals, Operators, Methods and Selection rules (GOMS)

Mit dem Aufgabenmodell GOMS (Goals Operators Methods Selectors) werden Aufgaben so häufig zerlegt, bis zeitlich messbare Einheiten entstehen. Entwickelt wurde es von Card, Moran und Newell (1983). Ähnlich wie bereits bei den Seven Stages of Action (Norman, 1998) oder dem 6-Ebenen-Modell (Herczeg, 2009) werden auf oberster Ebene zunächst die Ziele (Goals) und mögliche Teilziele definiert, die ein Benutzer erreichen möchte. Diese sollten in hierarchischer Form organisiert sein. Im Vergleich zu der oben beschriebenen HTA kann GOMS als kognitives Aufgabenmodell angesehen werden, in dem Aufgeben durch eine hierarchisch gegliederte, prozedurale Abfolge von Aktivitäten beschrieben werden. Durch Operatoren (Operators) werden elementare motorische oder kognitive Aktivitäten des Benutzers beschrieben. Sie sind unabdingbar für die Veränderung des Zustandes eines Handlungsraumes. Eine Methode (Method) beschreibt eine Abfolge von Aktivitäten, die sequentiell ausgeführt werden, um ein Ziel zu erreichen. Eine Aktivität kann beispielsweise das Drücken eines Knopfes oder einer Taste sein. Die Determinierung dieser Handlung hat die Erreichung eines Ziels zur Folge. Wenn mehrere Methoden zum gleichen Ziel führen, wird eine Auswahlregel (Selection Rule) verwendet, um die geeignetste wählen zu können (Limbourg & Vanderdonckt, 2004). Diese Form der Zerlegung einer Aufgabe beruht auf dem Verständnis der menschlichen Kognition. Durch die Beschreibung menschlichen Handelns in einer messbaren Sequenz von Verarbeitungsschritten kann eine messbare Aussage über die benötigte Zeit für die Bearbeitung der Aufgabe getroffen werden. Durch Zuordnung eines bestimmten Zeitaufwands für die jeweiligen Operatoren und Auswahlregeln, kann die Performanz jeder Methode überprüft und gegebenenfalls optimiert werden. Nach Herczeg (1994) eignet sich GOMS weniger für die konzeptionelle Entwicklung eines Systems. Vielmehr kann durch diese Methode das Verhaltens- und Problemlösungsmodell abgebildet werden, welches die Dynamik des Problemlösungsvorganges veranschaulicht. Somit ist GOMS im Vergleich zur HTA besser dafür geeignet, eine umfassendere und kognitiv spezifizierte Aussage über die Interaktion eines Mensch-Maschine-Systems zu treffen.

Méthode Analytique de Description de tâches (MAD*)

Ebenfalls auf der Dekomposition von Aufgaben aufbauend ist das MAD* Modell. MAD steht für Méthode Analytique de Description de tâches. Bei der MAD* werden zwei Arten von Aufgaben unterschieden: Die elementaren und die zusammengesetzten Aufgaben. Elementare Aufgaben können nicht weiter zerlegt werden und enthalten einen direkten Bezug zu einem oder mehreren Domänenobjekten. Zusammengesetzte Aufgaben bestehen aus verschiedenen Subaufgaben, die wiederum mit unterschiedlichen Operatoren verbunden sind. Die Operatoren lassen sich in vier unterschiedliche Kategorien unterteilen:

- Synchronisationsoperatoren, wie z.B. sequentiell, parallel oder simultan
- Ordnungsoperatoren, wie z.B. AND, OR oder XOR
- Zeitoperatoren, wie z.B. Anfang, Ende oder Dauer und
- Hilfsoperatoren, wie z.B. elementar oder unbekannt

Eine Aufgabe in MAD* besteht aus verschiedenen Attributen. Einem Namen, einem Ziel, einer Kennung, einer Priorität gegenüber anderen Aufgaben, einem optionalen Charakter, einer Spezifikation, ob die Aufgabe beispielsweise unterbrochen werden kann, einem Typ (z.B. sensomotorischen oder kognitiven), einer Modalität (also manuell, automatisch oder interaktiv) und der Wichtigkeit einer Aufgabe. Ein Anfangszustand gibt einen Zustand der Welt vor der Ausführung der Aufgabe an. Der Endzustand beschreibt den Zustand nach der Ausführung der Aufgabe. (Limbourg & Vanderdonckt, 2004)

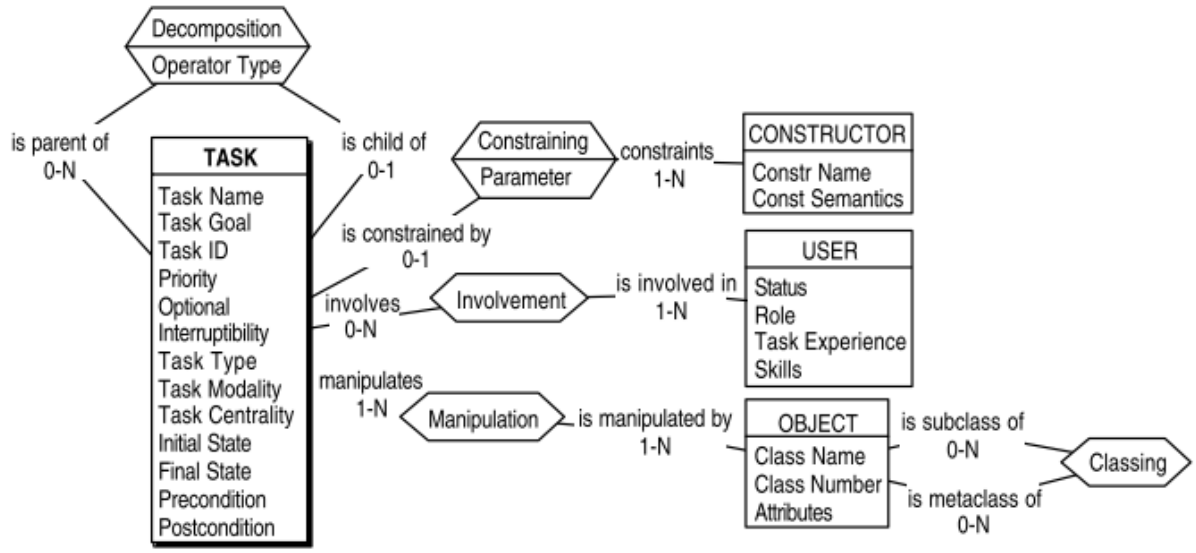


Abbildung 20: Datenmodell von MAD* (Limbourg & Vanderdonckt, 2004)

Groupware Task Analysis (GTA)

Nach Gerrit van der Veer kann eine Aufgabe in (Benutzer-)Aktionen und (System-)Ereignisse unterschieden werden. Wichtig dabei ist der Bezug zur abgeleiteten Aufgabe (Veer & van Welie, 1999). Das GTA-Modell wird um einige Entitäten erweitert. Neben den hierarchisch zerlegbaren Aufgaben sieht das Aufgabenmodell noch Ereignisse, Objekte, Vertreter und deren Rollen vor, wie aus Abbildung 21 hervorgeht. Ein Ereignis ist der Auslöser (Trigger) für eine Aufgabe. Aufgaben können in Teilaufgaben zerlegt und von einem Vertreter (Agent) ausgeführt werden. Dieser kann auch durch seine Rolle repräsentiert werden. Bei der Ausführung einer Aufgabe werden Objekte verwendet oder auch manipuliert.

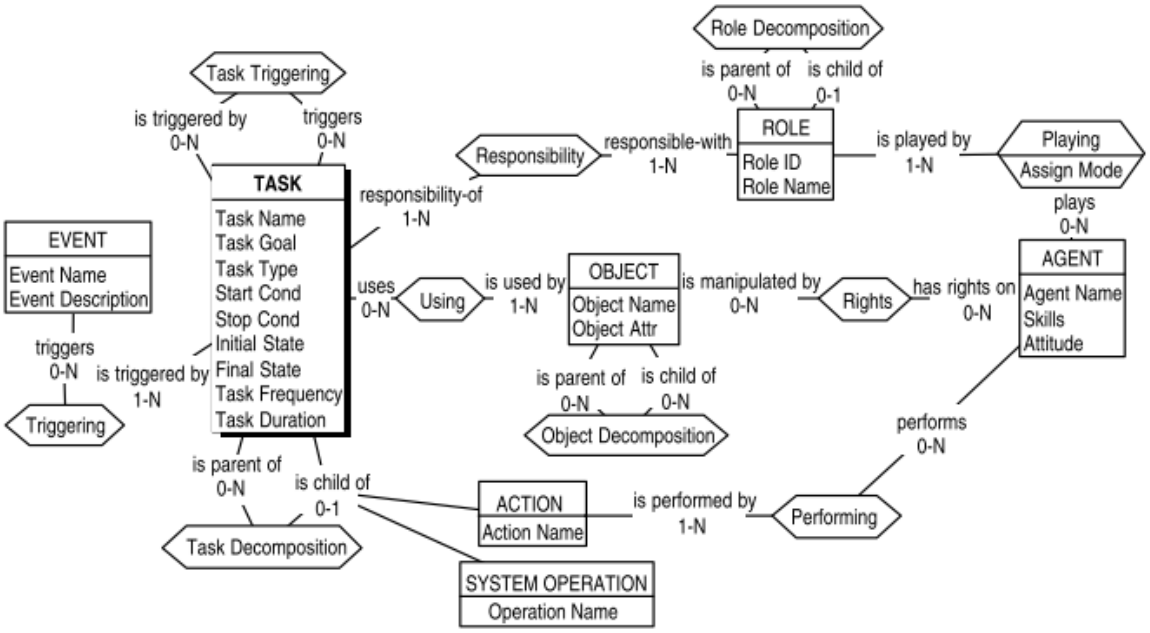


Abbildung 21: Datenmodell der Groupware Task Analysis (Limbourg & Vanderdonckt, 2004)

Method for Usability Engineering (MUSE)

Speziell für die Belange bei der Entwicklung interaktiver Software wurde MUSE (Method for Usability Engineering) entwickelt. Dieses Modell sieht eine umfassendere Einbettung der Aufgaben in ihren Kontext vor (siehe Abbildung 22). Auf der obersten Ebene dieses Modells steht die Organisation. Diese lässt sich in verschiedene Stellen (Jobs) unterteilen. Jede Stelle besteht aus drei verschiedenen Komponenten. Einer Hierarchie aus Zielen, einer Liste verschiedener Rollen, die der Stelle zugewiesen werden können und einer Liste von Funktionen, die zur Ausführung der Stelle benötigt werden. Jede aufgenommene Rolle und Funktion wird einer Aufgabe zugewiesen. Aufgaben können so häufig zerlegt werden, bis sie einer Aktion oder einem Objekt zugeordnet werden können. Jede Aufgabe erhält ein Terminierungskriterium.

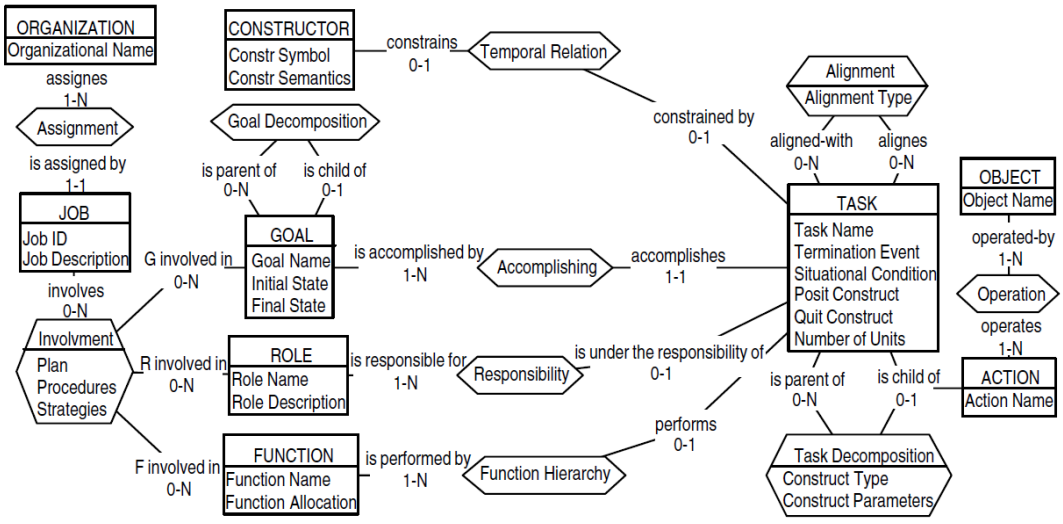


Abbildung 22: Datenmodell von MUSE (Limbourg & Vanderdonckt, 2004)

ConcurTaskTree (CTT)

Als letztes Aufgabenmodell soll hier die ConcurTaskTrees von Paternò vorgestellt werden (Paternó, 2004). Bei diesem Modell handelt es sich auch um eine Hierarchische Dekomposition von Aufgaben. Zusätzlich zu dem HTA-Modell werden hierbei Objekte und Rollen mit berücksichtigt. Über den Konstruktor (siehe Abbildung 23)

können sogenannte Operatoren (wie bei MAD*) die Ausführungsreihenfolge von benachbarten Aufgaben anzeigen. In den bisherigen Modellen steckte diese Information immer im Vaterknoten eines Elementes. Für die jeweiligen Operatoren wurde bei dem CTT-Modell auch eine bestimmte Notation festgelegt. So beschreibt beispielsweise der Operator „Enabling“ mit der Syntax „>>“, dass eine Aufgabe vor einer anderen ausgeführt werden muss, oder der Operator „Concurrent Tasks“ mit dem Zeichen „|||“, dass zwei Aufgaben in beliebiger Reihenfolge ausgeführt werden können. Ähnlich wie bei der HTA werden bei den ConqurTaskTrees Aufgaben so häufig zerlegt, bis sie dem Benutzer oder dem System zugeordnet werden können. Diese sogenannten Basisaufgaben sind dann entweder Handlungen (Actions) oder Objekte. Es existiert für dieses Aufgabenmodell auch eine Softwarelösung, das sogenannte CTTE (ConqurTaskTree Enviroment) welches in Kapitel 2.4.4 vorgestellt wird.

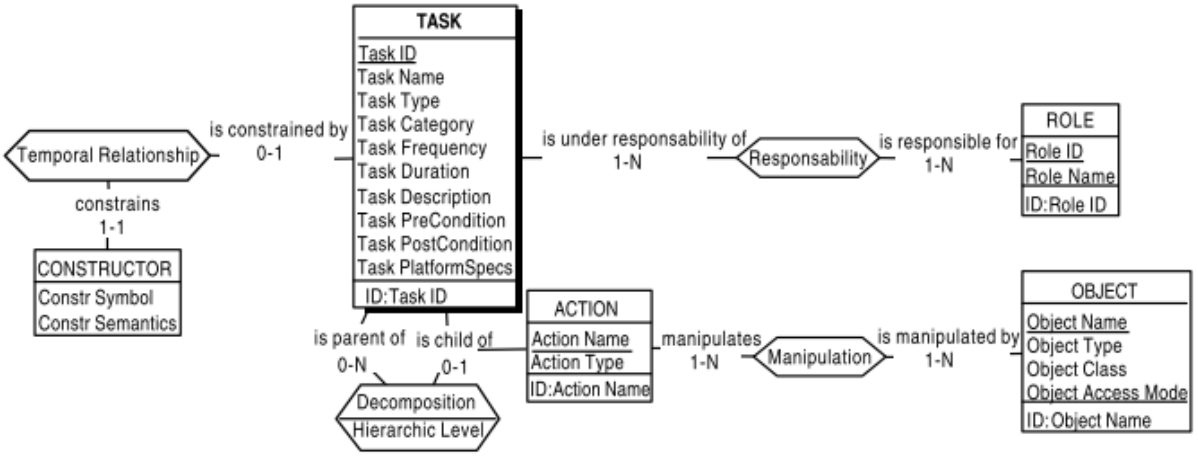


Abbildung 23: Datenmodell von ConqurTaskTree (Limbourg & Vanderdonckt, 2004)

Constantine und Lockwood sehen eine hierarchische Aufgabenanalyse als konzeptuell und kategorisierend führend, wenn sie gut durchdacht ist. Sie eignen sich für saubere Klassifikationen und Ausbalancierungen von Aufgaben. Somit sei es nicht verwunderlich, dass sie überwiegend im akademischen Umfeld Verwendung finden. (Constantine & Lockwood, 1999, S. 100)

2.3.1.2.2 Prozessorientierte Aufgabenmodelle

Für die Beschreibung logischer Abfolgen von Aufgaben, mit allen Schritten und notwendigen Entscheidungspunkten, wurden sowohl in der Welt der HCI als auch in der Softwareentwicklung eine ganze Reihe verschiedener Prozessmodelle entwickelt. Nach Hackos und Redish ist es an manchen Punkten in der Entwicklung notwendig sich ein genaues Bild über die Arbeitsweise eines Benutzers zu verschaffen. Eine Prozessanalyse ermöglicht einen Einblick in die physikalischen Schritte eines Benutzers und auch in ihre mentalen Entscheidungen, die sie treffen müssen (Hackos & Redish, 1998, S. 77).

> *„Ein Prozess ist eine Abfolge von Teilprozessen, Aktivitäten und Aktionen, die über explizite Kanten/Transaktionen, mit eindeutig definierten Start- und Endpunkten, in dem Akteure, unter Einsatz von Ressourcen, klar definierte Ergebnisse erzeugen.“ [sic]* (Schmid, 2009, S. 51)

Formal gesehen sollte ein Prozess stets mindestens ein Ergebnis liefern. Lediglich im Fall eines Fehlers kann ein Ergebnis ausbleiben. Eine etwas knappere Definition besagt, dass ein Prozess eine Aneinanderreihung von Aktivitäten ist, die Eingaben in Ausgaben bzw. Ergebnisse verwandelt (ISO/TS 18152:2010). Aus organisatorischer Sicht kann sich ein Prozess auch über mehrere Stellen erstrecken und jede Stelle erarbeitet nur einen Teil eines Gesamtergebnisses.

Aufgabensequenz

Die einfachste Form eines Prozessmodells ist die textuelle Dokumentation der einzelnen Arbeitsschritte als Aufgabensequenz. Eine Aufgabensequenz ist eine Liste von Aufgaben, die von einem Benutzer in einer bestimmten Reihenfolge abgearbeitet werden sollte (Hackos & Redish, 1998, S. 71). Diese Art von Aufgabenmodell hat Ähnlichkeiten mit den in Kapitel 2.3.3.1 vorgestellten Szenarien.

Die nach Kirwan erste grafische Visualisierung einer Aufgabe als Prozessdiagramm stammt aus dem Jahre 1921 von Gilbreth und Gilbreth (Kirwan & Ainsworth, 1992, S. 84). Aus diesem haben sich mit der Zeit eine Reihe unterschiedlicher Notationen ergeben, das Prinzip blieb jedoch das gleiche. Ein solches Prozessdiagramm hat immer einen Start- und einen Endknoten. Zwischen diesen befinden sich die sequentiell zu durchlaufenden Teilaktivitäten. An Punkten, bei denen alternative Schritte gewählt werden können, gibt es Entscheidungsknoten.

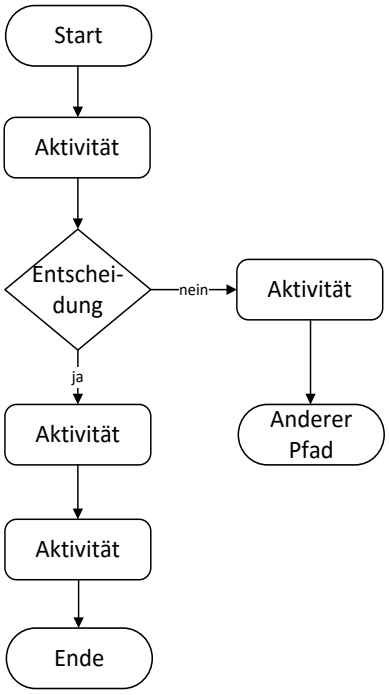


Abbildung 24: Einfaches Prozessdiagramm

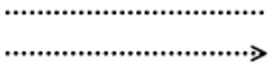
Business Process Model and Notation (BPMN)

Ein sich in den letzten Jahren rasant ausbreitendes Prozessmodell ist die „Business Process Model and Notation" Standard (BPMN). Grund für den Durchbruch, sei unter anderem die Tatsache, dass BPMN die Fachwelt und die IT-Welt miteinander vereinen könne. Grund hierfür ist die Kombination der grafischen Modellierung für die wirtschaftsnahen Disziplinen und der formalen Prozessgrammatik für die IT-Welt. (Göpfert & Lindenbach, 2013)

Andre Grass (2009) zeigt Möglichkeiten auf, wie BPMN-Modelle mit den in der Informatik etablierten UML-Modellen kombiniert werden können, um beispielsweise Produktstrukturen direkt an die BPMN-Geschäftsprozesse zu koppeln. BPMN unterstützt neben der Erstellung von *Prozessdiagrammen* auch drei weitere Diagrammtypen. Das *Kollaborationsdiagramm* stellt die Zusammenarbeit zwischen verschiedenen Partnern dar. Mit dem *Choreographiediagramm* bleibt der Prozess einzelner Partner geheim, indem eine Aktivitätsfolge zu einem Knoten zusammengefasst wird. Das *Konversationsdiagramm* kann als vereinfachte Ansicht für komplexe Konversationen zwischen einzelnen Partnern genutzt werden (Object Management Group, 2011a).

Tabelle 2: Grafische Basiselemente von BPMN (Object Management Group, 2011a)

Element	Beschreibung	Notation
Ereignis	Ereignisse können in einen Prozess eintreten. Meist werden sie durch einen Auslöser hervorgerufen oder bewirken ein Ergebnis.	

Aktivität	Eine Aktivität ist meist ein Teilschritt in einem Prozess. Sie kann jedoch auch einen gesamten Teilprozess beinhalten.	
Gateway	Ein Gateway ist ein Entscheidungsknoten, an dem zwischen verschiedenen Alternativen Prozessverläufen gewählt werden kann.	
Sequenzfluss	Ein Sequenzfluss gibt die Reihenfolge der Aktivitäten an	
Nachrichtenfluss	Ein Nachrichtenfluss zwischen zwei Parteien deutet auf eine Kommunikation hin. Bei BPMN erfolgt dies über unterschiedliche Pools hinweg.	
Verbindung	Mit Verbindungen werden Informationen und Artefakte verknüpft. Der Pfeil gibt die Richtung des Informationsflusses an.	
Pool	Ein Pool ist die grafische Repräsentation eines Akteurs.	
Lane	Eine Lane kann dazu verwendet werden, mehrere Akteure beispielsweise zu einer Organisationseinheit zusammenzufassen.	
Datenobjekt	Datenobjekte beinhalten die für eine Aktivität benötigten Informationen oder deren Ergebnisse.	
Nachrichten	In Nachrichten werden die Inhalte einer Kommunikation abgebildet.	
Gruppe	Eine Gruppe fasst mehrere grafische Elemente zusammen. Sie kann zu Kategorisierungszwecken verwendet werden.	
Bemerkungen	Eine Bemerkung kann an jedes beliebige Element gehängt werden.	

Erweitert werden die Basiselemente durch eine Reihe verschiedener Sonderereignisse und die Möglichkeit Teilprozesse in einer Aktivität zu schachteln.

2.3.1.3 Organisationsanalyse

Nachdem die Aufgaben und deren Abläufe bzw. Prozesse analysiert wurden sollten diese betrieblichen Strukturen zugeordnet werden. In der Betriebswirtschaft wird der letzte Schritt dieses Vorgangs auch als *Aufgabensynthese* bezeichnet (siehe Abbildung 26). Aufgaben bzw. deren Teilaufgaben werden häufig von unterschiedlichen Personen oder Gruppen erledigt. Eine Gruppe kann aus organisatorischer Sicht auch als Abteilung bezeichnet werden. Diese besitzt meist einen Vorgesetzten, der wiederum einer höheren Abteilungsebene zugehörig sein kann. Diese meist hierarchische Struktur einer Organisation kann verschiedene Formen einnehmen. Eine *funktionale Organisation* (siehe Abbildung 25 oben) wird unterteilt in der zweiten Ebene nach dem Verrichtungsprinzip, beispielsweise nach Beschaffung, Entwicklung und Vertrieb (Laux & Liermann, 2005).

Bei der *Spartenorganisation* (siehe Abbildung 25 unten) wird auf der zweiten Ebene vor allem nach Produkten oder Produktgruppen unterteilt. Alternativ könnte jedoch auch nach Absatzbereich oder Kundengruppe unterschieden werden. Innerhalb einer Sparte wird daraufhin erneut verrichtungsorientiert unterteilt. Synonym zu Spartenorganisation werden auch die Begriffe „divisionale Organisation“ oder „Geschäftsbereichsorganisation“ verwendet. (Kistner & Steven, 2002)

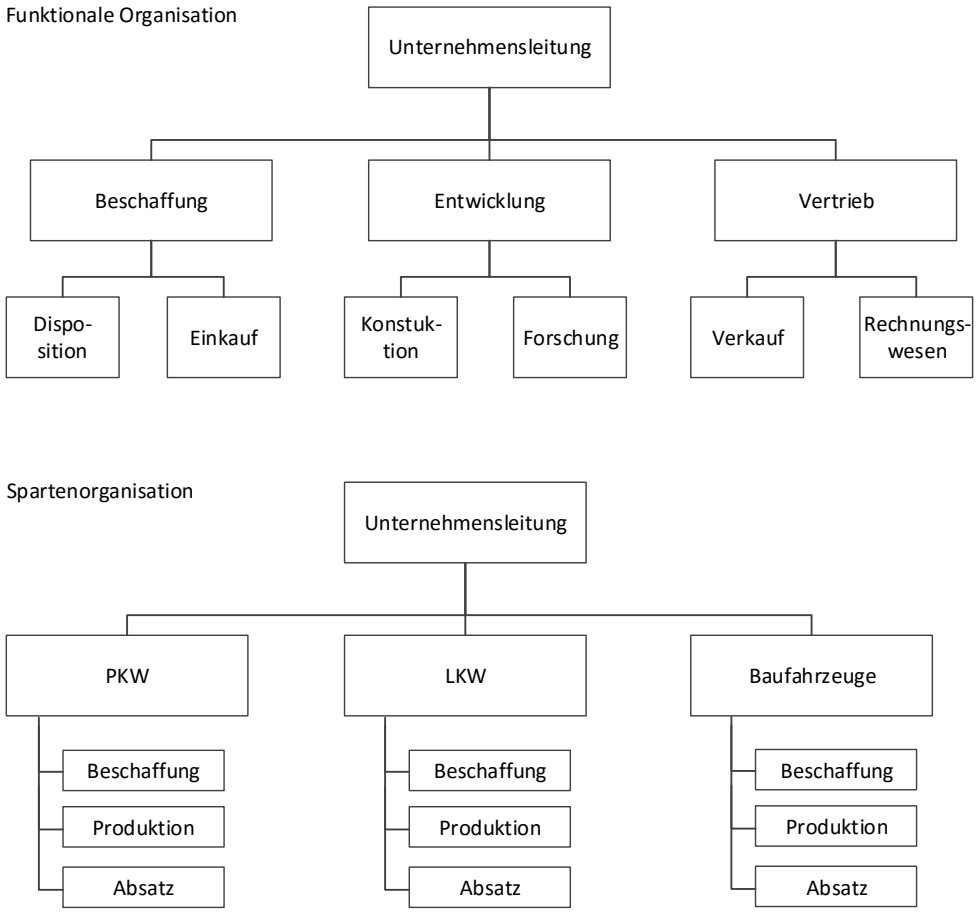


Abbildung 25: Organisationsaufteilung nach Funktion oder Sparte

Wie aus Abbildung 26 hervorgeht, lässt sich eine Organisationsanalyse in zwei Bereiche unterteilen. Sie setzt sich aus der *Aufbau- und der Ablauforganisation* zusammen. Ausgehend von den Aufgaben mit ihren jeweiligen Teilaufgaben wird in der Aufbauorganisation eine Aufgabenanalyse durchgeführt. Anschließend werden die Teilaufgaben den jeweiligen Stellen einer Organisation zugeordnet (Bergmann & Garrecht, 2008). Teilaufgaben, die

durch mehrere Stellen erledigt werden, können zu Abteilungen zusammengefasst werden. Eine *Stelle* kann wie folgt definiert werden:

> *"Die Stelle ist die kleinste organisatorisch zu definierende Organisationseinheit. Sie entsteht durch Zuordnung von (Teil-) Aufgaben und gegebenenfalls von Sachmitteln auf einen einzelnen menschlichen Aufgabenträger."* (Bühner, 2004, S. 61)

Eine Stelle wird durch eine Person besetzt. Dabei sollte bei der Besetzung darauf geachtet werden, dass die Person mit ihren Fähigkeiten die für diese Stelle vorgesehenen Aufgaben erfüllen kann.

Da in der Software-Entwicklung auch häufig der Begriff der *Rolle* verwendet wird (beispielsweise Balzert, 2009, S. 138; Wirdemann, 2011), soll dieser im Zusammenhang mit der Aufbauorganisation verortet werden. Den Definitionen einer Rolle aus Kapitel 2.3.1.3 und 2.3.1.1 zufolge stellt eine Rolle eine abstrakte Qualifikation eines Akteurs bzw. einer Person dar. Tätigkeitstheoretisch werden die Aktivitäten einer Person aus Sicht der Rolle ausgeübt. Dabei kann eine Person in ihrem Berufsalltag mehrere Rollen einnehmen. Beispielsweise könnte die Person mit der Stelle eines „Wissenschaftlichen Mitarbeiters“ zwischen den Rollen des „Dozenten“, des „Betreuers (studentischer Abschlussarbeiten)“ oder der des „Doktoranden“ wechseln.

Für das hier beschriebene Modell der hierarchischen Dekomposition einer Organisation beschreibt eine Stelle zwar die kleinste organisatorische Einheit, kann jedoch zum Zweck der abstrakten Repräsentation der Beziehung zwischen dem Benutzer und dem System noch in die jeweiligen Rollen unterteilt werden. Eine Rolle kann also Aufgaben innerhalb einer Stellenbeschreibung noch weiter kategorisieren.

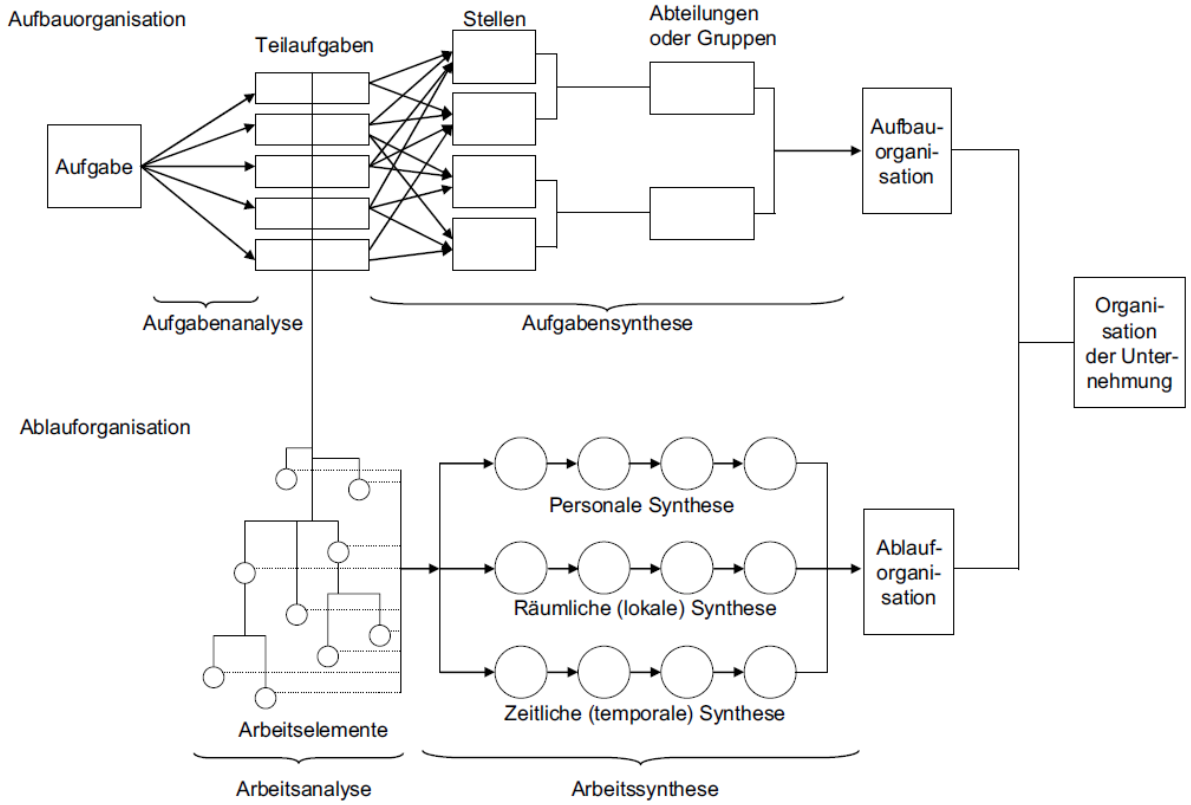


Abbildung 26: Modell Organisatorischer Gestaltung (Bleicher, 1991, S. 49)

Durch die Zuordnung von *Arbeitselementen* zu den jeweiligen Teilaufgaben einer Stelle, die wiederum in einer zeitlichen Reihenfolge und einem Ort zugewiesen werden können, wird die *Ablauforganisation* beschrieben (Bergmann & Garrecht, 2008). Als Synonym für das Arbeitselement soll hier auch der Begriff des Artefaktes verwendet werden. Wie bereits in Kapitel 2.2.1 vorgestellt handelt es sich hierbei um Gegenstände, die von Menschen während der Arbeit erstellt oder modifiziert werden. Sie spielen nach Specker häufig eine wichtigere Rolle als die Werkzeuge, mit denen sie manipuliert werden können.

> *„Aus Sicht des Benutzers liegt das Hauptinteresse zunächst nicht am Werkzeug als solchem, sondern an den zu manipulierenden Objekten des Gegenstandsbereiches. Softwarewerkzeuge haben also den Nachteil, dass die Tätigkeit häufig nicht am ‚eigentlichen Objekt' erfolgt, wie dies das ‚Postulat der Gegenständlichkeit' fordert, sondern nur vermittels eines Werkzeuges."* (Specker, 1998, S. 94)

2.3.1.4 Nutzungskontext

Der Nutzungskontext setzt sich nach der ISO 9241-11 aus den Benutzern, ihren Aufgaben, den verwendeten Arbeitsmitteln und der Umgebung zusammen (siehe Kapitel 2). Erst das Zusammenspiel aller Einzel Informationen ermöglicht ein umfassendes Verständnis des Nutzungskontextes. Um die einzelnen Analyseergebnisse zu den Benutzern, deren Aufgaben, der Arbeitsmittel und der Umgebung zusammenzuführen und festzuhalten, können Szenarien verwendet werden. Diese häufig narrativen Geschichten eignen sich, um neben der Beschreibung des Problems weitere wichtige Kontextinformationen zu berücksichtigen und einen Realitätsbezug herzustellen. In Abbildung 27 sind die einzelnen Entitäten grafisch zusammengefasst.

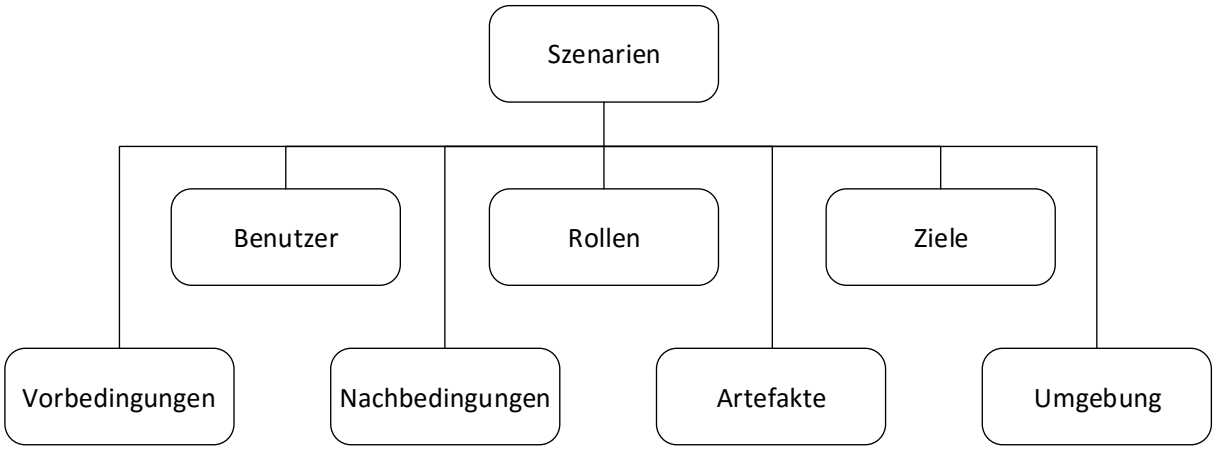


Abbildung 27: Kontextinformationen in Szenarien (angelehnt an Pohl, 2008, S. 125; Rosson & Carroll, 2002)

Szenarien, die der Definition des Ist-Zustandes eines Nutzungskontextes dienen, werden auch als Indikative Szenarien bezeichnet (Pohl, 2008). Häufig dienen Szenarien auch der Identifikation von bisher unberücksichtigten Stakholdern. Szenarien können darüber hinaus auch weitere *charakteristische Elemente* beschreiben (Rosson & Carroll, 2002), wie beispielsweise Details über die Situation, welche die Ziele, Aktivitäten und Reaktionen der Benutzer erklären oder motivieren:

- Persönlicher Attribute, die ein gewisses Verhalten erklären
- Motivation der Benutzer die Arbeit durchzuführen
- Geistige Tätigkeit, welche die Ziele in Verhalten umsetzen
- Geistige Tätigkeit, welche Aspekte der Situation analysiert
- Externe Aktionen oder Reaktionen des Computers oder anderer Artefakte
- Auditorisches oder taktiles Feedback von Eingabegeräten wie Maus und Tastatur

2.3.2 Spezifikation

Die Spezifikation könnte theoretisch ebenfalls der Analysephase zugeordnet werden. Neben den funktionalen und nichtfunktionalen Anforderungen, sollten hier auch die Anforderungen im *„[…] Zusammenhang mit der Beschreibung des vorgesehenen Nutzungskontext und der wirtschaftlichen Ziele des Systems […]"* (ISO 9241-210:2011) erfasst werden, den sogenannten Nutzungsanforderungen.

> *„Die Anforderungsspezifikation stellt somit die Problemdefinition dar, für die die Systemarchitektur die zugehörige Lösung beschreibt."* (Pohl, 2008, S. 20)

Ausgehend vom Nutzungskontext sollten die Bedürfnisse der Benutzer und weiterer Stakeholder identifiziert werden. Es sollte dabei dokumentiert werden, „was“ die Benutzer erreichen wollen und nicht „wie“. Es gilt die Intentionen der Benutzer bzw. Stakeholder zu erfassen. Diese sollten von der zu entwickelnden Systemlösung klar getrennt werden können. Pohl (2008) verwendete aus diesem Grund die Bezeichnung des *Ziels* für diese Form der Anforderungen. Für die Konkretisierung des mit den Zielen zusammenhängenden Nutzungskontextes schlägt Pohl die Verwendung von Szenarien vor (siehe Abbildung 28). Mit ihnen kann ein klareres Verständnis eines Ziels entwickelt werden. Eine ausführliche Beschreibung des von Pohl beschrieben Vorgehens, die Ziele und Szenarien in der Softwareentwicklung zu verwenden, wird in Kapitel 3.3.1 weiter verdeutlicht.

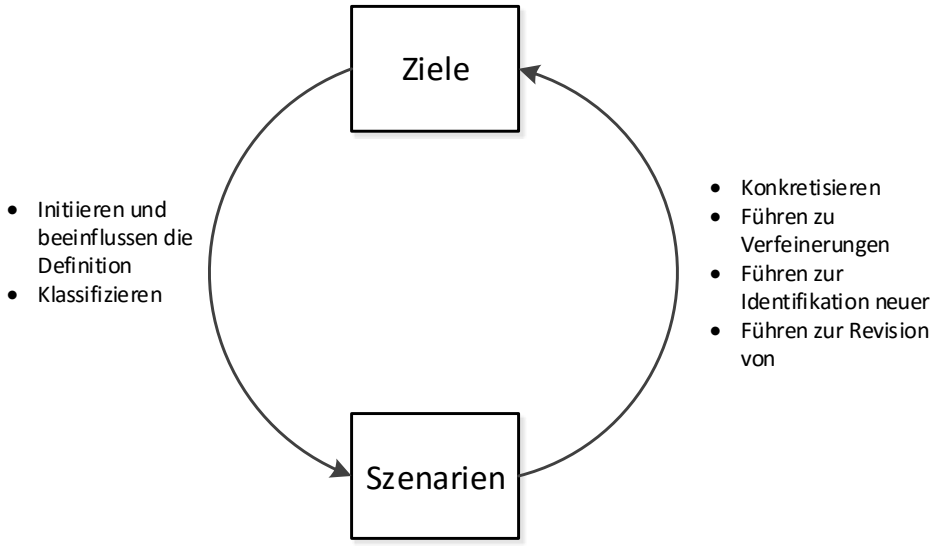


Abbildung 28: Wechselwirkung zwischen Zielen und Szenarien (Pohl, 2008)

Die Konkretisierung von Nutzungsanforderungen bzw. die Entwicklung von Lösungsansätzen wird im folgenden Kapitel behandelt.

2.3.3 Entwicklung

Bei der Methode der Entwicklung werden im UE häufig Prototypen verwendet. Prototypen dienen als abstrakte Beschreibungsform des zu entwickelnden Systems. Sie werden eingesetzt, um in frühen Entwicklungsstadien Lösungsansätze bewerten bzw. formativ evaluieren (siehe Kapitel 2.3.4) zu können. Darüber hinaus können Prototypen dazu verwendet werden, die Kommunikation zwischen Benutzern und Designern bzw. Entwicklern zu verbessern. Beide Seiten entwickeln während des Gestaltungsprozesses einer Software unterschiedliche mentale Modelle des Anwendungssystems (Abbildung 29). Der Benutzer, der auch als Experte der Anwendungswelt angesehen werden kann, entwickelt eine ganz bestimmte Vorstellung von der Funktionsweise des Systems, die nicht zwangsläufig mit der tatsächlichen Funktionsweise übereinstimmt. Der Entwickler hingegen hat in der Regel ein strukturierteres, umsetzungsnahes Modell des Systems, da er den tatsächlichen technischen Aufbau kennt bzw. realisiert. Dies wird als „konzeptuelles Modell“ (Herczeg, 2009, S. 51) bezeichnet. Aus den typischerweise stark abweichenden Vorstellungen von dem System ergeben sich in der Kommunikation erhebliche Schwierigkeiten, da Benutzer und Entwickler keine gemeinsame Basis haben und es somit zu Missverständnissen und Unverständnis auf beiden Seiten kommen kann.

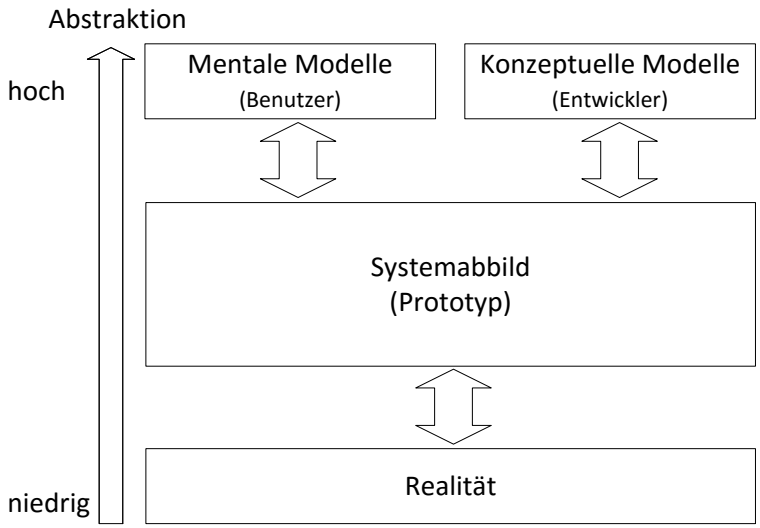


Abbildung 29: Szenarien als Kommunikationsmittel zwischen der Realität und der abstrakten Welt

Der Entwickler hat in der Folge die komplexe Aufgabe die Anwendungswelt bzw. Nutzungskontext zu erfassen, einen geeigneten Lösungsansatz zu entwickeln und diesen dann dem Benutzer nachvollziehbar zu vermitteln. Zu diesem Zweck eignen sich die von beiden Seiten durch ihre Anschaulichkeit einfach zu verstehenden Prototypen.

Prototypen werden im UE in unterschiedlicher Form eingesetzt. Sie dienen in erster Linie dazu ein System oder Teile davon zu repräsentieren. Diese Entwürfe können von einer einfachen Handskizze bis hin zu funktionsfähigen Teilkomponenten eines Systems reichen. Ziel ist es, mittels eines Prototyps dem Benutzer möglichst früh und kostengünstig erste Lösungsansätze eines Systems zeigen zu können. Zur Definition eines *Prototyps* kann folgende Beschreibung herangezogen werden:

> *"A preliminary type, form, or instance of a system that serves as a model for later stages or for the final, complete version of the system." (IEEE Std 610.12:1990, S. 60)*

Der Erstellungsprozess wird als *Prototyping* bezeichnet. Die Kernidee des Prototyping besteht darin, Zeit und Kosten bei der Erstellung von Lösungsansätzen, die von realen Benutzern getestet werden können, einzusparen. Diese Einsparungen können nur durch Reduzierung des Umfangs des Gesamtsystems erreicht werden (Nielsen, 1993). Dies kann in zwei verschiedenen Dimensionen erreicht werden (siehe Abbildung 30).

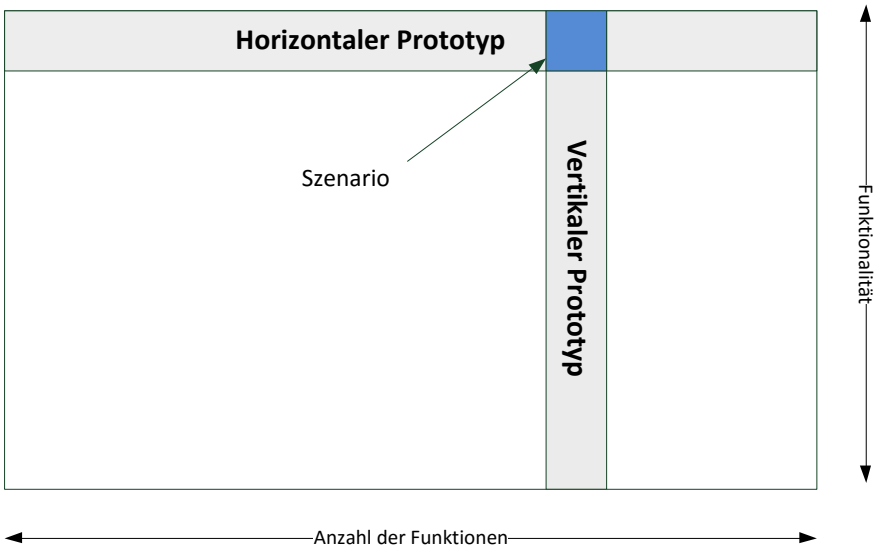


Abbildung 30: Die zwei Dimensionen von Prototypen (Nielsen, 1993, S. 94)

Durch die Reduzierung des Gesamtsystems auf einen Teil der Funktionen entsteht ein *vertikaler Prototyp*. Die Funktionen werden jedoch in die gesamte Tiefe durch alle Schichten eines Systems abgebildet. Das heißt, die gesamte Funktionalität einer Funktion wird implementiert und kann anschließend überprüft werden. Diese Art von Prototyp eignet sich vor allem zur Überprüfung der Machbarkeit bestimmter Funktionen (Grechenig, Bernhart, Breiteneder & Kappel, 2010). In manchen Fällen kann die Funktionalität anschließend auch weiterverwendet werden.

Horizontale Prototypen verdeutlichen die gesamte Benutzungsschnittstelle eines Systems. Sie bilden die gesamte Anwendung auf oberster Ebene ab, die für den Benutzer wahrnehmbar ist. Die Einsparung geschieht bei der dahinterliegenden Funktionalität. Dem Benutzer können die einzelnen Funktionen, ohne die dahinterliegende Funktionalität veranschaulicht werden. Mit horizontalen Prototypen kann überprüft werden, wie die einzelnen Funktionen zusammenhängen und ob alle vorhanden sind. Für die Erstellung horizontaler Prototypen existieren verschiedene Designwerkzeuge und Frameworks, welche die Erstellung von grafischen Benutzungsschnittstellen unterstützen können. Szenarien (siehe Kapitel 2.3.3.1) bilden einen Ausschnitt aus einem horizontalen und einem vertikalen Prototypen ab, indem die Funktionalität einiger Funktionen als Interaktionsfolgen beschrieben wird.

Je nach Verwendungszweck und Zielsetzung können Prototypen auch auf eine andere Weise unterschieden werden (Floyd, 1984). Als *explorative Prototypen* werden meist horizontale Prototypen bezeichnet, die dazu verwendet werden, einem Benutzer sehr komplexe Sachzusammenhänge bzw. Anforderungen zu veranschaulichen. Sie kommen zum Einsatz, wenn Problemstellungen noch nicht geklärt sind und eine gemeinsame Diskussionsgrundlage benötigt wird. Speziell in frühen Phasen eines Entwicklungsprozesses kann diese Form dazu dienen, neue Anforderungen aufzudecken und die konzeptionelle Machbarkeit eines Lösungsansatzes zu überprüfen. Häufig kommen dabei horizontale Prototypen zum Einsatz, um möglichst viele Eigenschaften (Funktionen) eines Systems veranschaulichen zu können (Grechenig et al., 2010).

Vertikale Prototypen, die der Überprüfung von Machbarkeiten dienen, können auch als *experimentelle Prototypen* bezeichnet werden. Nach der Evaluation werden sie meistens nicht weiterverwendet, wodurch sie manchmal auch als „Throwaway-Prototypen“ bezeichnet werden. *Evolutionäre Prototypen* sind eine Mischform aus horizontalen und vertikalen Prototypen. Sie dienen im Gegensatz zu den experimentellen Prototypen als erster Meilenstein eines Systems, welches kontinuierlich weiterentwickelt wird. Nach jeder Iteration werden die Ergebnisse einer Evaluation dazu verwendet, sie in die nächste Iteration einzubauen.

Eine weitere Unterscheidung des Prototyping kann in der Genauigkeit der Darstellung vorgenommen werden. Hier werden High- und Low-Fidelity Prototypen unterschieden. Mit *Low-Fidelity* werden Prototypen bezeichnet, deren Detaillierungsgrad nur gering ausgeprägt ist. Sie dienen der Überprüfung von Konzepten, Design-Alternativen und der generellen Bildschirmaufteilung (Rudd, Stern & Isensee, 1996). Low-Fidelity-Prototypen können mit einfachsten Werkzeugen wie Papier und Bleistift, jedoch auch mit speziellen Werkzeugen wie Balsamiq hergestellt werden. Durch den geringen Zeit- und Kostenaufwand bei der Erstellung eignen sie sich gut für die Evaluierung unterschiedlicher Designkonzepte, die anschließend leichter verworfen werden können. Da Low-Fidelity-Prototypen aus einer Serie von verschiedenen Fenstern und Menüs bestehen (Rudd et al., 1996), kann die spätere Funktionalität des Systems nur bedingt simuliert werden. Für den Einbezug realer Benutzer sollten jedoch möglichst komplette Arbeitsprozesse durch den Prototypen umgesetzt werden können (Dahm, 2006).

High-Fidelity-Prototypen sind im Gegensatz dazu komplett interaktiv. Benutzer können Daten in Felder eingeben, erhalten Antworten, können Icons selektieren und Fenster öffnen. Ein High-Fidelity-Prototyp fühlt sich für einen Benutzer wie eine reale Anwendung an. Zudem können sie auch die technischen Möglichkeiten besser vermitteln (Walker, Takayama & Landay, 2002). Auf Grund des höheren Aufwandes bei der Erstellung, werden Konzepte nur ungern von Designern geändert.

2.3.3.1 Szenarien

Szenarien können als textuelle Form der Systembeschreibung auch als Prototyp angesehen werden. Nach Nielsen sind sie die kostengünstigste Art eines Prototypen (Nielsen, 1993, S. 18). Sie beschreiben die Funktionalität einer gekapselten Sammlung von Funktionen eines Systems. Durch die Beschreibung eines konkreten Ablaufs wird in der Vertikalen (siehe Abbildung 30) die Funktionalität einer Funktion beschrieben.

Der Prozess des Scenario-based-Designs von Rosson und Carroll aus Kapitel 2.2.4 hatte bereits aufgezeigt, wie Szenarien sowohl zur Beschreibung des Ist-Zustandes bzw. des zu lösenden Problems über die sukzessive Transformation mittels Claims hin zu immer detaillierteren Design-Szenarien geführt werden können. In der dort beschriebenen Design-Phase werden drei verschiedene Szenario-Typen verwendet. Im ersten Schritt wird in *Aktivitäts-Szenarien* ausschließlich auf die Tätigkeiten eingegangen, die ein Benutzer zukünftig mit dem System erledigen sollte, ohne dabei zu konkretisieren, wie das System dabei aussieht. Erst im nächsten Schritt, nachdem alle zu unterstützenden Aktivitäten geklärt wurden, werden die Aktivitäts-Szenarien weiterentwickelt. Hier wird geklärt, welche Informationen von dem System zur Verfügung gestellt werden müssen, um Tätigkeiten ausführen zu können. Durch die Ergänzung dieser Information werden die Aktivitäts-Szenarien zu *Informations-Szenarien*. Dieser Prozessschritt kann mehrere Iterationen durchlaufen. Erst im letzten Schritt wird geklärt, wie das System den Benutzer bei der Interaktion mit den Informationen unterstützen soll. Durch diese letzte Transformationsstufe werden aus den Informations-Szenarien dann die *Interaktions-Szenarien*.

Neben diesem durchgehenden Prozess von Rosson und Carroll existieren weiterer Varianten von Szenarien. Je nach Verwendungszweck können diese variiert werden. Von Rolland stammt die nachfolgende Klassifizierung der unterschiedlichen Ausprägungen von Szenarien (Rolland et al., 1998). Sie unterscheidet Szenarien in folgenden vier Kategorien:

- Form
- Inhalt
- Zweck
- Lebensdauer

Die am häufigsten anzutreffende *Form* eines Szenarios ist die einer narrativen Erzählung. Bei diesen handelt es sich um Geschichten, die den Alltag eines Protagonisten mit möglichst relevanten Informationen beschreiben. Einige Autoren (Pohl, 2008; Richter & Flückiger, 2010) schlagen speziell für die Ermittlung von entwicklungsrelevanten Anforderungen eine strukturiertere Form der Szenarien vor. In der einfachsten Ausprägung bestehen diese aus einer Auflistung einzelner aufeinanderfolgender Interaktionsschritte. Diese Art der Beschreibung ermöglicht eine Erweiterung der beschriebenen Interaktionsfolge um mögliche Alternativ- und Ausnahmefälle (siehe *Abbildung 31*). Noch strukturierter ist die Darstellung eines Szenarios als Tabelle. Jede Spalte beschreibt dabei einen Benutzer oder das System. Die Zeilen sind von oben nach unten in chronologischer Reihenfolge aufgebaut. Szenarien können jedoch auch mit Grafiken, Bildern oder Zeichnungen der Anwendung oder der Umgebung ergänzt werden.

Eine weitere Kategorie befasst sich mit den *Inhalten* von Szenarien. Diese können in ihrer Ausprägung stark variieren. So kann sich der Inhalt ausschließlich auf Verhaltensinformationen wie Aktivitäten, Aktionen oder Ereignisse beziehen. Andererseits könnten organisatorische Informationen der Struktur einer Firma, ihr Aufbau und das Zusammenspiel einzelner Abteilungen oder Angestellter dargestellt werden. In manchen Fällen beschreibt ein Szenario jedoch auch ganz persönliche Charakteristiken wie Ziele, Wünsche oder Einschränkungen bestimmter Stakeholder. Rolland bezeichnet diese inhaltliche Ausprägung eines Szenarios als die „Berichterstattungsfacette“ (Rolland et al., 1998) und unterteilt die beschriebenen Inhalte in funktionale, nicht-funktionale und intentionale Aspekte. Ein weiterer inhaltlicher Aspekt ist der „Abstraktionsgrad“ eines Szenarios. In konkreten Szenarien, auch als „Instanzszenarien“ bezeichnet, werden konkrete Benutzer und Eingabewerte beschrieben. In abstrakten Szenarien reicht eine grobe Umschreibung der Rolle wie beispielsweise der eines „Kunden“ aus. Die von Rolland als „Kontextfacette“ bezeichnete inhaltliche Eigenschaft bezieht sich auf den Abstand des beschriebenen Inhaltes zu dem zu entwickelnden System. So kann zwischen „Systeminternen-, Interaktions- und Kontextszenarien“ unterschieden werden. Systeminterne Szenarien können dafür verwendet werden, Funktionsweisen innerhalb eines Systems, beispielsweise die Kommunikation einzelner Komponenten untereinander, zu beschreiben. Interaktionsszenarien beschreiben die Interaktion eines oder mehrerer Benutzer mit dem System. Kontextszenarien können als Erweiterungen der Interaktionsszenarien angesehen werden, da sie auch die Kommunikation zwischen Systemnutzern berücksichtigen, die nicht direkt mit dem System interagieren (Pohl, 2008, S. 135).

Der *Zweck* der Verwendung eines Szenarios wird von Rolland durch drei verschiedene Eigenschaften charakterisiert (Rolland et al., 1998). Deskriptive Szenarien verdeutlichen einzelne Abläufe und Interaktionen. Sie werden

in der Regel von mehreren Personen gemeinsam erstellt, um Anforderungen zu konkretisieren und noch ungeklärte Operationen, Ereignisse oder Stakeholder zu identifizieren. Explorative Szenarien werden zur Analyse und Bewertung alternativer Lösungsansätze verwendet (Rolland et al., 1998). Das Szenario beschreibt dabei die möglichen Alternativen und unterstützt damit eine mögliche Entscheidungsfindung im weiteren Entwicklungsprozess. Erklärende Szenarien sind in Situationen nützlich, in denen offene Fragen Erklärungen erfordern. In solchen Szenarien werden die jeweiligen Situationen mit den Begründungen detailliert dargestellt.

Als abschließender Punkt sollte abgewogen werden, zu welchem Zeitpunkt des Entwicklungsprozesses mit den Szenarien gearbeitet werden soll. Rolland bezeichnet diesen Punkt als den *Lebenszyklus* der Szenarien. So sollte von der Projektleitung überlegt werden, ob die Szenarien lediglich vorübergehend oder langanhaltend im Entwicklungsprozess verwendet werden. Rolland bezeichnet dies als die Lebensspanne der Szenarien. „Transiente Szenarien" bzw. „vergängliche Szenarien" werden ausschließlich für die Ermittlung der Anforderungen oder der Klärung einer komplizierten Fragestellung eingesetzt und danach verworfen. Anders verhält es sich bei „Persistenten Szenarien". Sie werden über den gesamten Entwicklungsprozess wiederverwendet und können sogar als Bestandteil eines Spezifikationsdokuments[9] angesehen werden. Vor allem wenn sie der Spezifizierung von Anforderungen dienen. Den Aspekt, ob ein Szenario verfeinert oder erweitert werden darf, bezeichnet Rolland als die „Einsatzfacette".

Anhand der Klassifizierung von Rolland konnte gezeigt werden, dass Szenarien sehr vielseitig einsetzbar sind. Die verschiedenen Typen und Facetten ergänzen den bereits vorgestellten Prozess des Scenario-based-Designs. Ein bisher nur kurz ausgeführter Aspekt, der speziell für die Ermittlung von Anforderungen wichtig ist, ergibt sich aus der folgenden Definition:

> *„Ein Szenario beschreibt ein konkretes Beispiel für die Erfüllung bzw. Nichterfüllung eines oder mehrere Ziele. Es konkretisiert dadurch eines oder mehrere Ziele. Ein Szenario enthält typischerweise eine Folge von Interaktionsschritten und setzt diese in Bezug zum Systemkontext."* (Pohl, 2008, S. 123)

Hinter dieser Definition steckt ein Verfahren, welches vor allem im Bereich des Requirements-Engineerings Anwendung findet. Dieses wird im Detail in Kapitel 3.3.1 vorgestellt. Die Möglichkeit mit Szenarien nicht nur eine Interaktionsfolge sondern einen gesamten Anwendungsfall bzw. Use-Case zu beschreiben, wird im Folgenden diskutiert.

2.3.3.2 Use-Cases

Der Begriff des Use Cases ist in der Informatik ein überladener Begriff. Als Synonym für Use Case wird häufig die deutsche Übersetzung „Anwendungsfall" verwendet. Use Cases repräsentieren nach Constantine die Funktionalität eines Systems, die der Benutzer von außen wahrnehmen kann (Constantine, 1995). Häufig werden unter dem Begriff auch Übersichtsgrafiken verstanden wie sie in der UML verwendet werden, ebenso können sie jedoch auch als textuelle Beschreibungen verstanden werden. So definiert beispielsweise Alistair Cockburn einen Use Case als eine Sammlung von Szenarien, die alle ein Ziel erfüllen (Cockburn & Dieterle, 2008). Alle verfolgen das Ziel einer besseren Übersichtlichkeit. In sämtlichen Entwicklungsprozessen gilt es, die rasch komplex werdende Funktionsvielfalt in zusammengehörige Einheiten aufzuteilen. Im weiteren Entwicklungsprozess werden diese dann Schritt für Schritt näher spezifiziert. Von der Object Management Group wird die in *Abbildung 31* dargestellte Attributierung für einen Use Case vorgeschlagen.

[9] Im englischen Sprachgebrauch als „specification" bezeichnet IEEE Std 610.12 (1990). Dokument, das alle relevanten Produktcharakteristiken, wie Anforderungen, Design und die einzelnen Systemkomponenten beschreibt.

ID	UC-1
Title	Export Requirement Specifications
CHARACTERISTIC INFORMATION	
Goal in Context	A user of a requirements authoring tool wants to export requirement specifications and relations between them from the requirements authoring tool to an exchange XML document.
Preconditions	The user has a requirements authoring tool installed. The user has a ReqIF tool installed that is capable of exporting requirement specifications from this requirements authoring tool. The requirement specifications the user wants to export are available in the requirements authoring tool and their contents are accessible by the user.
Success End Condition	The requirement specifications the user wanted to be exported have successfully been exported from the requirements authoring tool to an exchange XML document.
Failed End Condition	The requirement specifications the user wanted to be exported have not successfully been exported from the requirements authoring tool to an exchange XML document.
Primary Actor	The user of a requirements authoring tool.
MAIN SUCCESS SCENARIO	
Step 1	The user uses the ReqIF tool to specify the requirements specifications he wants to export and to request the export of the requirements specifications.
Step 2	The ReqIF tool exports each specification to one or several exchange XML documents. The exported exchange XML documents include information about requirements, types, attributes, and (optionally) access policies relations between requirements; the relations may be grouped. The structure of the specifications.
ALTERNATIVE SCENARIOS	
Alternative B: Export Parts of a Specification (Step 1 + Step 2)	Instead of exporting complete requirement specifications, a ReqIF tool MAY additionally have the feature to export only parts of a specification.

Abbildung 31: Notation eines Use Cases (Object Management Group, 2011b, S. 18)

Anhand dieser Spezifikation wird ein weiterer wichtiger Aspekt von Use Cases deutlich. Use Cases sollten nach dieser Definition um die Attribute Ziel, Vorbedingung, Nachbedingung bei Erfolg, Nachbedingung bei Misserfolg und einen Primärbenutzer ergänzt werden können. Pohl bezeichnet diese zusätzlichen Informationen als „Kontextinformationen“. Häufig existiert mehr als nur eine Interaktionsfolge, die zu einem definierten Ziel führen kann. Pohl unterscheidet somit drei verschiedene Typen, die zusammen einen Use-Case ergeben. Die typische Interaktionsfolge, die zur Erfüllung eines Ziels führt nennt er das Hauptszenario.

> *"Ein Hauptszenario ist ein Szenario, das die Interaktionsfolge dokumentiert, die normalerweise ausgeführt wird, um eines oder mehrere mit dem Szenario assoziierte Ziele zu erfüllen."* (Pohl, 2008, S. 137)

Bei der Erstellung eines Szenarios tritt häufig die Situation ein, eine alternative Interaktionsfolge beschreiben zu wollen. Dieses Problem kann auf zweierlei Wegen gelöst werden. Zum einen wird entweder die Alternative in Form eines explorativen Szenarios nach dem Schema von Rolland mit einer kurzen Erläuterung in das Szenario hineingeschrieben, oder es wird an der entsprechenden Stelle ein Verweis auf ein alternatives Szenario gesetzt. Pohl bezeichnet dieses als Alternativszenario:

> *„Ein Alternativszenario ist ein Szenario, das zu einem Hauptszenario eine alternative Interaktionsfolge definiert. Ein Alternativszenario erfüllt die gleichen Ziele wie das zugehörige Hauptszenario.“* (Pohl, 2008, S. 137)

Manchmal treten in einer Interaktionsfolge auch Ausnahmen ein, die von dem System abgefangen werden sollten. Dies sind meist Ereignisse die einen Arbeitsprozess stören und das Erreichen des eigentlichen Zieles verhindern. Diese Störungen können innerhalb eines Use-Cases als Ausnahmeszenarien bezeichnet werden. Zur Verdeutlichung des Zusammenspiels von Haupt-, Alternativ- und Ausnahmeszenarien dient Abbildung 32.

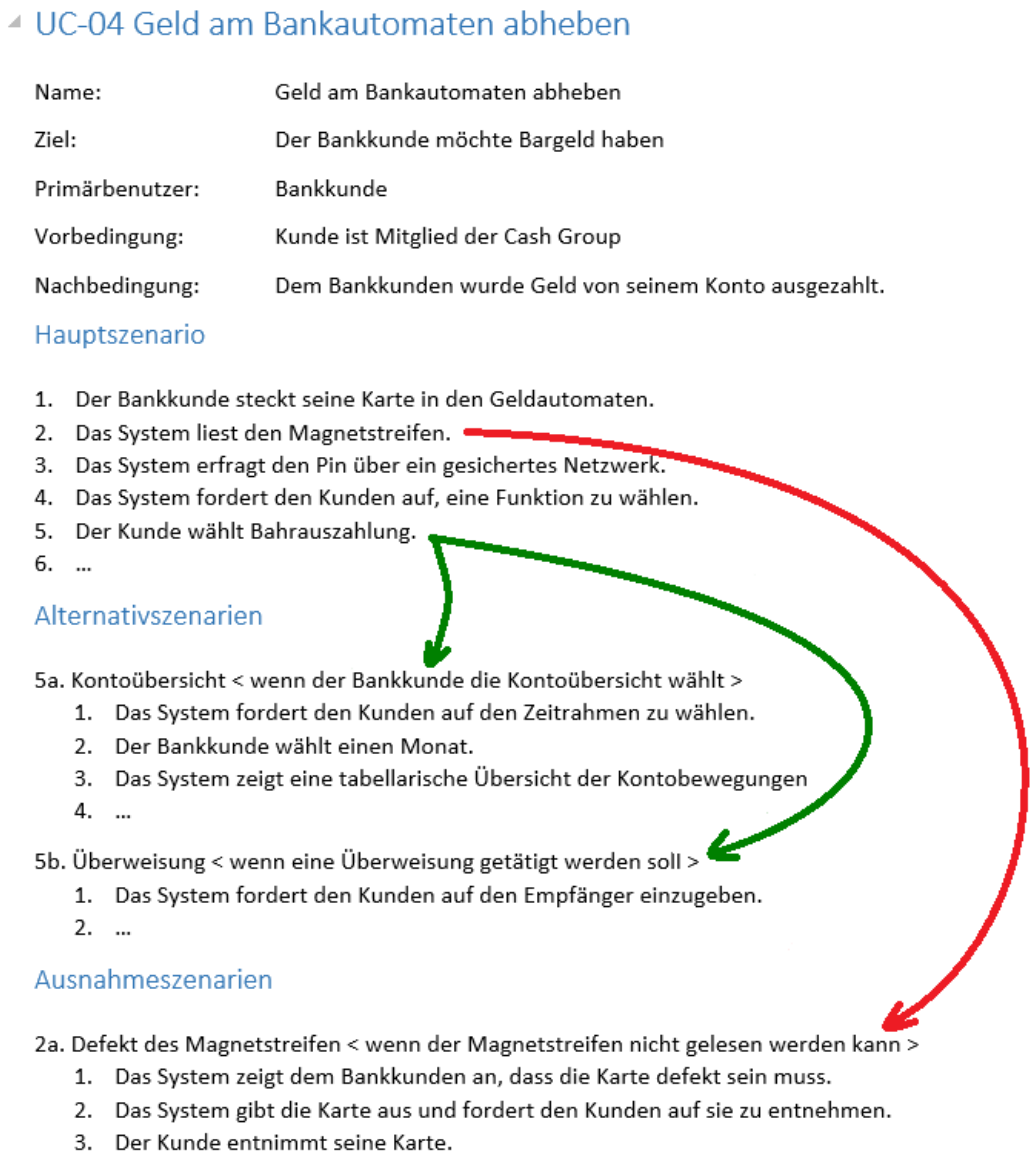


Abbildung 32: Aufbau eines Anwendungsfalls mit Szenarien

In dem Beispiel ersetzt das erste Alternativszenario einen Teil des Hauptszenarios. Das zweite Alternativszenario wird zusätzlich zu der Interaktionsfolge des Hauptszenarios ausgeführt. Tritt das Ausnahmeszenario ein, so wird das Hauptszenario unterbrochen.

2.3.3.3 Storyboards

Die fortlaufende Erzählung eines Szenarios kann auch als Sequenz von Bildern in Form eines Storyboards umgewandelt werden (Constantine & Lockwood, 1999). Abhängig vom Kommunikationszweck kann ein Storyboard in unterschiedlichen Ausprägungen erstellt werden. Dies reicht von skizzenartigen oder realistisch gestalteten Abfolgen der Benutzungsschnittstellen bis hin zu Bildergeschichten, die auch Kontext und handelnde Personen darstellen. Ebenso sind Mischformen aus beidem möglich.

Ein Storyboard zeigt mithilfe der Benutzungsschnittstelle, wie ein System oder Produkt verwendet wird. Es stellt wichtige Aspekte der Anwendung bildlich dar und dient damit der Kommunikation zwischen allen Beteiligten. Im Wesentlichen handelt es sich dabei um die Visualisierung eines Szenarios, indem es das Produkt im Umfeld des Benutzers zeigt.

2.3.4 Evaluation

Mit Hilfe der Evaluation wird im UE entschieden, ob ein Gestaltungsansatz bereits die gestellten Anforderungen erfüllt oder ob dieser weiter überarbeitet werden muss. Die Methode dient der Bewertung, „[…] ob ein Konzept, Design, Prototyp oder Produkt den Wünschen und Bedürfnissen der Benutzer entspricht." (Grechenig et al., 2010, S. 549). Abhängig vom Fortschritt des zu evaluierenden Gestaltungskonzeptes, kann zwischen einer *formativen* und einer *summativen* Evaluation unterschieden werden. Formative Evaluationen werden in frühen Phasen des Prozesses eingesetzt, um ein besseres Verständnis der Erfordernisse der Benutzer zu gewinnen. Sie ermöglichen eine frühzeitige Intervention bzw. Korrektur der bisherigen Zwischenergebnisse. Bei einer summativen Evaluation werden die erzielten Ergebnisse zu den anfangs formulierten Anforderungen bewertet. Für beide Arten der Evaluation existieren eine Reihe verschiedener *Kriteriensysteme*, welche diesen Bewertungsprozess messbar gestalten. Erst anhand dieses Kriteriensystems kann das UE als analytische Wissenschaft angesehen werden (Herczeg, 2009). Viele dieser stammen aus dem Bereich der Software-Ergonomie. Einige wurden in international geltende Standards übernommen, andere sind als Empfehlungen in diversen Veröffentlichungen zu finden. Die Kriteriensysteme vieler Standards sind häufig in einer Weise formuliert, um einen möglichst breiten Geltungsbereich zu erfüllen. Daher sollten sie für den jeweiligen Anwendungsbereich angepasst werden. Die Normreihe ISO 9241, die sich mit der Ergonomie der Mensch-Maschine-Interaktion befasst, beschreibt eine Reihe verschiedener Kriteriensysteme für diverse Anwendungsbereiche. Abbildung 33 veranschaulicht verschiedene Kriteriensysteme dieser Normreihe, die sich vor allem mit der Dialogschnittstelle von Software befassen.

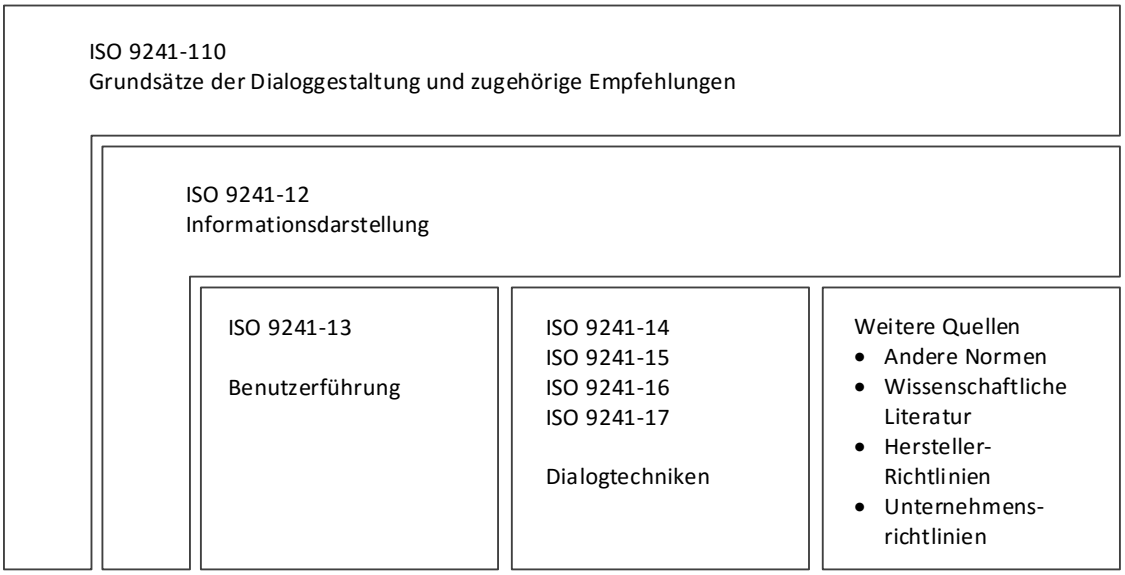


Abbildung 33: Gestaltungsempfehlungen der Normreihe ISO 9241 (nach ISO 9241-110)

In der ISO 9241-110 aufgeführte Dialogkriterien mit jeweiligen Erläuterungen und Beispielen betreffen folgende Punkte:

- Aufgabenangemessenheit
- Selbstbeschreibungsfähigkeit
- Erwartungskonformität
- Lernförderlichkeit
- Steuerbarkeit
- Fehlertoleranz
- Individualisierbarkeit

Darüber hinaus umfasst die Normreihe auch Leitsätze zur Aufgabengestaltung (ISO 9241-2), ergonomische Grundsätze für physikalische Eingabegeräte (ISO 9241-400) sowie Kriterien für die Ergonomie der Arbeitsumgebung (ISO 9241-600).

Als Basiskriterien für die Gebrauchstauglichkeit von Benutzungsschnittstellen kann die Definition für Gebrauchstauglichkeit aus der ISO 9241-11 angesehen werden. Die Gebrauchstauglichkeit ist nach der Norm folgendermaßen definiert:

> *„Das Ausmaß, in dem ein Produkt durch bestimmte Benutzer in einem bestimmten Nutzungskontext genutzt werden kann, um bestimmte Ziele effektiv effizient und zufriedenstellend zu erreichen."*

Der Nutzungskontext aus dieser Definition steht dabei für das Zusammenspiel aus Benutzern, der zu lösenden Aufgaben, der dabei verwendeten Arbeitsmittel und der jeweiligen Umgebung in der das Produkt eingesetzt werden soll (siehe auch Abbildung 1). Dies verdeutlicht erneut, dass bei der Evaluation eines Softwareproduktes der gesamte Nutzungskontext berücksichtigt und bewertet werden sollte, um eine umfassende Aussage über den Grad der Gebrauchstauglichkeit zu erhalten. Die Effektivität beschreibt dabei die Genauigkeit und Vollständigkeit mit der ein Benutzer seine Ziele erreichen kann. Die Effizienz steht für den dabei eingesetzten Aufwand des Benutzers. Die Zufriedenheit spiegelt die Freiheit von Beeinträchtigungen und der positiven Einstellung bei der Verwendung des Produktes wider.

Funktionale Kriterien betreffen die Konstruktion und Strukturierung eines interaktiven Softwaresystems und sollten als generell geltende Kriterien in der Software- und Systementwicklung verstanden werden. Hierzu zählen folgende Punkte (Herczeg, 2009):

- Verfügbarkeit
- Zuverlässigkeit
- Wiederverwendbarkeit
- Kombinierbarkeit
- Erweiterbarkeit
- Komplexität
- Transparenz

In der Prozessbeschreibung der ISO 9241-210 zur Gestaltung gebrauchstauglicher Software (siehe Kapitel 2.2.1) entscheidet der Prozessschritt der Evaluation darüber, ob ein Produkt den Ansprüchen eines Benutzers entspricht oder ob anhand einer weiteren Iteration das Produkt weiter optimiert werden muss. Alle Aktivitäten der Designer und Entwickler bis zum Zeitpunkt der Evaluation beruhen auf Annahmen. Erst eine Bewertung des aktuellen Lösungsansatzes anhand einer Evaluation ermöglicht es, dass UE als präzise und messbare Ingenieursdisziplin zu betreiben.

Bei der Evaluation können zwei grundlegende Herangehensweisen unterschieden werden. Evaluationen mit Benutzerbeteiligung werden als *empirische Methoden* bezeichnet ohne eine Beteiligung von Benutzern als *inspektionsbasierte bzw. analytische Methoden*. Obwohl eine Evaluation durch die Benutzer generell zu bevorzugen ist, weist selbst die ISO Norm 9241-210 darauf hin, dass die Bewertung durch Benutzer nicht in jeder Phase des Projektes praktikabel und kostengünstig ist (ISO 9241-210:2011, S. 22).

2.3.4.1 Benutzerorientierte Evaluation

Bei Empirischen Methoden werden Probleme und Fehler gemeinsam mit dem Benutzer erfasst. Zu diesem Zweck gibt es eine Vielzahl an Möglichkeiten. Diese hängen damit zusammen, wie viel Zeit laut Auftraggeber dafür aufgewendet werden kann und wie kooperativ die jeweiligen Stakeholder sind. Die Methoden reichen dabei von *Interviews, Fragebögen* oder *Usability Tests* bis zu Methoden wie *Thinking Aloud*, *Eye Tracking* oder *Feldvalidierungen*. Im besten Fall sollten die Zielbenutzer für die Bewertung schon sehr früh in den Entwicklungsprozess involviert werden.

So können den Stakeholdern nach der ISO 9241-210 bereits erste Skizzen, Modelle oder Szenarien vorgelegt werden. Hier können sämtliche Formen von Prototypen, die in dem Kapitel 2.3.3 beschrieben wurden, zur Anwendung kommen. In manchen Fällen erhöht eine frühe Involvierung die Akzeptanz der Benutzer, da ihnen das Gefühl der Mitbestimmung vermittelt wird. Die verwendeten Modelle sollten Interaktionen des tatsächlichen Arbeitsablaufs der Benutzer simulieren. Reine Demonstrationen sind für Verbesserungen der Lösungsansätze nicht so hilfreich wie Simulationen der realen Arbeitsbedingungen anhand von Prototypen. Ein weitere von der Norm empfohlene Form der Evaluation sind die sogenannten *Feldvalidierungen*, bei denen das System im realen Umfeld

getestet werden soll. Dazu werden meist sogenannte Beta-Versionen des Produktes eingesetzt. Dem Benutzer sollte dabei vermittelt werden, dass es sich um eine noch nicht fertige Version handelt. Informationen dieser Feldvalidierungen können in Form von Fehlermeldungen, Hotlines oder Benutzerberichten in den Entwicklungsprozess zurückfließen. (ISO 9241-110:2008)

Bei *Usability Tests* wird ebenfalls mit realen Benutzern gearbeitet. Diese Testpersonen sind nach Grechenig Repräsentanten echter Benutzer. Den Probanden werden praxisnahe Aufgaben gegeben, die im Vorwege, beispielsweise durch Use Cases (siehe Kapitel 2.3.3.2), definiert wurden, die sie mit dem jeweiligen System lösen sollen. Dabei werden die Benutzer von einer Gruppe von Experten beobachtet. Die Beobachtungen sollten dabei aufgezeichnet werden, um sie später analysieren zu können. (Grechenig et al., 2010)

Eine leichte Modifikation der Usability Test ist das *Thinking Aloud* mit nahezu identischem Testaufbau. Der Unterschied besteht darin, dass die Probanden dazu angehalten werden ihre Gedanken und Überlegungen, während der Bewältigung der Aufgaben zu kommunizieren. Der Vorteil dieser Methode für das Testteam ist es, dass die Gedankengänge der Probanden dadurch sichtbar werden. In einer weiteren Studie von Jacob Nielsen konnte gezeigt werden, dass mit dieser Methode über 40 % der schweren Usability Probleme erkannt werden (Nielsen, 1992).

Fragebögen zählen zu den quantitativen Evaluationen. Sie ermöglichen die Erhebung großer Datenmengen. Fragebögen werden in der Regel von den Probanden ohne Beisein eines Versuchsleiters durchgeführt. Somit sollte der Fragebogen sehr verständlich geschrieben sein. Neben gedruckten Fragebögen gibt es auch online- Lösungen. Neben einer kleinen Einleitung, in der die Probanden über das Vorhaben und die Ziele der Evaluation informiert werden sollte, können auch problemrelevante demografische Daten erhoben werden. Dann folgen in der Regel die eigentlichen Fragen bzw. Items. Die Items können entweder über eine Skala oder als Freitext von den Probanden beantwortet werden. Eine bekannte Skala für solche Fragebögen ist die „Likert-Skala“ mit einer geraden oder ungeraden Anzahl an Werten, die eine Aussage über die persönliche Einstellung von *trifft völlig zu* bis *trifft überhaupt nicht zu*, treffen soll. Ein Fragebogen, der auf die Dialogkriterien der DIN EN ISO 9241-110 aufbaut, stammt von Jochen Prümper (Prümper & Anft, 2009). Dieser ist in sieben Gruppen von Items unterteilt, wobei jede Gruppe ein Dialogkriterium aus der Norm beschreibt.

Nach der ISO/TR 16982 haben *Interviews* große Ähnlichkeit zu Fragebögen. Sie werden jedoch im direkten Kontakt mit den jeweiligen Stakeholdern durchgeführt, wodurch flexibler auf Probanden eingegangen werden kann. Es kann auch auf Fragen eingegangen und nachgefragt werden, wodurch auch unerwartete Punkte aufkommen können. Interviews sind zwar zeitaufwändiger als Fragebögen, können unter Umständen jedoch zu aussagekräftigeren Antworten führen. Interviews haben unterschiedliche Ausprägungsformen, von sehr strukturierten bis hin zu Interviews mit einem offenen Ende. (ISO/TR 16982:2002)

Egal welche Evaluationsmethode auch angewendet wird, am Ende kommt immer eine Liste mit unterschiedlichen Problemen zustande. Häufig ist eine solche Liste zu lang, um alle Probleme gleichzeitig lösen zu können. Somit sollten sie in einer geeigneten Form priorisiert werden. Eine sehr naheliegende Form priorisiert die Probleme nach ihrem Schweregrad. Häufig wird dieses Verfahren auch als *Severity Rating* bezeichnet.

2.3.4.2 Inspektionsbasierte Evaluation

Inspektionsbasierte Evaluationen sollten nach Möglichkeit vor einer Evaluation mit Benutzern eingesetzt werden und als Ergänzung angesehen werden. Sie werden in der Regel von Fachleuten durchgeführt. Normalerweise werden dazu *Checklisten*, *Richtlinien*, *Normen* oder *Heuristiken* eingesetzt, anhand derer Fehler und Verbesserungen frühzeitig erkannt werden sollen. Inspektionsbasierte Evaluationen werden auch als *dokumentenbasierte Methode* bezeichnet (ISO/TR 16982:2002).

Eine Bewertung sollte nach Möglichkeit immer sehr realitätsnah angelegt sein (ISO 9241-230:2009). Mit dem sogenannten *Cognitive Walkthrough* versetzt sich ein Experte in die Rolle des Benutzers und erforscht das System aus dessen Blickwinkel. Speziell bei dieser Methode ist es wichtig, im Vorwege möglichst viele Informationen über den Benutzer, dessen Aufgaben und weitere Kontextinformationen in Erfahrung gebracht zu haben. Der Experte überprüft bei dieser Methode, wie effektiv, effizient und zufriedenstellend sich mit dem System die zuvor

definierten Ziele des Benutzers erfüllen lassen, um anschließend mögliche Verbesserungen des Systems zu initiieren. Der Vorteil dieser Methode liegt im geringeren Zeitaufwand, welcher sich in der Regel direkt auch auf die Kosten auswirkt.

Die *heuristische Evaluierung* wurde in den 1990er-Jahren von Nielsen entwickelt (Grechenig et al., 2010). Der Begriff „Heuristik" wird vom Duden als „[…] methodische Anleitung […] zur Gewinnung neuer Erkenntnisse" (Duden, 2013) gedeutet. Diese Methode benötige nach Nielsen einige Erfahrung, um sie in jeder Situation anwenden zu können. Dafür sei es auch „Nicht-Experten" möglich, Usability Problemen zu erkennen (Nielsen, 1993).

> *„Heuristiken sind allgemein formulierte, grobe Richtlinien, die im Allgemeinen von einer Gruppe von Experten definiert wurden und im jeweiligen Fachbereich von anderen Experten anerkannt werden."* (Grechenig et al., 2010, S. 550)

Richtlinien listen gut bekannte Prinzipien aus dem Interface-Design auf, die während der Entwicklung berücksichtigt werden sollten. In jedem Projekt sollte es mehrere Richtlinien geben. Hierzu zählen sowohl generell geltende als auch Anwendungs- bzw. domänenspezifische Richtlinien (Nielsen, 1993, S. 91). Richtlinien können jedoch auch für die Codierung oder andere Inhalte einer Anwendung gelten (Eller, 2009). Eine allgemein bekannte Form einer Richtlinie sind *Styleguide*. Styleguides dienen primär der Wahrung der Konsistenz einer Anwendung. Sie beschreiben beispielsweise die Gestaltung einzelner Interaktionselemente, deren Anordnung sowie die Verwendung von Begriffen und Icons (Dahm, 2006). Ein Styleguide kann für ein spezielles Produkt als festgelegte Richtlinie angesehen werden und sollte als Bezugsquelle für eine kontinuierliche Überprüfung der Einhaltung der beschriebenen Eigenschaften über den gesamten Lebenszyklus verwendet werden. Diese Methode kann ebenfalls als *Guideline Review* bezeichnet werden. Ein im HCI-Bereich sehr bekanntes Beispiel für eine allgemein gültige Richtlinie sind die „Acht goldenen Regeln" von Ben Shneiderman (Shneiderman, 1998).

Wie bereits eingangs erwähnt sollten inspektionsbasierte Ansätze generell als Zusatzmaßnahme für das Aufspüren möglicher Fehler verwendet werden. Dadurch soll verhindert werden, dass Probleme zu spät oder im schlimmsten Fall gar nicht erkannt werden.

2.3.4.3 Szenarienbasierte Evaluation

Als eine Mischung aus benutzerzentrierter und inspektionsbasierter Evaluation kann die szenariobasierte Evaluation angesehen werden. Szenarien eignen sich wie bereits in Kapitel 2.3.3.1 erläutert sowohl für die Darstellung des Anwendungskontextes für den Entwickler, als auch zur Verdeutlichung von Lösungsansätzen für den Benutzer. Im vorhergehenden Spezifikationskapitel 2.3.2 wurde grob auf die Kopplung von Zielen und Szenarien eingegangen. Nachfolgende Abbildung 34 erweitert dieses Modell um die sogenannten lösungsorientierten Anforderungen (siehe Kapitel 3.3.1).

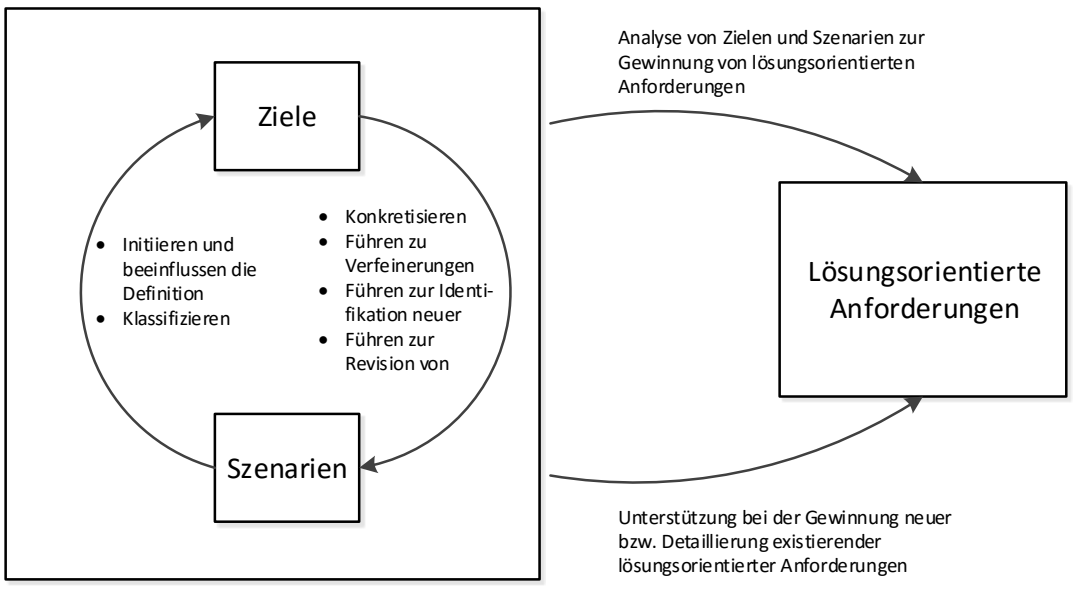


Abbildung 34: Goal-Scenario-Coupling (Pohl, 2008)

Die Abbildung verdeutlicht einen wichtigen Aspekt in der Methode der Evaluation. Meist erst nach mehreren Iterationen der Zieldefinition sowie deren Konkretisierung in Form von Szenarien, kann der Aufwand der tatsächlich zu implementierenden Anforderungen (Lösungsorientierte Anforderungen) abgeschätzt werden. Dieses Modell entspricht daher auch dem Prozessmodell zur Entwicklung gebrauchstauglicher interaktiver Systeme aus Kapitel 2.2.1 der ISO 9241-210. In diesem entscheidet der Punkt „Lösung aus Benutzerperspektive evaluieren“ (Abbildung 3), ob ein Lösungsansatz weiter verbessert, spezifiziert, konkretisiert oder bereits implementiert werden kann.

2.4 Ansätze und Werkzeuge des UE

Speziell in dem Bereich des UE waren die den gesamten Entwicklungsprozess umfassenden Werkzeuge kaum zu finden. In den letzten drei Jahren scheint jedoch hierfür etwas Bewegung in die Entwicklungen gekommen zu sein. Die meisten Softwarewerkzeuge, die eindeutig dem UE zuzuordnen waren, beschränken sich jedoch häufig nur auf einzelne Methoden aus dem UE.

2.4.1 CREWS

In den Jahren von 1996 bis 1999 befasste sich eine internationale Arbeitsgruppe thematisch mit der Verwendung von Szenarien in der Anforderungsanalyse. In dieser Zeit wurden Methoden und vier prototypische Werkzeuge entwickelt, welche die „[…] Erfassung und Validierung von Anforderungen, basierend auf konkreten Anwendungsszenarien […]“ (Wolter, 2004), ermöglichen sollen. Dabei sollten die Defizite der rein zielorientierten und der rein Szenarien basierten Ansätze über die Kombination beider umgangen werden. Die dabei verwendete Notation sieht die Verwendung von sogenannten Requirement Chunks (RC) vor. Ein solcher Chunk besteht immer aus einem Paar <G, Sc>, bestehend aus einem Ziel (G) und einem Szenario (Sc). Somit beschreibt ein solcher Chunk immer genau einen möglichen Weg einer Zielerreichung. Die dabei auftretenden Verbindungsmöglichkeiten zwischen diesen Chunks sind in Abbildung 35 zu sehen.

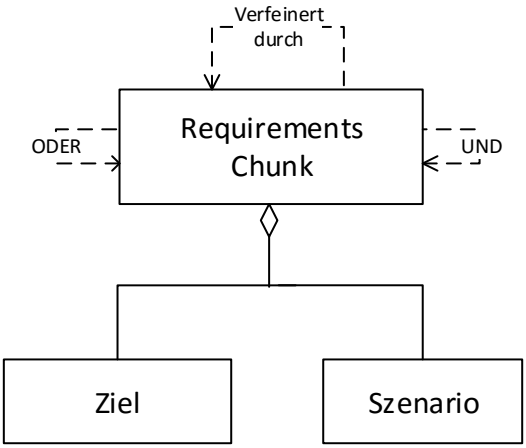


Abbildung 35: Datenmodell eines "Requirement Chunks" (Tawbi & Souveyet, 1999)

Ein einzelner Requirement Chunk kann über eine „UND“, „ODER“ oder „Verfeinerung“ mit einem anderen Chunk verbunden sein. Die drei Relationen ermöglichen die Bildung einer Hierarchie von Requirement Chunks. Ein Ziel wird als Satz formuliert. Bei den Szenarien existieren zwei Arten von Szenarien. Die „normalen“ Szenarien dienen der Erfüllung eines Ziels, und die „außergewöhnlichen“ Szenarien führen zur Nichterfüllung des Ziels. Darüber hinaus besitzt jedes Szenario einen Initial- und einen Finalstatus. Die in dem Szenario beschriebenen Aktionen führen vom Initial- zum Finalstatus. Aktionen selbst, können auch geschachtelt sein oder sie sind atomar. Eine Aktion beschreibt immer eine Interaktion zwischen zwei Akteuren, die ein Objekt beeinflussen. (Tawbi & Souveyet, 1999)

Das CREWS-Verfahren zielt darauf ab Anforderungen über die bidirektionale Kopplung von Zielen und Szenarien aufzuspüren, indem Ziele zu Szenarien führen, die wiederum neue Ziele aufdecken. Abbildung 36 soll dieses Verfahren verdeutlichen.

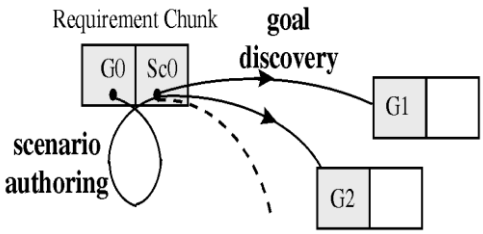


Abbildung 36: Überblick über das CREWS-Verfahren (Rolland, Souveyet & Achour, 1998)

Ein Ziel wird über das Schreiben eines Szenarios konkretisiert. Die Analyse des Szenarios wiederum führt wieder zu neuen Zielen bzw. Anforderungen. Dieser Prozess wird so häufig wiederholt bis alle Ziele/Anforderungen aufgedeckt sind. Rolland unterscheidet drei unterschiedliche Zielebenen: Kontextebene, Funktionsebene und physikalische Ebene. Die Funktionsebene beschreibt meist die Ziele der Interaktion zwischen Mensch und System, während die physikalische Ebene dann die Anforderungen beschreibt, die von dem System benötigt werden, um die Ziele auf der Funktionalen Ebene zu erfüllen.

2.4.2 REUSE

Im Jahre 2001 untersuchte eine Studie, in wie weit der HCD in vier großen deutschen Unternehmen angewendet wird (Metzker & Offergeld, 2001). In den untersuchten Unternehmen wurden sehr unterschiedliche Entwicklungsprozesse angewandt. Kaum eines der Unternehmen war mit den Methoden aus dem HCD wie beispielsweise der Benutzeranalyse oder dem *Kognitive Walkthrough* vertraut. Metzker und Offergeld identifizierten drei Hauptgründe für den mangelnden Einsatz von HCD-Methoden.

- Den HCD-Methoden wird zu wenig Zeit in den Entwicklungsprozessen zur Verfügung gestellt
- Die HCD-Methoden sind nur ungenügend in den Unternehmen bekannt
- Der Aufwand für die Verwendung der Methoden wird als zu hoch eingeschätzt.

Aus der Untersuchung ergaben sich eine Reihe unterschiedlicher Schlussfolgerungen, aus denen Anforderungen für ein mögliches Unterstützungswerkzeug aus dem HCD-Bereich abgeleitet wurden. So sollte das System beispielsweise die HCD-Prozessmodelle flexibel unterstützen. Das System sollte die Unternehmen nicht zu einem stringenten Vorgehen zwingen, sondern sich flexibel in bestehende Prozesse integrieren lassen. Die Unternehmen sollten selbst auswählen dürfen, welche Methoden zu ihrem jeweiligen Entwicklungsvorhaben am besten passen. Es sollten evolutionäre Entwicklungsprozesse unterstützt werden, in denen HCD-Erfahrungen wiederverwendet werden können. Eine weitere Anforderung war die generell fehlende Unterstützung durch Werkzeuge bei der Verwendung von komplexen HCD-Methoden. Zudem sollte das System die Effizienz bei der Verwendung der Methoden erhöhen.

Das dabei entwickelte System bekam den Namen REUSE (Repository for Usability Engineering). Das Tool soll das Auffinden, Organisieren und Wiederverwenden von HCD-Wissen ermöglichen. Dazu wurden sogenannte *USEPACKS* und *Context Models* entwickelt. Ein USEPACK ist eine semi-formale Notation, um Wissen über HCD-Aktivitäten zu strukturieren. Es kapselt Wissen über bewährte Verfahren aus dem HCD-Bereich, wie beispielsweise Dokumente, Codefragmente, Vorlagen und Werkzeuge. Zu diesem Zweck ist ein USEPACK in fünf Bereiche unterteilt. Im ersten steht immer die Kerninformation des USEPACKs und welchen Zweck es erfüllen soll. Die Autoren des USEPACKs werden dazu angehalten, die Informationen nach dem Pyramidenprinzip zu verfassen, d.h. die Komplexität der Informationen steigt stetig mit der Länge des Textes. Dadurch kann ein Leser schnell entscheiden, ob die Information für ihn relevant ist oder nicht. Der zweite Bereich beinhaltet Informationen zur Kontextsituation. Dieser wird durch die Context Models generiert und ermöglicht die Verwendung eines geteilten Vokabulars, um kontextabhängig auf USEPACKs zugreifen zu können. Der dritte Bereich kann mögliche externe Artefakte wie Checklisten, Fragebögen oder andere Dokumente beinhalten. Der vierte Bereich ist für semantische Verknüpfungen zu anderen USEPACKs oder für weitere externe Informationen gedacht. Im fünften und letzten Bereich werden Meta-Informationen über den Erstellungszeitpunkt oder über dem Autor festgehalten.

Das REUSE System besteht aus vier verschiedenen Komponenten. Mit dem *USEPACK editor* können neue USEPACKs erstellt werden. Der *USEPACK explorer* ermöglicht durch verschiedene Ansichten und Filter das Aufspüren von USEPACKs. So können beispielsweise USEPACKs mit der Filterfunktion für einen ganz bestimmten

Entwicklungsschritt gesucht werden. Mit Hilfe des *Context Model Managers* können die Context Models bearbeitet und erweitert werden. Die letzte Komponente von REUSE ist das *USEPACK servlet*. Dieses ermöglicht die Darstellung von USEPACKs in einem Browser.

2.4.3 Usability Planner

Der *Usability Planner* von Bevan Nigel, Xavier Ferré und Tomás Antón Escobar unterstützt Software-Entwickler bei der Auswahl von Methoden des User-centered Design (UCD) Prozess (Ferre, Bevan & Escobar, 2010). Abhängig von der Größe des Entwicklungsvorhabens, der verfügbaren Arbeitskräfte und der vorgegebenen Zeit wird aus einem Methodenkatalog ein Vorschlag generiert, der zu dem jeweiligen Projekt passen könnte. Die vom Usability Planner vorgeschlagenen Methoden werden größtenteils grob beschrieben, jedoch nicht mit integrierten und anwendbaren Werkzeugen unterstützt. Wirtschaftlich ist das System interessant, da es abhängig vom jeweiligen Entwicklungsstadium den relativen Kostenvorteil bei Verwendung der vorgeschlagenen Methoden berechnen kann.

Ferré sieht den Vorteil des Usability Planner in der Herangehensweise (Ferre et al., 2010). Bisherige Ansätze beginnen immer mit der Methode selbst und nicht mit der Frage nach dem Zweck, den diese erfüllen soll. Es wurden häufig nur die Stärken und Schwächen der einzelnen Methoden untersucht und nicht der Beitrag, den eine Methode zur benutzerzentrierten Entwicklung beitragen kann (Bevan, 2009). Ebenfalls schwierig für die Auswahl war die Vielfalt an Themen. Diese Aufgaben werden vom Usability Planner übernommen. Typ und Priorität der Methoden beruhen auf der ISO PAS 18152. Darüber hinaus werden Kriterien aus der ISO TR 16982[10] verwendet, um festzulegen, welche Methode am besten geeignet ist (ISO/TR 16982:2002).

Der Usability Planner ist als Webanwendung[11] entwickelt worden und frei verfügbar. Die Anwendung wurde in drei Teilabschnitte gegliedert. Im ersten Abschnitt wird die jeweilige Projektphase erfragt. Hier kann der Benutzer dann wählen, ob er bereits ein erstes Systemkonzept oder erste Anforderungen des Kunden hat oder nicht (siehe Abbildung 37). Anschließend gelangt der Benutzer in den Methodenabschnitt, in dem eine nach Projektphasen geordnete Liste mit den jeweiligen Methoden generiert wird. Über eine zusätzliche Filterfunktion kann die Liste dann noch verfeinert werden. Die Filterfunktion, wie beispielsweise „Brauche schnelle Resultate“ oder „eingeschränktes Budget“ verkleinert die Liste und zeigt gleichzeitig an, welche Methoden unter diesen Bedingungen am wichtigsten sind. Im letzten Abschnitt wird eine Übersicht angezeigt, welche Methoden in dem Projekt am besten angewendet werden sollte. Erwähnenswert ist auch die Möglichkeit, auf der Hauptseite seine eigene Benutzerrolle – als Entwickler oder Usability Experte – auswählen zu können. Dies führt dann in dem Abschnitt des Projektstatus zu einer unterschiedlichen Ansicht.

[10] Seit 2009 befindet sich die ISO TR 16982 in der Überarbeitung und wird zukünftig als Teil 230 die Normreihe ISO 9241 übernommen.

[11] Die Anwendung ist unter folgender URL zu finden: http://www.usabilityplanner.org

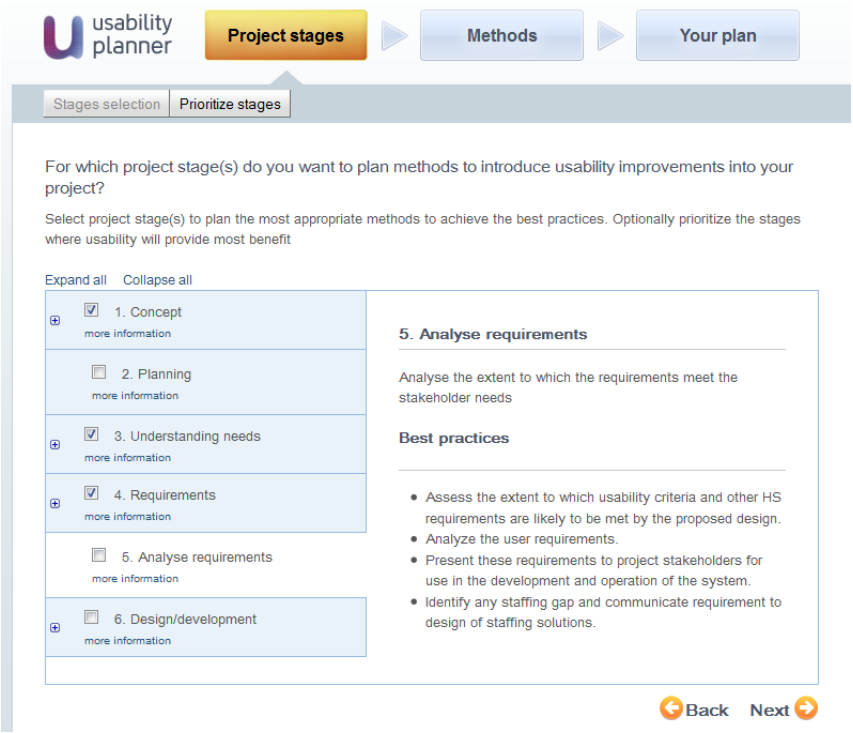


Abbildung 37: Auswahl der Projektphase im Usability Planner (http://www.usabilityplanner.org)

Der Usability Planner ist sehr einfach zu bedienen, und ein Benutzer erhält eine schnelle und unkomplizierte Übersicht der Methoden, die in seinem Projekt unter den angegebenen Bedingungen den größten Erfolg versprechen. Leider ist das System nicht hundertprozentig fehlerfrei, manchmal muss eine Eingabe auch wiederholt werden. Auch die Informationen zu den vorgeschlagenen Methoden sind sehr kurz gehalten oder nicht vorhanden. Für die Erstellung eines Projektplans mit den durchzuführenden Methoden ist das System jedoch sehr zu empfehlen.

2.4.4 CTTE

Das *Concur Task Tree Enviroment (CTTE)* ist eine Werkzeugunterstützung für die Erstellung hierarchischer Aufgabenmodelle nach der CTT-Notation (siehe Kapitel 2.3.1.2.1). Nach Paternò sei eine Systemunterstützung bei Verwendung von Aufgabenmodellen zwingend notwendig (Paternó, Mori & Galiberti, 2001). Erst sie ermöglicht eine breite Akzeptanz bei der Verwendung. Das CTTE erlaubt somit eine dynamische Erstellung und Analyse komplexer Aufgabenmodelle. Paternò sieht den Haupteinsatzzweck in der Unterstützung des Designs von Benutzungsschnittstellen. Nach einer informellen Aufgabenanalyse beim Kunden ist es wichtig, die identifizierten Hauptaufgaben miteinander in Beziehung zu setzen. Dadurch kann sowohl der Design- als auch der Evaluationsprozess interaktiver Systeme erleichtert werden. Die geringe Akzeptanz bei der Verwendung von Aufgabenmodellen sieht Paternò in dem großen Zeitaufwand für die Erstellung dieser. In manchen Fällen sei dieser sogar entmutigend. Abhilfe könne nur durch eine geeignete Werkzeugunterstützung wie dem CTTE geschaffen werden.

Das System unterstützt die Darstellung vier verschiedener Aufgabenkategorien. Eine abstrakte Aufgabe (Wolke) beschreibt eine Aufgabe mit Teilaufgaben. Eine Benutzeraufgabe (Männchen) ist eine Aufgabe, die durch einen Benutzer symbolisiert wird. Eine Anwendungsaufgabe (Computer) wird von dem System ausgeführt, und eine Interaktionsaufgabe ergibt sich aus der Zusammenwirkung eines Benutzers mit dem System. Beim Start des Programms wird automatisch eine abstrakte Aufgabe (Wolke) angezeigt. Um einen Knoten in dem Modell zu ergänzen, muss zuerst ein Knoten selektiert werden. Danach kann über das Menü auf der linken Seite einer der vier Aufgabenkategorien ausgewählt werden. Um einen Operator, wie beispielsweise „>>“, hinzuzufügen, muss der Knoten von dem aus der Operator starten soll markiert werden, um dann wieder über das Menü den entsprechenden Operator auszuwählen.

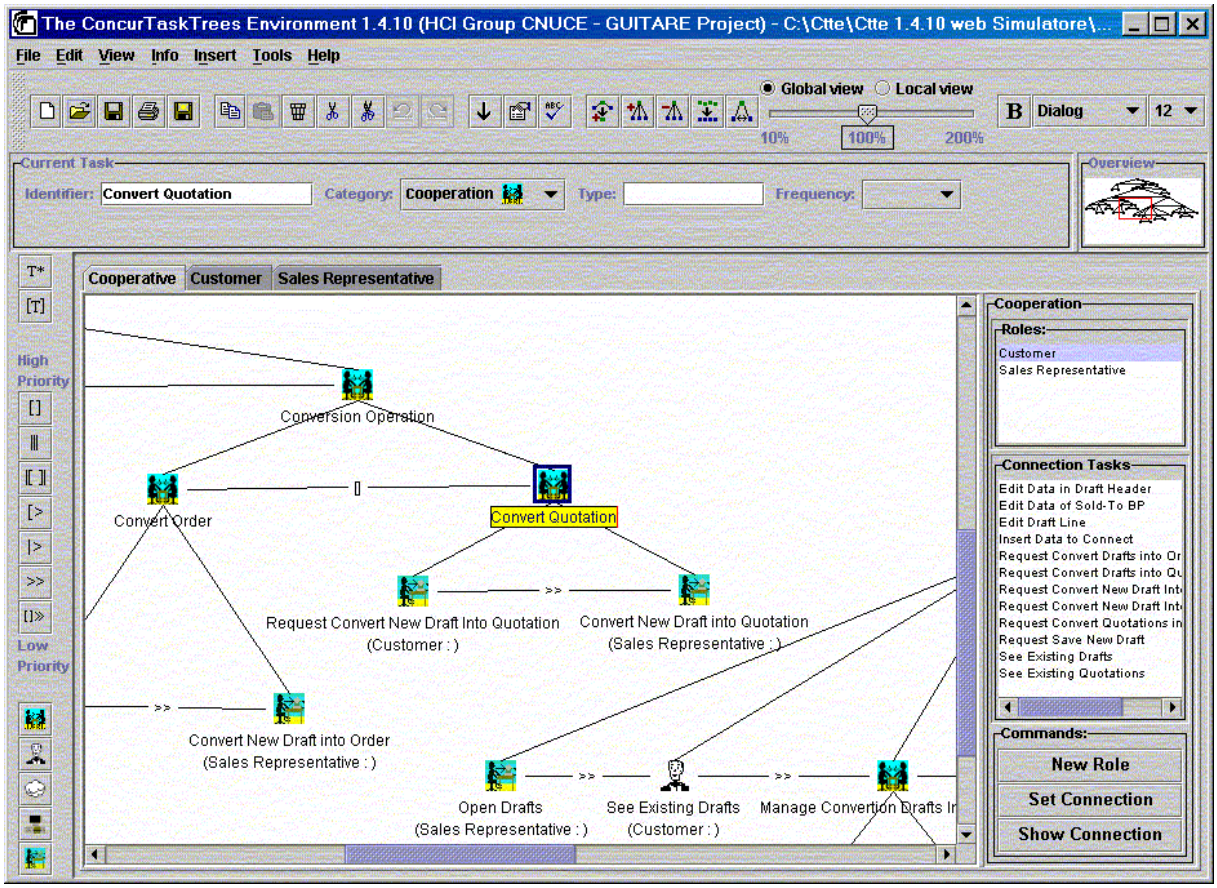


Abbildung 38: Concur Task Tree Enviroment (HIIS Laboratory, 2013)

Neben der hierarchischen Darstellung der Aufgabendekomposition können den Aufgaben zusätzliche Informationen hinzugefügt werden. Neben der Aufgabenbezeichnung (Identifier) und der Kategorie, die direkt in dem Diagramm angezeigt werden, kann sowohl die Häufigkeit der Ausführung, der Typ der Aufgabe, ihre Vorbedingung, als auch die verwendete Plattform wie ein Tablet oder ein stationäres Gerät ergänzt werden.

Über einen Simulator kann nach Fertigstellung des Aufgabenmodells das Verhalten getestet werden. Durch Festlegen eines Startknotens werden über eine Konsole die Reihenfolge der Aufgaben und mögliche Fehler angezeigt.

2.4.5 UsabilityTools

Die Firma *UsabilityTools*[12] wurde 2009 von zwei Usability Experten gegründet. Die Idee für das gleichnamige System entstand aus dem Umstand, für eine Benutzerstudie im Internet eine Vielzahl unterschiedlicher Werkzeuge verwenden zu müssen. Da diese sich für eine umfangreiche Analyse nicht miteinander integrieren ließen, entstand die Idee für UsabilityTools.

Das System ist primär für Webseiten ausgelegt. Dafür wurden eine Reihe unterschiedlicher Module entwickelt, die zum Testen und Optimieren einer Seite verwendet werden können. Mit Hilfe der Module aus der sogenannten User Experience Suite kann eine Benutzerumfrage gestartet werden. Dazu stehen vier Module zur Verfügung, die in beliebiger Reihenfolge aneinander gereiht werden können. In Abbildung 39 ist die Erstellung einer solchen Benutzerbefragung zu sehen. In diesem einfachen Beispiel muss der Benutzer sechs Schritte bei der Befragung durchlaufen. Er beginnt mit einer Willkommensseite, die der Benutzer als erstes zu sehen bekommt, wenn er an der Befragung teilnimmt. Im zweiten Schritt wird der Benutzer nach seinem Alter und dem Geschlecht befragt. Im dritten Schritt (Web Testing A) soll der Benutzer eine Aufgabe ausführen. Dazu muss in dem Modul die Aufgabe selbst formuliert werden, sowie die Start- als auch die Ziel-URL. Bei der Benutzerbefragung wird dem

[12] https://usabilitytools.com/

Benutzer dann die Frage wie auch die entsprechende Seite angezeigt. Wenn er die Zielseite erreicht hat, wird ihm angezeigt, dass er die Aufgabe erfolgreich erledigt hat. Das Modul „Web Testing“ misst dabei die benötigte Zeit. Im vierten Schritt soll der Benutzer Karten zuordnen. So könnten beispielsweise Inhalte einer Seite von den Benutzern thematisch gruppiert werden. Mit Hilfe des letzten Moduls kann das Design einer Seite überprüft werden. So kann sich der Befragte zu bestimmten Bereichen einer Seite äußern. Für die Erstellung dieses Befragungsteils muss ein Bild der entsprechenden Seite in das System geladen werden. Bei der Befragung kann der Benutzer bestimmte Teile des Bildes markieren.

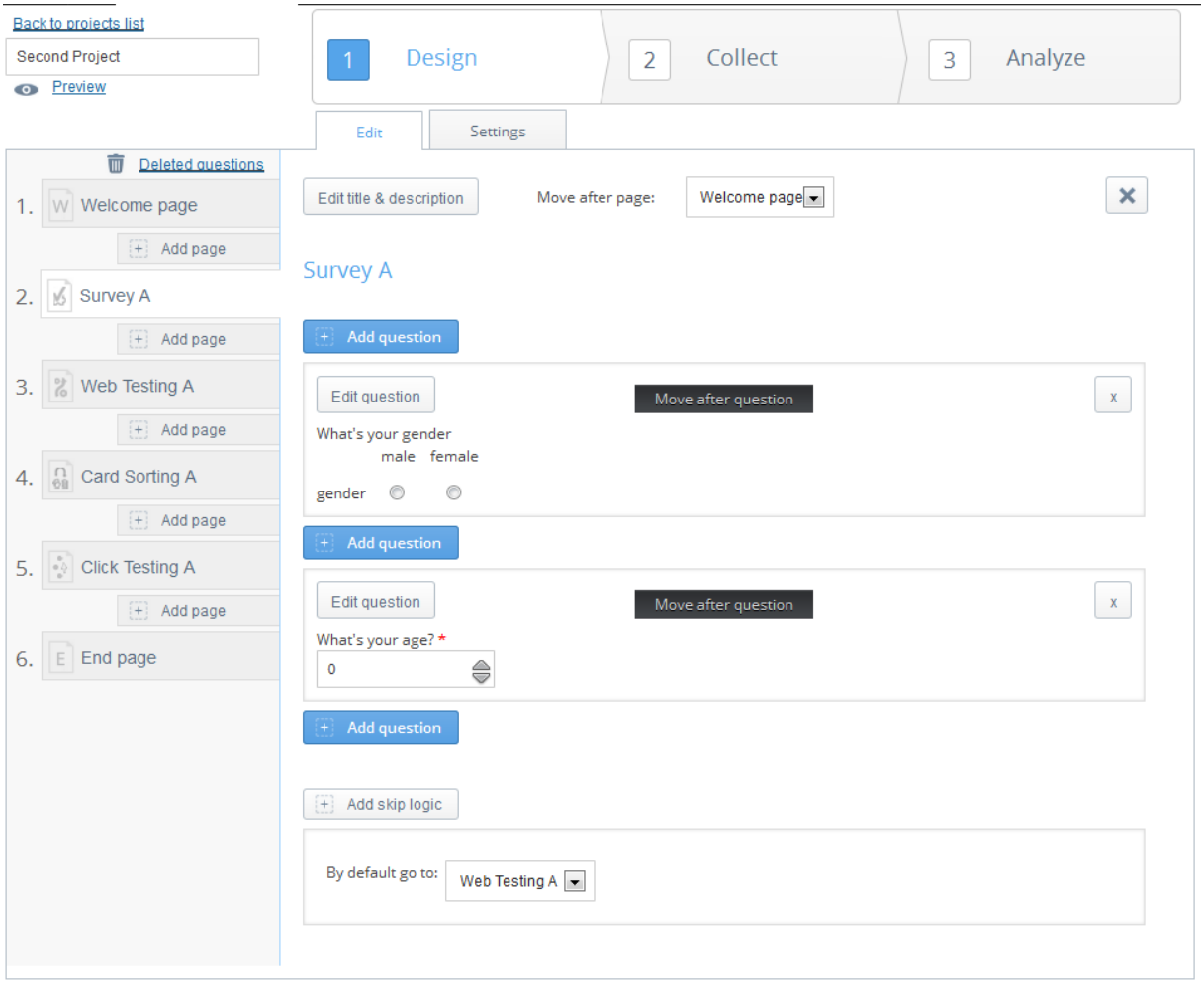


Abbildung 39: Screenshot einer Benutzerbefragung aus der User Experience Suite

Ist die Erstellung einer Befragung fertiggestellt, wird eine URL generiert die dann an die Probanden verschickt werden muss, damit sie an der Befragung teilnehmen können. Die Ergebnisse der Befragung werden dann in übersichtlichen Tabellen und Tortendiagrammen für den Tester aufbereitet.

2.5 Fazit

Die in diesem Kapitel vorgestellten Prozesse, Methoden und Ansätze geben einen orientierenden Eindruck der unterschiedlichen Ansätze aus dem Bereich UE.

Aus dem *Human-centered Design-Prozess* der Norm ISO 9241-210:2011 konnte das generelle Vorgehen für UE abgeleitet werden. Die in der Norm aufgeführten Grundsätze sollten auf jeden Fall übernommen werden. Somit sollten folgende Punkte aus der ISO Norm für alle Prozesse berücksichtigt werden und übertragen werden:

- Das Design sollte auf einem expliziten Verständnis und der Dokumentation des Benutzers, der Aufgaben und des Kontextes beruhen.
- Die Benutzer und Kunden sollten in den Design- und Entwicklungsprozess involviert werden.

- Das Design wird durch benutzerzentrierte Evaluationen verfeinert und vorangetrieben.
- Der Prozess ist iterativ.
- Das Design berücksichtigt die User Experience.
- Das Design-Team verfügt über multidisziplinäre Kompetenzen und Perspektiven.

Die Norm ist als Rahmenwerk für menschenzentrierte Systementwicklungen zu verstehen. Es beschreibt die dabei zu berücksichtigen Kernaktivitäten. Es fehlen jedoch konkrete Methodenansätze für die Durchführung der verschiedenen Analysen oder die Involvierung von Kunden und Benutzern. Auch die Verweise auf entsprechende Normen sind in diesem Punkt wenig aufschlussreich.

Jacob Nielsen geht in dem von ihm beschriebenem *Usability Engineering Lifecycle* auf konkrete Methoden ein, die sich in der realen Praxis anwenden lassen. Ein von ihm sehr hervorgehobener Aspekt, der in seinem Buch auch mehrfach erwähnt wird, ist das Verständnis des UE als *Ingenieurdisziplin*. UE besteht seiner Meinung nach aus unterschiedlichen Komponenten, die im besten Fall während des gesamten Software-Entwicklungsprozesses angewandt werden sollten. Wenigstens jedoch in den frühen Phasen, zur umfassenden Ermittlung der Anforderungen. Erfolge müssen durch *Usability Attribute* überprüft werden. Erst sie ermöglichen eine systematische Verbesserung des Produktes. Genau wie im Human-centered Design-Prozess schlägt Nielsen zwei Formen der Überprüfung vor. Je nach Reifegrad des Produktes können diese mit oder ohne Benutzerbeteiligung durchgeführt werden. Für zeitkritische Projekte schlägt Nielsen das sogenannte *Discount UE* vor. Dieses beschreibt eine Minimalanforderung von UE-Methoden, die immer noch zu guten Ergebnissen führen.

Im *Contextual Design* von Beyer und Holtzblatt werden eine ganze Reihe von grafischen Modellen aufgezeigt, mit denen kontextrelevante Informationen des Kunden aufgenommen werden können. Die Art und Weise wie aus den Modellen des aktuellen Ist-Zustands durch Konsolidierung und Visionssitzungen erste Prototypen abgeleitet werden können, ist in der Praxis zwar sehr aufwändig, jedoch gut durchdacht. Im Unterschied zum Usability Engineering Lifecycle von Nielsen propagieren Beyer und Holtzblatt vor dem Design der Anwendung, die Überarbeitung der generellen Arbeitsweise. Erst sollte das Problem gelöst werden, bevor eine Software dafür entwickelt wird.

Ein überwiegend auf der Verwendung von Szenarien beruhendes Verfahren des UE wird durch das *Scenario-based Design* von Rosson und Carroll beschrieben. Das Verfahren durchläuft, ähnlich wie auch das Contextual Design, die Phasen der Beschreibung des Ist-Zustands (Problemszenarien), des Designs und der anschließenden Entwicklung von Prototypen inklusive Evaluation. Eine weitere Gemeinsamkeit ist die Beschreibung einer neuen Arbeitsweise (Aktivitätsszenarien), die im ersten Schritt unabhängig von der Systemkonzeption sein sollte. In einer empirischen Studie (Tawbi, Velez, Souveyet & Achour, 2000) konnte gezeigt werden, dass sich Szenarien gut für die Ermittlung von Anforderungen eignen. Darüber hinaus können nach Obendorf und Finck (2008) Szenarien auch als gute Kommunikationsmittel zwischen Softwareherstellern und Kunde eingesetzt werden. Dies konnte bereits in gewerblichen Feldstudien unter Beweis gestellt werden (Obendorf & Finck, 2008). Als in der Literatur häufig vernachlässigtes Element des Szenario-based-Designs erscheinen die angesprochenen „Claims“ zu sein. Diese werden häufig mit Anforderungen aus dem Software-Engineering gleichgesetzt, wodurch der Prozess der iterativen Verfeinerung gestört werden kann.

Das von Donald Norman als *Activity-centered Design* bezeichnete Verfahren unterscheidet sich vom Prozess zur Gestaltung gebrauchstauglicher interaktiver Systeme (ISO 9241-210:2011) lediglich in der Betrachtungsweise. Nicht einzelne Benutzer sollten Gegenstand der Analyse im UE sein, sondern die von ihnen ausgeführten Aktivitäten. Das ACD kann nach Norman also als eine weitere Verfeinerung des HCDs verstanden werden.

Ähnlich wie bei dem ACD liegt beim *Usage-centered Design* von Constantine und Lockwood der Schwerpunkt der Untersuchung bei den auszuführenden Aktivitäten der Benutzer. Der dabei beschriebene UE-Prozess basiert auf einer Mischung von grafischen Modellen und textuellen Beschreibungen, wobei die erhobenen Informationen semantisch miteinander verknüpft sein sollten. Speziell die „Context Map“ verdeutlicht die Unterteilung von Benutzern und deren eingenommenen Rollen. Hiernach nimmt ein Benutzer in der Regel verschiedene Rollen ein. Umgekehrt kann ebenso eine Rolle von mehreren Benutzern eingenommen werden. Auch in Bezug auf die Darstellung von Aktivitäten, Aufgaben und Aktionen (siehe Abbildung 9) verhilft die Activity-Task-Map von Constantin und Lockwood zu einem ersten praktischen Ansatz. Diese Unterteilung von Aktivitäten ist identisch

zu der von Donald Norman. Eine weitere nützliche Ergänzung von Constantine und Lockwood beschreibt die Methode aus Szenarien sogenannte Essential Use Cases zu entwickeln. Durch das Hinterfragen jedes einzelnen Interaktionsschrittes und durch die Frage nach dem „Warum“ beschreibt diese neue Form des Szenarios eine Abfolge von Intentionen der Benutzer. Dies ermöglicht eine elegante Abstraktion der Problematik wodurch mehr Spielraum für innovative Lösungsansätze gegeben ist.

Zusammenfassend kann der Prozess zur Gestaltung interaktiver Systeme aus der ISO 9241-210:2011 als grobe Prozessbeschreibung verwendet werden. Die zu durchlaufenden Prozessschritte sollten wie in Abbildung 40 mit entsprechenden Methoden ergänzt werden. Die durch die jeweiligen Methoden erstellten Informationen müssen in geeigneter Weise in Form eines Arbeitsdokumentes, wie in der ISO 9241-210:2011 beschrieben, dokumentiert werden. Das dabei entstehende Arbeitsdokument sollte iterativ während der gesamten Entwicklung verfeinert werden. Dabei ist es wichtig, dass dieses Dokument während des gesamten Prozesses von allen Stakeholdern eingesehen werden kann.

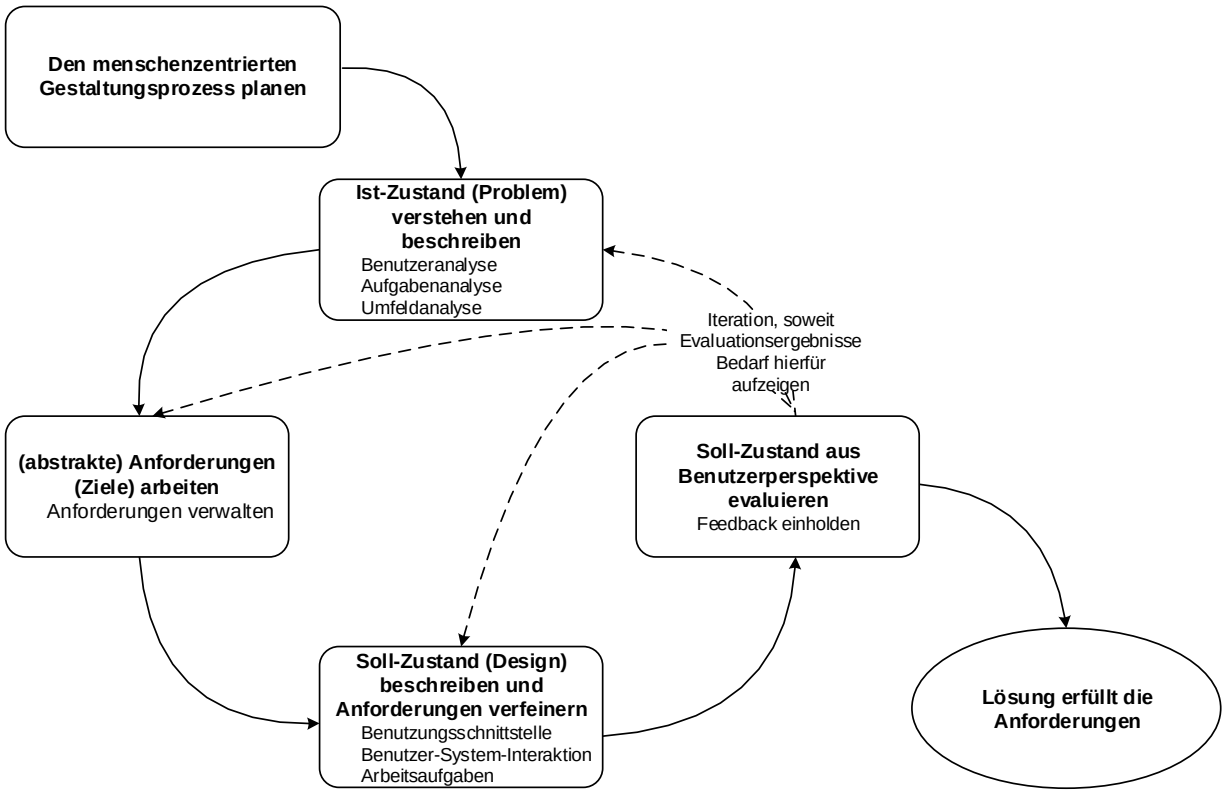


Abbildung 40: Um Methoden erweiterter HCD-Prozess

3 Software-Engineering

Genau wie das UE beschäftigt sich das Software-Engineering (SE) mit der Entwicklung von Software-Systemen. Zu diesem Zweck haben sich in beiden Bereichen unterschiedliche Vorgehensmodelle entwickelt. Einige aus dem Bereich des UEs wurden bereits in Kapitel 2 vorgestellt. SE-Prozesse werden von Unternehmen benötigt, um die Herstellung eines Produktes systematisch planen und koordinieren zu können. UE-Prozesse verbessern nachhaltig die Gebrauchstauglichkeit eines interaktiven Systems als auch die Akzeptanz bei den Benutzern. Bis heute mangelt es an Konzepten und Werkzeugen wie genau die Methoden und Verfahren des UE in etablierte Software-Entwicklungsprozesse integriert werden können, bzw. miteinander kombiniert werden sollten. Da das UE nur gemeinsam mit dem SE praktiziert werden kann, sollen die nachfolgenden Kapitel einen groben Überblick vergangener und aktueller SE-Prozesse geben, um im Anschluss notwendige Gemeinsamkeiten ausarbeiten zu können. Zu diesem Zweck müssen im Vorwege einige spezifische Eigenschaften, Begrifflichkeiten und Verfahren aus diesem Bereich analysiert werden. So gilt es herauszufinden, wie aktuelle Vorgehensmodelle in der Softwaretechnik aufgebaut sind, sowie welche Arbeitsschritte von wem zu welchen Ergebnissen führen (siehe Kapitel 3.1. bis 3.1.5). Auch Ansätze aus dem SE, die sich bereits jetzt verschiedener Methoden aus dem UE bedienen, werden in Kapitel 3.3 näher betrachtet.

3.1 Prozesse

Auch die Prozessmodelle des Software-Engineering sollen, wie im UE einen einheitlichen Rahmen und eine kontrollierbare Vorgehensweise bei der Softwareentwicklung gewährleisten. Sie unterteilen den Entwicklungsprozess in definierte Phasen und geben so vor, wann eine Rolle durch eine bestimmte Aktivität welches Arbeitsergebnis erreichen soll. Ein Prozessmodell soll darüber hinaus gewährleisten, dass ein Softwaresystem zu einem bestimmten Zeitpunkt mit definierten Qualitätsanforderungen und einem vorher kalkulierten Preis entwickelt werden kann. Die große Anzahl unterschiedlicher Prozessmodelle deutet darauf hin, dass derzeit kein ultimatives Vorgehensmodell existiert, welches den Ansprüchen aller softwareentwickelnden Unternehmen genügen kann.

3.1.1 Wasserfall

Im Jahre 1970 stellte Winston Royce das erste Mal ein siebenstufiges Prozessmodell vor, welches seinen Projekterfahrungen nach für größere Softwareprojekte geeignet sei (Royce, 1970). Dieses Prozessmodell wurde später unter dem Namen *Wasserfallmodell* oder *Phasenmodell* bekannt. Royce selbst bezeichnete es als Entwicklungszyklus. Die Sortierung der einzelnen Schritte beruht auf der Idee, dass jeder Schritt die Entwicklung vorantreibt und das Design immer weiter detailliert. Das Prozessmodell berücksichtigte, im Gegensatz zu vorhergehenden Überlegungen, dass ein realer Entwicklungsprozess in den seltensten Fällen linear von der Anforderungsaufnahme bis zur Auslieferung durchlaufen werden kann. In jeder Entwicklung existieren unvorhersehbare Designschwierigkeiten. Somit besitzt das Wasserfallmodell bei jedem Schritt die Möglichkeit zum vorhergehenden Schritt zu springen.

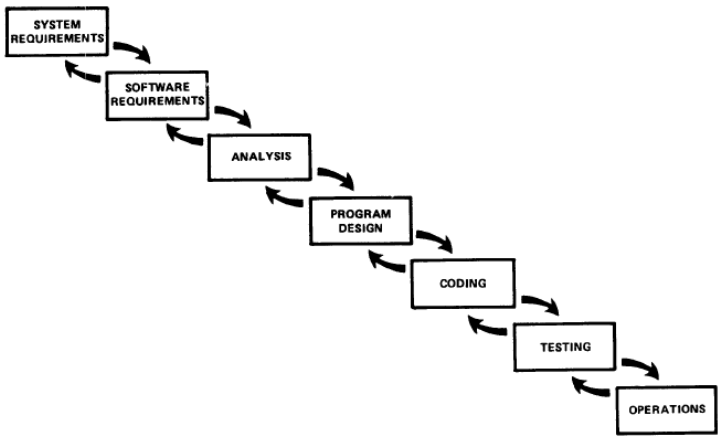


Abbildung 41: Wasserfallmodell nach Royce (Royce, 1970)

Schon bei Einführung hatte Royce eine kleine Schwäche in dem Modell erkannt und auch diskutiert. Das erste Mal, dass Ergebnisse der Systemumsetzung überprüft werden, ist hiernach in der Testphase. Viele Fehler lassen sich erst dann erkennen. Es kann dann der Fall eintreten, dass die Fehler nicht durch isolierte Codeverbesserungen korrigiert werden können, sondern erst durch eine komplette Neugestaltung des Programmdesigns. Das führe dann auch zu neuen Anforderungen. Über den Rücksprung eines neuen Programmdesigns müssen so auch neue Anforderungen formuliert werden, womit der Prozess eigentlich einen Zyklus beschreibt.

Eine weitere Optimierung des Prozesses sieht Royce in der Entwicklung eines vorläufigen Programmdesigns. Heute würden wir wahrscheinlich von einem Prototypen sprechen. Dieser solle zeitgleich zu der Phase der Analyse entworfen und ausführlich dokumentiert werden. Bevor mit der Implementierung begonnen wird, sollte anhand der Dokumentationen so häufig getestet werden, bis es keine Fehler mehr gibt. (Royce, 1970)

3.1.2 V-Modell

Das *V-Modell* wurde 1979 das erste Mal von Barry Boehm vorgestellt. Der Name stammt vermutlich aus der V-förmigen Darstellung des Modells (Boehm, 1979). Boehm selbst wollte in dem Modell den Unterschied und die Wichtigkeit von Validierung und Verifikation (V&V) in der Softwareentwicklung verdeutlichen. Das V-Modell kann als Erweiterung des Wasserfallmodells angesehen werden. Auf den beiden Flanken des Modells (siehe Abbildung 42) ist der sequentielle Aufbau des Wasserfallmodells wiederzuerkennen. Für jede dieser Phasen beschreibt die rechte Seite des Modells eine korrelierende prüfende Phase. Nach Grechenig ist dieses Teststufenkonzept, in ähnlicher Form, Grundlage für viele der heutigen Teststrategien (Grechenig et al., 2010).

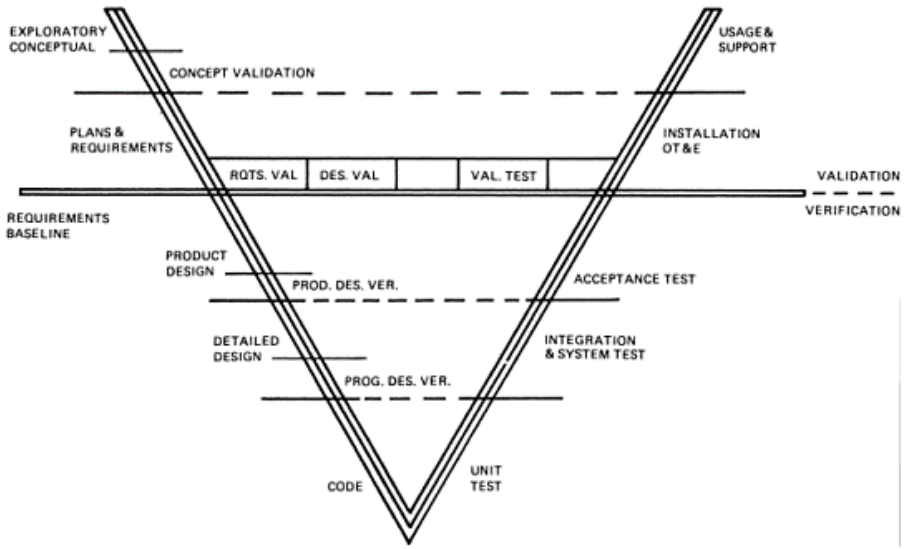


Abbildung 42: Darstellung des klassischen V-Modells (Boehm, 1979)

Der primär von Boehm diskutierte Aspekt wird in dem Modell durch die horizontale Unterteilung des Prozesses deutlich. Phasen oberhalb dieser Unterteilung müssen während eines Entwicklungsprozesses validiert werden. Alle darunterliegenden Phasen werden verifiziert. Konkret bedeutet dies, dass in den oberen Prozessschritten überprüft wird, ob das richtige Produkt gebaut wird. In den darunterliegenden wird getestet, ob das Produkt richtig gebaut wird. Das Modell ist Grundlage vieler heutiger Prozessmodelle wie dem V-Modell 97 oder dem Nachfolger V-Modell XT. Das XT in der aktuellsten Version steht für „Extreme Tailoring“ und soll auf die bessere Modularisierbarkeit sowie eine bessere Unterstützung von agilen und inkrementellen Ansätzen hindeuten (Grande, 2011).

3.1.3 Spiralmodell

1988 stellte Barry Boehm das *Spiralmodell* vor (Boehm, 1988). Der Hauptunterschied zu vorhergehenden Modellen sei ein „Risiko-getriebener Ansatz“ im Gegensatz zu den bisherigen „Dokumenten- oder Code-getriebenen Ansätzen“. In diesen war es wichtig, die Übergangskriterien von einer Phase in die nächste genau festzulegen. Das Spiralmodell basiert auf jahrelangen Erfahrungen verschiedener Verfeinerungen des Wasserfallmodells. Die radiale Dimension (siehe Abbildung 43) des Modells repräsentiert die kumulativen Kosten bei der Entwicklung.

Der Prozess ist in vier Hauptphasen unterteilt. Bei jedem Zyklus bzw. jeder Iteration werden immer wieder alle vier Phasen durchlaufen.

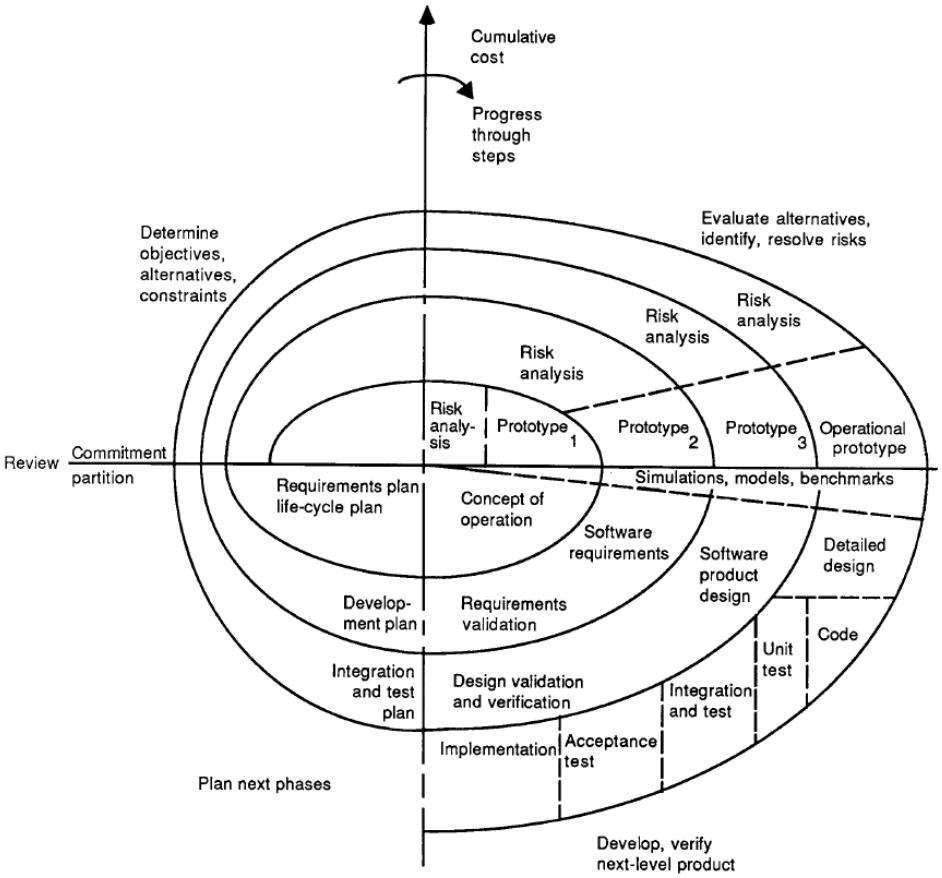


Abbildung 43: Spiralmodell (Boehm, 1988)

Jeder Durchlauf beginnt mit der Bestimmung der *Ziele*, möglichen *Einschränkungen* (z.B. Zeit, Kosten oder technischen Machbarkeiten) und den sich aus den Einschränkungen ergebenen *Alternativen* der jeweiligen Produktiteration, an der gearbeitet werden soll. Im nächsten Schritt werden die Alternativen relativ zu den Zielen und Einschränkungen hin bewertet. Häufig deckt dieser Schritt Unsicherheiten und mögliche Projektrisiken auf. Wenn dem so ist, sollten effiziente Lösungsstrategien dokumentiert werden und wie diese gelöst werden können. Die Bewertung der Risiken bestimmt den nächsten Schritt. Abhängig von dem Risiko und den jeweiligen Lösungsstrategien (z.B. Simulationen, Modelle, Leistungstests oder Benutzerbefragungen) sollte der nächste Schritt ein möglichst kleines Restrisiko beinhalten. In der dritten Phase wird die jeweilige Lösungsstrategie durchgeführt und führt zu einer neuen evolutionären Produktiteration. Ausgehend von der Begutachtung der jeweiligen Produktergebnisse wird dann die nächste Phase geplant. Dieser Vorgang wird so häufig wiederholt, bis keine Risiken mehr existieren, denn dann ist das Produkt fertig.

3.1.4 Rational Unified Process (RUP)

Der *Rational Unified Prozess (RUP)* ist ein etablierter Software-Entwicklungsprozess der Firma IBM Rational. Interessant für die Zusammenführung der beiden Prozesse „Software-Engineering“ und „UE“ ist die Tatsache, dass wesentliche Methoden des UEs explizit verwendet bzw. vorgeschlagen werden. Nach Richter und Flückiger (2010) werden in der Anforderungsanalyse des Prozesses explizit die Bedürfnisse der Benutzer erhoben. Die Modellierung der Anforderungen geschieht aus Benutzersicht und fließt so in die Spezifikation der Entwicklung. Der Prozess ist iterativ und erlaubt somit die Überprüfung und Verfeinerung von Ergebnissen. Durch die Verwendung von Storyboards zur Darstellung von Dialogabläufen können die Benutzer zudem in den Prozess involviert werden.

Wesentliche Kernkonzepte des „Usage-centered Designs“ aus Kapitel 2.2.6 seien nach Constantine mit in den Rational Unified Prozess eingeflossen (Constantine, Biddle & Noble, 2003). Darüber hinaus wird eine komponentenbasierte Architektur propagiert, welche auch die Verwaltung komplexer Anwendungen ermöglicht (Essigkrug & Mey, 2007). Wie von Balzert gefordert (siehe Kapitel 3.1.2), wird auch im RUP der Verwaltung und dem Management (Nachverfolgen, Überwachen und Ändern) von Anforderungen eine wichtige Rolle beigemessen. Das Besondere an diesem Prozess ist die Aufteilung der beiden Sichtweisen in die *Managementsicht* und die der *Entwicklungssicht* (siehe Abbildung 44). In der vertikalen *Entwicklungssicht* ist eine Auflistung der einzelnen Tätigkeiten angeführt, die durchzuführen sind. Diese besitzen jedoch keinen festen Zeitbezug, sondern beschreiben lediglich eine logische Gruppierung. Diese werden anders als bei einem klassischen Wasserfallmodell in jeder Iteration einmal komplett durchlaufen, abhängig von der jeweiligen Phase des Projektes in unterschiedlich starker Intensität (Zuser, Grechenig & Köhle, 2004).

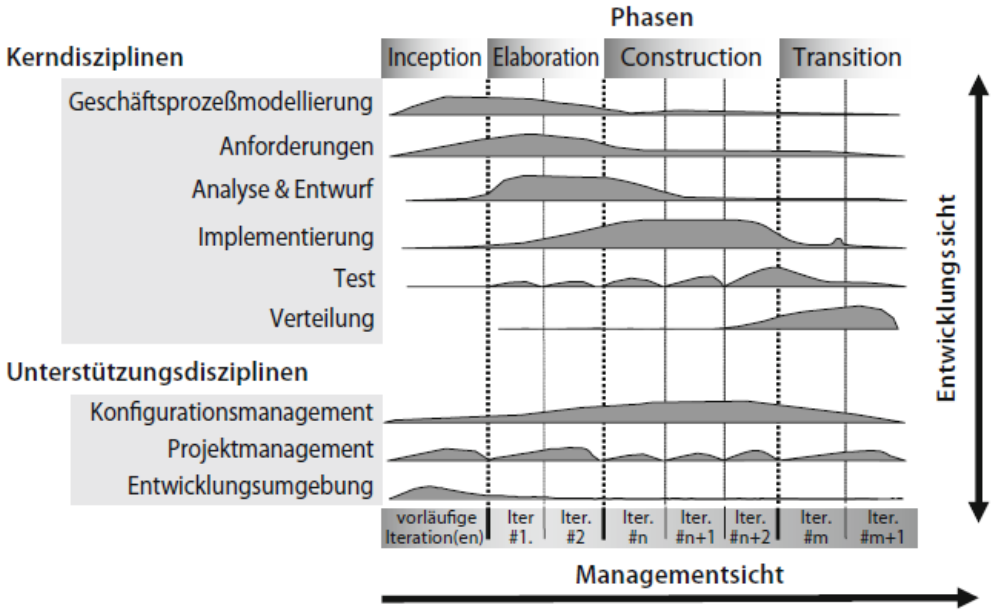


Abbildung 44: Aufteilung des RUP-Prozesses (Hruschka, Rupp & Starke, 2009, S. 78)

In der horizontalen *Managementsicht* werden die für das Projekt relevanten planungstechnischen Aspekte beschrieben. Hier werden Aussagen über die einzelnen Phasen, Iterationen und Meilensteine von der Vorbereitungsphase (Inception) bis zur Auslieferung beim Kunden (Transition) getroffen. In der Unterstützungsdisziplin *Projektmanagement* werden beispielsweise die einzelnen Iterationen detailliert. Zwei verschiedene Detailstufen beschreiben zum einen die Phasenplanung mit den jeweiligen Meilensteinen, zum anderen wird für jede Iteration ein neuer Iterationsplan aufgestellt, der sich mit der Feinplanung auseinandersetzt. Die Iterationsplanung als auch die Phasenplanung kann beispielsweise in Form von GANTT[13]-Diagrammen verwaltet werden (Gantt, 1910). Bei der *Geschäftsprozessmodellierung* geht es darum, die Struktur und Arbeitsweise der Organisation, für die das Produkt entwickelt werden soll, zu verstehen. Nach Zuser sollte dabei die zukünftige Struktur der Organisation beschrieben werden. Auf dieser lassen sich dann alle Prozesse, Rollen und Verantwortlichkeiten abbilden (Zuser et al., 2004). Die Verwendung von *Anforderungen* soll eine gemeinsame und einheitliche Sichtweise von Systementwicklern und Stakeholdern über das System ermöglichen. Neben der Unterteilung in mögliche Anwendungsfälle sollen auch verschiedene Kategorien wie funktionale und nichtfunktionale Anforderungen möglich sein. Als *Analyse und Entwurf* wird die Überführung der Anforderungen in einen Entwurf bezeichnet. Dabei werden vor allem für den Entwickler verständliche Beschreibungssprachen wie beispielsweise Klassendiagramme verwendet. In der Implementierung soll das eigentliche und ausführbare System entwickelt werden. RUP sieht auch die Implementierung von Prototypen zur Überprüfung von Machbarkeiten und Demonstrationszwecken vor. Diese soll-

[13] Balkenplandarstellung nach Henry L. Gantt

ten aus Qualitätsgründen jedoch nicht in das tatsächliche System integriert werden. Nach Zuser besitzt der Arbeitsschritt des *Testens* im RUP einen sehr hohen Stellenwert. Dabei werden Testfälle, Testprozesse, Testskripte sowie Testklassen eingesetzt. Darüber hinaus wird von allen Entwicklern ein eigenständiger Qualitätsanspruch ihrer eigenen Arbeit erwartet. Das *Konfigurationsmanagement* verantwortet die Versionierung und Identifizierung. Auch die Abhängigkeiten der Anforderungen untereinander sind zu managen. Mit dem Punkt Entwicklungsumgebung wird die Auswahl und Installation der entsprechenden Frameworks, sowohl für die Entwicklung selbst als auch die Verwaltung der einzelnen Branches und Backups bezeichnet. Die *Auslieferung* bzw. *Verteilung* des Systems beim Kunden wird bei der Entwicklung häufig unterschätzt. Dieser sieht neben der Installation des Systems, der Migration bestehender Daten des Kunden auch dessen Schulung auf das neue System vor. (Zuser et al., 2004)

3.1.5 Agile Prozesse und Scrum

Die Bewegung hin zu den *Agilen Prozessen* kam durch Kent Beck (Beck, 2000) und Ward Cunningham im Jahre 2000 auf. Diese prägten ein neues Leitbild für die Softwareentwicklung, durch provozierende Aussagen in Form des „Agile Manifests“. Aus diesem stammen die folgenden vier Leitsätze (Beck & Grenning, 2001).

- Individuen und Interaktionen mehr als Prozesse und Werkzeuge
- Funktionierende Software mehr als umfassende Dokumentation
- Zusammenarbeit mit dem Kunden mehr als Vertragsverhandlung
- Reagieren auf Veränderung mehr als das Befolgen eines Plans

Die Liste sollte so verstanden werden, dass die linke Seite einen höheren Stellenwert als die Rechte hat und somit beispielsweise „umfassende Dokumentationen in der Agilen Softwareentwicklung nicht ausgeschlossen werden sollen. Grund für diese Bewegung waren die immer komplexer gewordenen Vorgehensmodelle zur Softwareentwicklung (Hruschka et al., 2009), wie beispielsweise das V-Modell. Agile Prozesse zeichnen sich dadurch aus, schlank zu sein, flexibel auf Änderungen reagieren zu können sowie iterativ und inkrementell vorzugehen. Bemängelt werden könnte, dass der Scrum-Prozess kaum belastbare Referenzdokumente und ausgereifte Entwürfe vorsieht, die über die Lebensdauer der Software gepflegt werden. Die Tatsache, dass große Unternehmen wie Microsoft und Google (Sutherland, 2010, S. 7) in der Softwareentwicklung auf Scrum umgestiegen sind, unterstreicht die Bedeutung agiler Prozesse und im Besonderen die von Scrum. Aus diesem Grund wird das nächste Kapitel sich mit dem Thema „Scrum“ beschäftigen.

3.1.5.1 Der Aufbau von Scrum

Scrum beschreibt ein Prozess-Rahmenwerk, welches dazu verwendet werden kann, komplexe Softwarelösungen zu managen (Sutherland & Schwaber, 2011). Es kann als eine spezielle Ausbildung der agilen Softwareentwicklung angesehen werden. Scrum ist aus mehreren Gründen als Softwareentwicklungsframework in Verbindung mit UE interessant. Zum einen orientiert es sich genau wie die ISO 9241-210:2011 nach einem iterativen Vorgehen, zum anderen gibt es bereits einige Ansätze, Scrum mit Usability Methoden zu verbinden. Einer zwei Jahre andauernden industriellen Fallstudie nach, soll die Umstellung auf Scrum die Anzahl der Überstunden bei den Entwicklern verringert haben, die Qualität der Softwarelösungen und die Kundenzufriedenheit gestiegen sein (Mann & Maurer, 2005). Bevor einige Ansätze zur Verschmelzung von UCD und Scrum eingehender beleuchtet werden, soll im Folgenden ein kurzer Überblick über das empfohlene Vorgehen und die dabei benötigten Komponenten gegeben werden. Ein Großteil der nachfolgenden Informationen stammen aus dem „Scrum Guide“ der beiden Mitbegründer Jeff Sutherland und Ken Schwaber (Sutherland & Schwaber, 2011).

Das Scrum Team

Ein Scrum Team besteht aus drei verschiedenen Rollen. Einem „Product-Owner“, der die Vertretung des Kunden oder Endanwenders übernimmt und als Einziger für die Verwaltung des „Product-Backlog“ verantwortlich ist, dem „Entwicklungs-Team“, welches für die Umsetzung sorgt und dem „Scrum Master“, der die Rolle eines „Methodenfachmannes“ einnimmt und dafür sorgt, dass der Entwicklungsprozess nicht zerfällt. Das Scrum Team sollte autonom arbeiten können, d.h. es verfügt über alle Kompetenzen, die es benötigt, um seine Arbeitsziele zu erreichen.

Tabelle 3: Scrum Rollen

Rolle	Aufgaben
Product-Owner	Verwaltung des „Product-Backlogs“ • Formuliert die Einträge • Priorisiert die Einträge • Sorgt für das Verständnis aller Einträge bei allen Beteiligten • Stellt die Einsehbarkeit für alle Beteiligten sicher
Entwicklungs-Team	Erstellung eines verwendbaren (potentiell auslieferbares) Produktinkrements zum Ende eines Sprints • Organisieren sich selbst • Arbeiten selbstbestimmt das jeweilige „Product-Backlog“ ab
Scrum Master	Verantwortlich für die Einhaltung und das Verständnis der Scrum-Methoden Arbeitet dem Product-Owner zu, indem er • Techniken zur Verwaltung des Product-Backlogs etabliert • Ziele, Visionen und Einträge des Product-Backlogs dem Entwicklungsteam kommuniziert • überprüft die Eintragungen des Scrum Teams in das Product-Backlog

Der Scrum Prozess

Herzstück des Scrum Prozesses ist der Sprint, der maximal innerhalb eines Monats abgeschlossen sein sollte. Als Ergebnis dieses Sprints sollte immer ein potentiell auslieferbares Produktinkrement entstehen. Ausgehend vom Product-Owner, der die Verantwortung für das Resultat der Arbeit des Entwicklerteams übernimmt, hat er auch dafür Sorge zu tragen das Produktbacklog zu verwalten und allen zugänglich zu machen. In dem Product-Backlog befinden sich alle Anforderungen und Änderungen, die für das Produkt benötigt werden, in Form einer geordneten Liste.

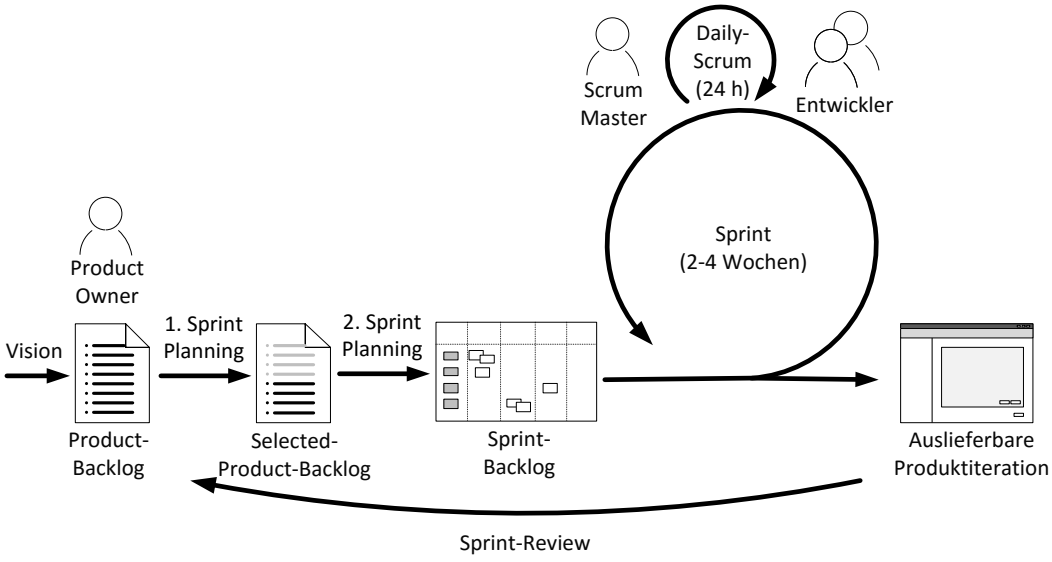


Abbildung 45: Scrum-Prozess angelehnt an Schwaber (2004) und Hanser (2010)

Die Backlog-Liste kann nach verschiedenen Attributen sortiert werden. Mögliche Kriterien könnten dabei „Wert Risiko, Priorität und Notwendigkeit“ (Sutherland & Schwaber, 2011) sein. Auf diese Weise befinden sich in der Backlog-Liste zu jeder Zeit alle das gesamte Produkt angehende Anforderungen, Änderungen, Features und Fehlerverbesserungen. Je höher sich die Einträge in der Backlog-Liste befinden, desto detaillierter und feingranularer

sollten diese sein. Jeder Eintrag, der mit dem Attribut „ready“ gekennzeichnet wurde, was den Reifegrad eines Eintrages kennzeichnet in den nächsten Sprint zu kommen, sollte auch in diesem Zeitfenster umzusetzen sein, andernfalls müsste er noch weiter zerteilt werden. Das „Sprint Meeting“ teilt sich in zwei Teile auf, das „1. Sprint Planning“ und das „2. Sprint Planning“. Es sollte bei einem vier-wöchigen Sprint maximal acht Stunden dauern.

Im ersten Sprint Planning entscheidet das Entwicklerteam gemeinsam, welche bearbeitbaren - mit „ready“ gekennzeichneten - Einträge aus der Backlog Liste in dem kommenden Sprint übernommen werden können. Verständnisfragen zu den Einträgen werden vom Product-Owner beantwortet. Die Entscheidung, welche Einträge in den nächsten Sprint übernommen werden, entscheiden ausschließlich die Entwickler. Anschließend wird gemeinsam ein Sprintziel erarbeitet.

> *„Das Sprint-Ziel ist eine Vorgabe, die innerhalb des Sprints durch die Umsetzung der Einträge des Product Backlogs verwirklicht wird. Das Sprint-Ziel begründet dem Entwicklungs-Team, warum das kommende Produktinkrement erstellt wird.“* (Sutherland & Schwaber, 2011, S. 10)

Die ausgewählten Einträge der Backlog-Liste gemeinsam mit den Sprint-Zielen können jetzt als „Selected-Product-Backlog“ bezeichnet werden. Es handelt sich dabei lediglich um einen neuen Zustand des eigentlichen Product-Backlogs.

Im zweiten Teil des Sprint Meetings erarbeitet das Entwickler Team einen Plan, wie die ausgewählten Einträge und das Sprint Ziel erreicht werden sollen. Praktisch bedeutet dies, die Einträge in kleine Teilaufgaben zu zerteilen. Am Ende des 2. Sprint Meetings sollte das Entwickler Team genügend Teilaufgaben für die ersten Arbeitstage notiert haben und dazu in der Lage sein, dem Product-Owner erläutern zu können, wie sie das Gesamtziel des Sprints erreichen wollen. Die erarbeiteten Ergebnisse werden dann als Sprint Backlog bezeichnet. In der Praxis wird in diesem Arbeitsschritt häufig mit „User Stories“ (siehe Kapitel 3.1.5.2) auf einer Pinnwand gearbeitet. Der Product-Owner muss dem zweiten Teil des Meetings nicht zwingend beiwohnen, es kann jedoch für die Klärung unklarer Teilaspekte hilfreich sein.

3.1.5.2 Erweiterter Prozess

Um Scrum in der Praxis produktiv verwenden zu können, müssen noch einige Punkte näher beleuchtet werden. Gerade die Auswahl dieser Methoden und ihre konsequente Umsetzung, können über den Erfolg oder Misserfolg der Verwendung des Scrum-Rahmenwerkes entscheiden.

Produktkonzept: Von der Vision zum Product-Backlog

Am Anfang eines jeden Projektes steht immer eine Vision, gewissermaßen die Idee für die Anwendung oder das System, welches umgesetzt werden soll. Die erste Aufgabe des Product-Owners ist es in einem sogenannten Produktkonzept diese Vision in verständlicher Form festzuhalten. In dieser textuellen Beschreibung sollte die Kernidee und Basisfunktionalität beschrieben werden.

Um die Ideen möglichst vieler Interessensvertreter in das Product-Backlog einfließen zu lassen, kann es zu Anfang eines Projektes hilfreich sein, einen Anforderungsworkshop durchzuführen. Von Ralf Wirdemann (2011) wird vorgeschlagen, hierzu mit einer Mindmap (siehe Abbildung 46) zu arbeiten, in der ausgehend von den verschiedenen Benutzerrollen alle Inhalte und Ideen notiert werden können.

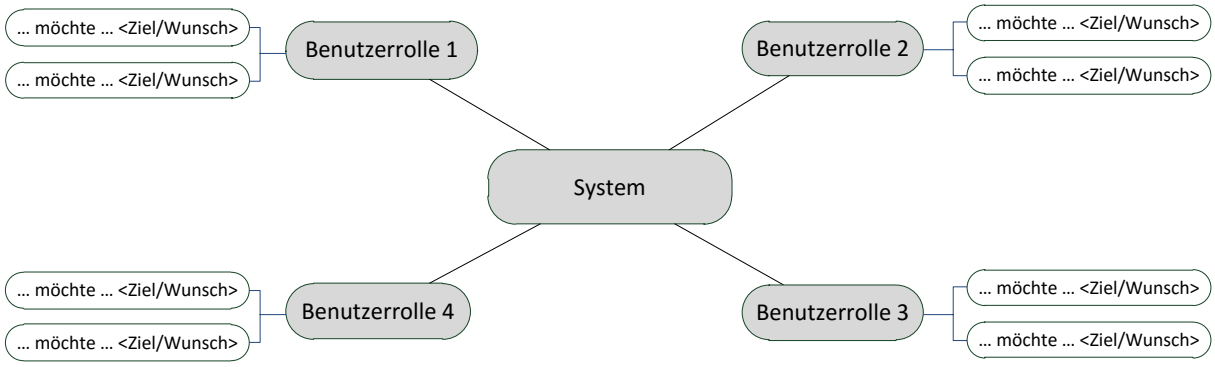


Abbildung 46: Mindmapstruktur eines Anforderungsworkshops

Anschließend ist es die Aufgabe des Product-Owners die erlangten Ideen der Mindmap in das Product-Backlog zu übertragen.

User Stories und Story-Karten

User Stories beschreiben Anforderungen aus Sicht der Benutzer (Wirdemann, 2011). So kann das gesamte Product-Backlog aus einer Liste dieser User Stories bestehen. Für den eigentlichen Sprint werden diese häufig auf einfache Karten sogenannten Story-Karten geschrieben. Diese Karten fungieren im weiteren Verlauf als Platzhalter für Diskussionsnotizen. Von Mike Cohn (2004) stammt der nachfolgende Ansatz einer formalen Spezifikation der User Story:

> „Als <Rolle> möchte ich <Ziel/Wunsch>, um <Nutzen>“

Die Anforderungen in Form von User Stories sollten aus „[…] in der Sprache des Kunden […] einen konkreten und für den Kunden sichtbaren Mehrwert […]“ (Wirdemann, 2011, S. 10) liefern.

Die Evolution des Product-Backlogs

Wie bereits erwähnt beinhaltet das Product-Backlog sämtliche – das Produkt angehende – Anforderungen, Änderungen und Verbesserungen. Häufig werden diese Einträge auch als „Backlog Items“ bezeichnet. In der Regel durchlaufen diese Backlog Items unterschiedliche Abstraktionsgrade von der vagen Vision bis hin zu konkreten Aufgaben, die von den Entwicklern in einem Sprint umgesetzt werden können.

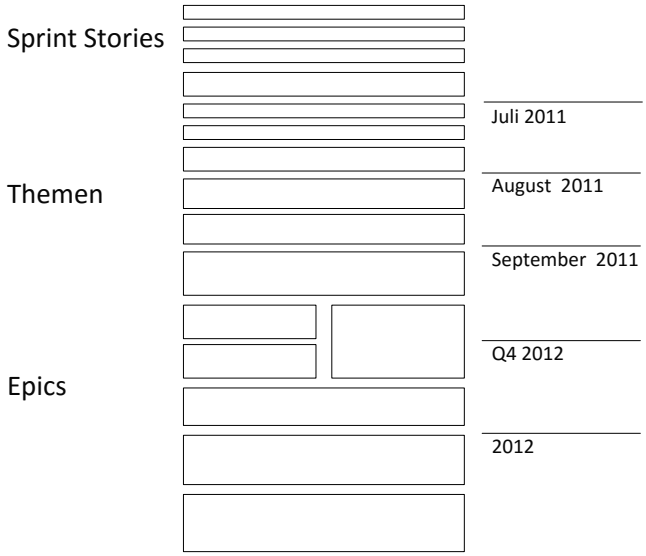


Abbildung 47: Verschiedene Arten von User Stories im Product-Backlog (Wirdemann, 2011, S. 64)

Ausgehend von den Benutzerrollen sollte geklärt werden, was die jeweilige Rolle mit dem System erreichen möchte. Diese Funktionen werden, ohne auf die genauen Details der Umsetzung einzugehen, in das Product-Backlog eingetragen. In der Agilen Softwareentwicklung werden diese grobgranularen Anforderungen auch als „Epics“ bezeichnet. Im weiteren Verlauf müssen diese Epics in „sprintfähige“ User bzw. Sprint Stories aufgeteilt werden. Auf diese Weise sedimentieren die Backlog Items auf iterative Weise von sehr groben Anforderungen zu konkreten User Stories. Dieser Verfeinerungsprozess soll durch Abbildung 47 verdeutlicht werden. In dieser wird ein weiteres Backlog Item verwendet – das sogenannte Thema nach Mike Cohn (2004) – um eine Gruppen von User Stories zusammenzufassen.

3.2 Methoden des SE

Genau wie die Methoden des UE, dienen die Methoden im SE der Lösung bzw. Veranschaulichung wichtiger Teilaspekte in einem Entwicklungsprozess. So können die verschiedenen Prozessmodelle des vorherigen Kapitels zu fünf Teilaktivitäten zusammengefasst werden. Diese werden in ähnlicher Weise und unterschiedlicher Ausprä-

gung von nahezu allen Entwicklungsprozessen durchlaufen. So entwickelte sich beispielsweise in der Analysephase in der es bei der SE primär um die Ermittlung der zu implementierenden Anforderungen geht, das Requirements-Engineering (RE) als eigenständige Disziplin.

Abbildung 48: Grundlegende Schritte des SE

Speziell dieses Gebiet hat thematisch große Überschneidungen mit den Anliegen des UEs. Für eine optimalere Verschmelzung von SE und UE sollen somit im Folgenden die einzelnen Aktivitäten und verwendeten Artefakte des SE betrachtet werden.

3.2.1 Analyse

Nach Balzert (2009) sollte vor der eigentlichen Ermittlung der Anforderungen die jeweiligen Stakeholder identifiziert und einbezogen werden. Es reiche nicht aus, nur mit den Auftraggebern bzw. Geldgebern zu reden. Diese können in einer Tabelle aufgelistet werden. Nützliche Informationen seien neben der Bezeichnung des Stakeholders, deren Erwartungen, ihre Haltung zum Projekt (positiv, neutral oder abgeneigt) und ihrer Macht in dem Unternehmen. Anschließend müsse das Produktumfeld bestimmt werden. Zu diesem gehören alle Elemente, die einen Einfluss auf das Projekt haben können, beispielweise Vorgaben, Normen, Standards und einzuhaltende Gesetze. Weitere wichtige Informationen könnten mit Methoden wie „Brainstorming“ und „Mindmaps“ gesammelt und strukturiert werden. Am Ende sollten alle wichtigen Elemente des Projektumfeldes in Form einer Liste oder Mindmap dokumentiert sein. Anschließend könne mit der Ermittlung der Anforderungen bei den jeweiligen Stakeholdern begonnen werden. Anhand verschiedener *Befragungstechniken* wie einem strukturierten Interview mit Hilfe einer Anforderungsschablone, Fragebögen, Feldbeobachtungen oder der sogenannten Lehrlingsrolle, die vergleichbar mit dem Contextual Inquiry (siehe Kapitel 2.2.3) sind. Alternativ könnten auch *Selbstaufschreibungen* durchgeführt werden, in denen Stakeholder ihren Arbeitsalltag sowie Wünsche selbst aufschreiben, oder es werden moderierte Gruppendiskussionen zur Erhebung der benötigten Informationen durchgeführt. Eine *Anforderungsschablone* für ein strukturiertes Interview setzt sich aus einer Vision sowie einer Liste von Zielen zusammen, die diese verfeinern kann. Ergänzt werden sollten eventuelle Restriktionen organisatorischer oder technischer Art sowie einige weitere Rahmenbedingungen. Zu den *Rahmenbedingungen* gehört der Anwendungsbereich beispielsweise „Textverarbeitung im Büro“, die Zielgruppe des Systems, wie Sekretärin oder Kundenbetreuer. Bei der Zielgruppenbeschreibung seien ebenfalls Informationen über die Vorkenntnisse und Qualifikationen dieser Gruppe wichtig. Weitere Rahmenbedingung sind die Betriebsbedingungen mit den Unterpunkten physikalische Umgebung (z.B. Büroumgebung), Betriebszeit (z.B. Dauereinsatz) und Beobachtungsaufwand (z.B. unbeaufsichtigt oder unter ständiger Beobachtung). Zudem wichtig sind die technischen Rahmenbedingungen wie das eingesetzte Betriebssystem oder die Internetverfügbarkeit ebenso wie die eingesetzte Hardware oder etwaige Entwicklungsschnittstellen. Ergebnisse der Ermittlung sind Notizen, Protokolle und falls von den Stakeholdern gestattet Video- oder Audioaufzeichnungen.

3.2.1.1 Dokumentation

Ähnlich wie bei der Analysephase des UEs ist im SE die Formulierung eines Anforderungsdokumentes ein wichtiger Teilbestand. Diese Phase wird auch häufig als *Spezifikation der Anforderungen* bezeichnet (Balzert, 2000). Denkbar sind Gliederungen ohne feste Regeln, Listen nummerierter Anforderungen oder festgelegte Gliederungsschemata. Im deutschsprachigen Raum werden diese Anforderungsdokumente auch häufig als Lastenheft bezeichnet.

Ein *Lastenheft* beschreibt die „Gesamtheit der Forderungen an die Lieferungen und Leistungen eines Auftragnehmers innerhalb eines (Projekt-) Auftrags (DIN 69901-5)“ (Angermeier, 2013). Im weiteren Prozess eines Entwicklungsprozesses kann das Lastenheft auch zu einem Pflichtenheft weiterentwickelt werden. Somit kann das

Lastenheft auch als „grobes Pflichtenheft“ bezeichnet werden (Balzert, 2000). Hier werden alle Forderungen eines Auftraggebers notiert. Häufig wird es zu Ausschreibungszwecken verwendet um es an mehrere mögliche Auftragnehmer zu versenden. Bei Auslieferung des zu entwickelnden Systems dient das Lastenheft dazu, die Leistungen präzise zu überprüfen.

> *„Im Lastenheft sind die Forderungen aus Anwendersicht einschließlich aller Randbedingungen zu beschreiben. Diese sollten qualifizierbar und prüfbar sein. Im Lastenheft wird definiert, was für eine Aufgabe vorliegt und wofür diese zu lösen sind.“* (Gebhardt, 2012)

Eine mögliche Gliederung für ein solches Lastenheft könnte folgende Komponenten beinhalten (Balzert, 2000, 2009):

1. Vision und Zielbeschreibung
2. Rahmenbedingungen
3. Kontext und Überblick
4. Funktionale Anforderungen
5. Qualitätsanforderungen
6. Produktdaten (optional)
7. Ergänzungen
8. Glossar

Jede Vision und jedes Ziel kann in ein bis zwei Sätzen in eine Liste geschrieben werden. Dabei sollte jede Beschreibung mit einer eindeutigen Kennung versehen werden. In den Rahmenbedingungen wird ausführlich beschrieben, für welchen Anwendungsbereich (z.B. Bürotätigkeit) und Zielgruppe das System vorgesehen ist. Der Kontext beschreibt die physikalische Umgebung und die Betriebszeiten (z.B. Dauereinsatz oder nur gelegentlich). Der Kontext kann entweder in Form einer Liste oder als grafische Übersicht dargestellt werden. Die funktionalen Anforderungen werden als Liste mit den im Folgenden beschriebenen Attributen aufgenommen. Qualitätsanforderungen beschreiben die Zuverlässigkeit, Benutzbarkeit oder Effizienz des Systems. Langfristig zu speichernde Hauptdaten und Umfang können in dem Kapitel Produktdaten festgehalten werden. Sollte es noch weitere spezielle Anforderungen geben, werden diese unter den Punkt Ergänzungen hinzugefügt. Wichtig ist immer ein Glossar mit domänenspezifischen Begriffserklärungen.

Ein weiteres anerkanntes und standardisiertes Gliederungsschema stammt von dem Institute of Electrical and Electronics Engineers (IEEE). Dieses sieht folgende Inhalte vor (IEEE Std 830:1998):

1. Einleitung
   1.1. Anwendungsbereich
   1.2. Definitionen, Akronyme und Abkürzungen
   1.3. Referenzen
   1.4. Übersicht
2. Allgemeine Beschreibung
   2.1. Produkt-Perspektive
   2.2. Produkt-Funktionen
   2.3. Benutzereigenschaften
   2.4. Beschränkungen
   2.5. Voraussetzungen und Abhängigkeiten
3. Spezifische Anforderungen
4. Anhänge
5. Index

Nach der DIN 69905-VDI/VDE 3694 – VDA 6.1, beschreibt das Pflichtenheft das vom Auftragnehmer erarbeitete Realisierungsvorhaben aufgrund der Umsetzung des Lastenhefts.

> *„Das Pflichtenheft enthält das Lastenheft. Im Pflichtenheft werden die Anwendervorgaben detailliert und in einer Erweiterung die Realisierungsforderungen unter Berücksichtigung konkreter Lösungsansätze beschrieben. Im Pflichtenheft wird definiert, wie und wo die Forderungen zu realisieren sind.“* (Gebhardt, 2012)

Bei der Formulierung von Anforderungen sei es nach Balzert wichtig, einige Attribute zu berücksichtigen. Jede *Anforderung* sollte aus folgenden Informationen zusammengesetzt sein, die unter Umständen auch während des Entwicklungsprozesses ausgefüllt werden können (Balzert, 2009, S. 479–480).

- Identifikator
- Kurzbeschreibung
- Anforderungstyp
- Beschreibung
- Anforderungssicht
- Querbezüge zu anderen Anforderungen
- Status über die Reife (Idee oder detailliert)
- Abnahmekriterien bei Erfüllung der Anforderung
- Schlüsselwörter (optional), welche die Anforderung charakterisieren (z.B. Interface oder Persistenz)
- Priorität der Anforderung
- Stabilität (Maß der Wahrscheinlichkeit einer Änderung der Anforderung)
- Kritikalität (Maß über die Sicherheitskritikalität einer Anforderung)
- Entwicklungsrisiko (Maß der Einhaltung die Anforderung umsetzen zu können)
- Aufwand
- Konflikte (optional) bzgl. Bestimmter Sicherheitsaspekte oder der Gebrauchstauglichkeit
- Autor
- Quelle (z.B. Stakeholder oder Gesetz)
- Versionsnummer
- Bearbeitungsstatus

Der IEEE 610.12-1990 Standard definiert eine *Anforderung* als:

> *„(1) A condition or capability needed by a user to solve a problem or objective. (2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents. (3) A documented representation of a condition or capability as in (1) or (2).” (IEEE Std 610.12:1990, S. 62)*

3.2.1.2 Genehmigung

Da alle weiteren Aktivitäten in einem Entwicklungsprozess auf den bisher formulierten Anforderungen gestützt werden, sollte als nächster Schritt eine Analyse durchgeführt werden. Dabei müssen alle Anforderungen bestimmten „Gütekriterien“ genügen. Hierfür sind vor allem formale Kontrollen mittels Checklisten gut geeignet, um nichts zu übersehen (Balzert, 2009). Zudem muss durch eine Validierung überprüft werden, ob alle Anforderungen auch das gewünschte Produkt beschreiben. Da in dieser Phase jedoch noch keine Produktbeschreibung existiert, kann nur kontrolliert werden, ob eine Anforderung zur Verwirklichung einer angegebenen Vision und der dazugehörigen Ziele beiträgt. Für eine Ermittlung der zu erwartenden Entwicklungskosten und der Dauer bis zu einem Auslieferungstermin muss eine Aufwandsabschätzung durchgeführt werden. Diese beruht stark auf vorhergehende Erfahrungen. Häufig entsteht eine große Zahl an Anforderungen, jedoch nicht alle sind gleich wichtig. So gibt es Anforderungen, die so essentiell sein können, dass ohne sie das Produkt nicht durch den Kunden akzeptiert werden kann. Andere Anforderungen werten das Produkt nur auf oder werden vom Kunden nicht erwartet. Aus diesem Grund sollten Anforderungen noch priorisiert werden. Hierzu gibt es verschiedene Modelle. Ein schnelles und leicht zu verwendendes Verfahren ist die MuSCoW-Priorisierung. Bei diesem werden die Anforderungen in die vier Kategorien eingeteilt: „Must have“ für zwingend erforderliche, „Should have“ wichtig, jedoch das System funktioniert auch ohne, „Could have“, werden angegangen wenn alle höher priorisierten Anforderung erledigt wurden, und „Won´t have this time“ für Anforderungen, die nur vorgemerkt sind jedoch aktuell nicht umgesetzt werden (Wirdemann, 2011). In der IEEE 830 wird eine Unterteilung in essentiell, bedingt und optional vorgeschlagen (IEEE Std 830:1998). Anschließend sollte das Spezifikationsdokument durch den Kunden abgenommen werden und kann unter Umständen auch als vertragliche Grundlage dienen.

3.2.2 Design

Ziel dieser Phase ist es, aus den bisher angegebenen Anforderungen eine *fachliche Lösung* zu entwickeln. Die fachliche Lösung, auch Produktmodell genannt, umfasst ein Modell des zu entwickelnden Systems. In diesem sollten alle Anforderungen vollständig überführt werden. In der Software Entwicklung werden in dieser Modellierungsphase Beschreibungsmittel aus der Unified Modeling Language (UML) verwendet.

3.2.2.1.1 Use-Case-Diagramm

Use-Case-Diagramme eignen sich für die Darstellung einzelner Funktionalitäten eines Systems (Jacobson, 1992). Eine Aufgabe wird dabei als Ellipse dargestellt (siehe Abbildung 49). Beziehungen zwischen einzelnen Aufgaben werden mittels Pfeilen und Linien verdeutlicht. Es existieren vier verschiedene Beziehungstypen:

- Assoziation: Kommunikation zwischen Akteur und Use Case.
- Generalisierung: Verallgemeinerung eines Use-Cases.
- Extension: Erweiterung eines Use-Cases.
- Inklusion: Einschließen eines Use-Cases.

Das System bzw. dessen Grenze wird als Kasten gekennzeichnet. Dies ermöglicht die Darstellung der Kommunikation weiterer Systeme untereinander.

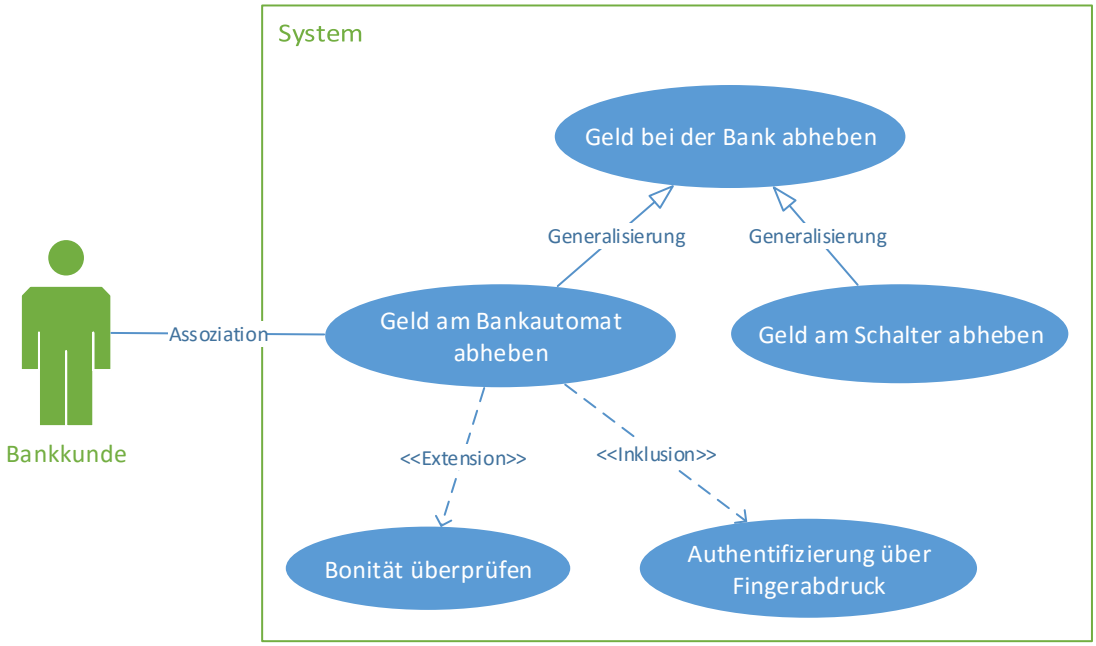


Abbildung 49: Beispiel für ein Use-Case-Diagramm

Use-Case-Diagramme eignen sich jedoch nicht für die Darstellung der Interaktion zwischen System und Benutzer (Pohl, 2008).

3.2.2.1.2 Aktivitätsdiagramm

Für die Darstellung der Interaktion eignen sich vor allem Aktivitätsdiagramme (siehe Abbildung 50). Sie ermöglichen die Darstellung des Kontrollflusses mehrerer Interaktionsfolgen (Szenarien). Es wird die Reihenfolge der einzelnen Aktivitäten verdeutlicht. Entscheidungspunkte, die als Rauten dargestellt werden, ermöglichen die Dokumentation alternativer Interaktionsfolgen. Über Swimlanes kann die Rollenaufteilung zwischen Akteur und System dargestellt werden.

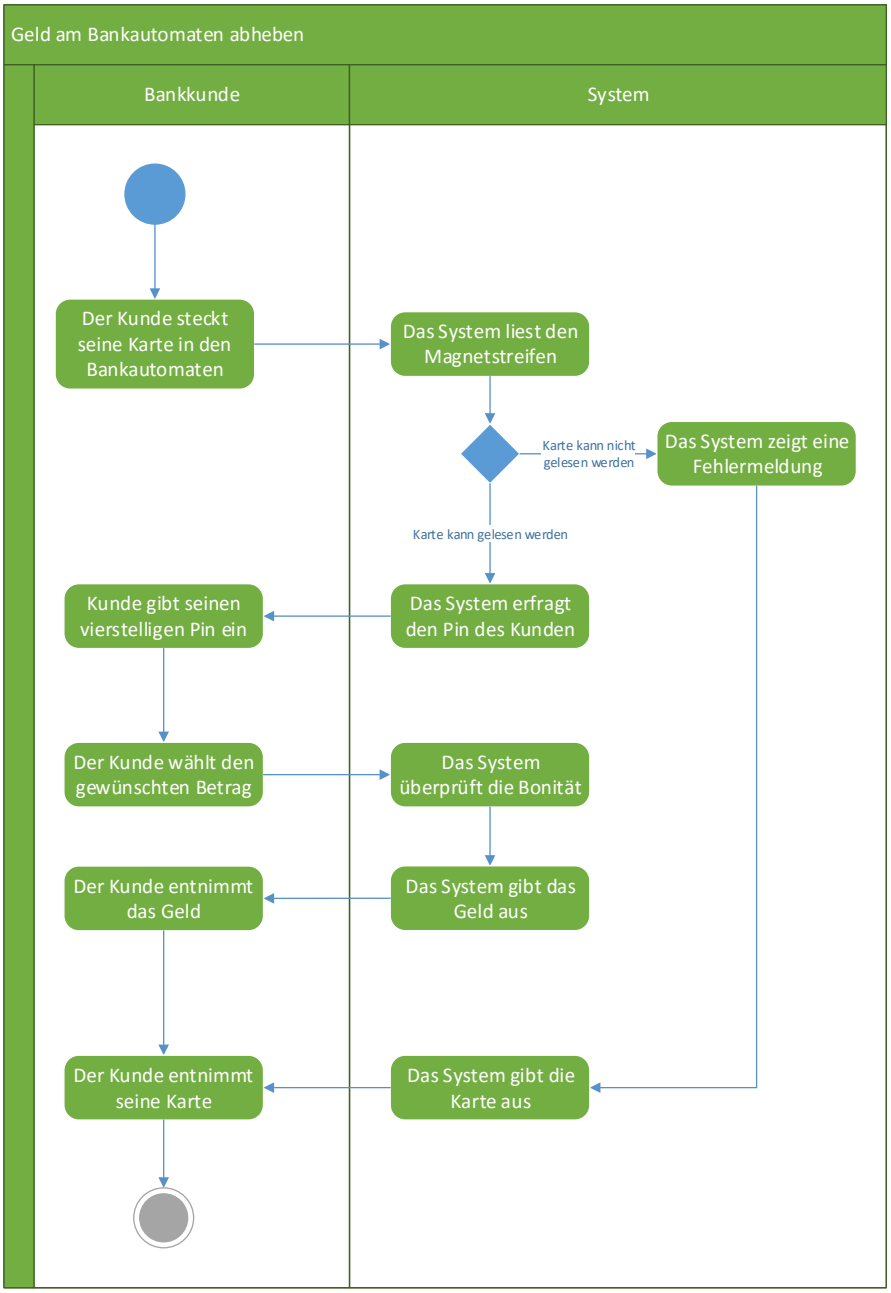


Abbildung 50: Beispiel für ein Aktivitätsdiagramm

Aktivitätsdiagramme haben gegenüber dem BPMN-Standard (siehe Kapitel 2.3.1.2.2) den Nachteil, dass sie keine Daten-/Arbeitsobjekte und Nachrichten beinhalten. Dies erschwert die Abbildung von Geschäftsprozessen in denen diese häufig fundamentaler Gegenstand des Arbeitsprozesses sind. Hierfür existieren im SE weitere eigenständige Diagrammformen wie beispielsweise die ER-Diagramme oder Datenflussmodelle.

3.2.2.1.3 Datenflussdiagramm

Ein Datenflussdiagramm veranschaulicht den Weg bzw. Transport von Daten und Informationen. Es soll verdeutlichen an welchen Stellen des Systems Informationen bereitgestellt oder verändert werden müssen (DeMarco, 1979). Prozesse (blaue Kreise in Abbildung 51) beschreiben dabei die Aktivitäten, die später von dem System umgesetzt werden sollen. Externe Objekte wie der Bankkunde oder die Bank werden in der Diagrammform als Rechtecke dargestellt. In der Terminologie von DeMarco werden diese auch als Terminatoren bezeichnet. Sie sind die Elemente, die über Datenflüsse mit dem System kommunizieren. Von der Systementwicklung können Terminatoren nicht verändert werden. Die eigentlichen Daten oder Datenspeicher, wie die Kontodaten werden als Datenspeicher bezeichnet. Auf diese können die Prozesse sowohl lesend als auch schreibend zugreifen.

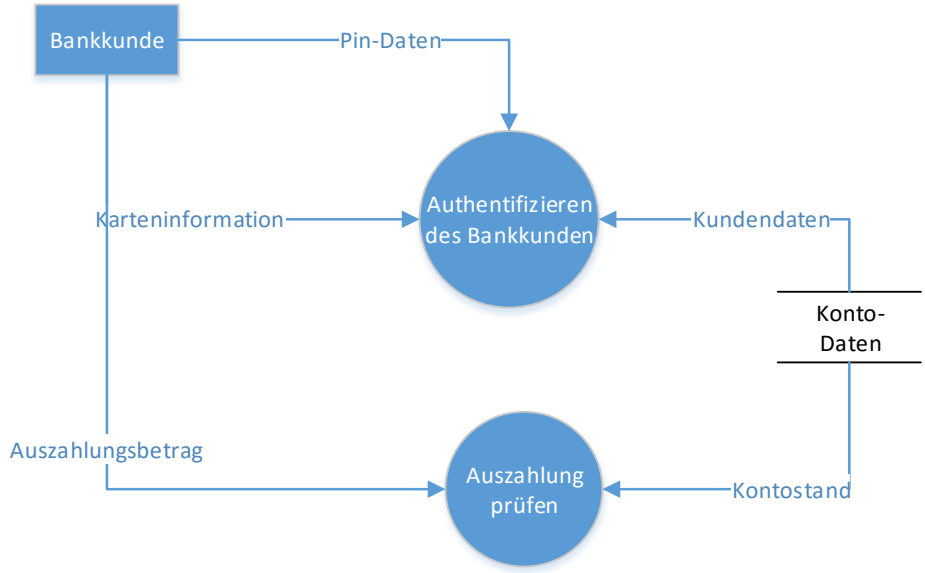


Abbildung 51: Beispiel für ein Datenflussdiagramm

Datenflussdiagramme können auch ineinander geschachtelt werden, indem ein Prozess (blauer Kreis) durch ein weiteres eigenständiges Datenflussdiagramm mit den eingehenden und ausgehenden Daten beschrieben wird. Auf diese Weise können die verschiedenen Abstraktionsgrade eines Systems beschrieben werden. Der Nachteil dieser Diagramme ist, dass sie schnell unübersichtlich werden können. Auch die Trennung der jeweiligen Prozesse auf einer Ebene ist nicht immer eindeutig. Für die Funktionsaufteilungen eines Systems haben sich hier vor allem objektorientierte Modelle wie beispielsweise Klassendiagramme etabliert. Bei allen Modellen sollte nachvollziehbar sein, welche Anforderungen mit ihnen erfüllt werden. Häufig wird diese Eigenschaft auch als „Traceability" von Anforderungen bezeichnet. In dieser Phase der Modellierung können sich eine Reihe unterschiedlicher Rückfragen durch die Stakeholder ergeben, die in die weitere Modellierung einfließen sollten (Balzert, 2009).

3.2.3 Implementierung

Eine Anforderung durchläuft während des Entwicklungsprozesses verschiedene Zustände (siehe Abbildung 52).

Abbildung 52: Lebenszyklus einer Anforderung (Ebert, 2008, S. 260)

Nachdem eine Anforderung spezifiziert und verifiziert wurde, kann sie implementiert werden. Die Verwaltung der Anforderungen sollte mittels speziell dafür entwickelter Werkzeuge vorgenommen werden wie beispielsweise Track der Firma Edgewall Software. Häufig ermöglichen auch die Entwicklungsframeworks diese Aufgabe. Es sollten darüber hinaus Möglichkeiten der Nachverfolgbarkeit, Versionierung und der Änderbarkeit geschaffen werden. Viele Systemanforderungen betreffen immer wiederkehrende Problemstellungen, für die bereits bewährte Lösungswege existieren. Unter dem Begriff *Design Pattern* (Entwurfsmuster) sind für die Implementierung eine Vielzahl unterschiedlicher Lösungswege aufgezeigt. Ein solches Pattern setzt sich immer aus einem spezifischen Design-Kontext, einem Problem und einer etablierten Lösung zusammen (Grechenig et al., 2010). Ein wahrscheinlich sehr bekanntes Architektur-Pattern in der Software-Entwicklung ist das MVC[14]-Modell. Der Vorteil bei der Verwendung solcher Architektur- oder Design-Pattern ist vielseitig. Hierbei kann bei der Implementierung viel Zeit gespart und so mancher Fehlversuch umgangen werden. Entwickler finden sich in diesen Problemlösungsstrategien schneller zurecht, da sie in der Regel gut dokumentiert sind. Der einzige Nachteil besteht darin, dass bei den Entwicklern Vorkenntnisse vorausgesetzt werden müssen.

3.2.4 Test und Inbetriebnahme

Nachdem eine Anforderung implementiert wurde, gilt es diese zu testen. Hierfür haben sich im SE verschiedene Verfahren etabliert. Normalerweise ist jeder Entwickler auch gleichzeitig Tester. Nachdem er die Anforderung verstanden und umgesetzt hat, gilt es zu überprüfen, ob die Änderung auch die gewünschte Funktionsweise nach sich zieht. Häufig ist diese Form des Testens lückenhaft und oberflächlich. Änderungen wirken sich in vielen Fällen auf bestehende Funktionen aus oder es werden einfach nur bestimmte Aspekte der Änderungen von dem jeweiligen Entwickler übersehen. Hierfür haben sich im SE automatisierte Unit-Tests bewährt. Diese automatischen Tests haben den Vorteil, dass sie theoretisch nach jeder Änderung automatisch ausgeführt werden können. Änderungen, die sich auf bestehende Funktionalitäten auswirken, werden auf diese Weise automatisch immer wieder mitgetestet und der Entwickler spart sich die Zeit des manuellen Tests. Automatische Unit-Tests eignen sich in der SE zur Verifikation (siehe 3.1.2) der Änderungen.

Der Test, ob eine Änderung tatsächlich den Anforderungen des Kunden entspricht, wird im SE als Validierung bezeichnet. Hierbei reicht es nicht, einen Test als erfolgreich zu verzeichnen, wenn der Quellcode erfolgreich kompiliert wird. Erst der Test einer Software gewährleistet die Qualität des ausgelieferten Produktes. Das Testen von Software umfasst eigene Testzyklen, die bei der Planung der Fertigstellung unbedingt berücksichtigt werden sollen. Bei der Phase der Validierung gilt es Abweichungen von der zuvor festgelegten Spezifikation aufzudecken. Hier kann es sich dann im Endeffekt auszahlen, wenn die Phase der Spezifikation gewissenhaft ausgeführt und dokumentiert wurde. Abhängig von der Änderung müssen einzelne Komponenten- oder ganze Systemtests durchgeführt werden. Hier ist es dann hilfreich, entsprechende Systemarchitekturdiagramme angefertigt zu haben, um die Abhängigkeiten der Komponenten untereinander sehen zu können. Alternativ muss sonst im schlimmsten Fall ein gesamter Systemtest durchgeführt werden.

3.3 Integrationsansätze von UE in das SE

Nachfolgend wurde nach Ansätzen gesucht, die sich mit der Verbindung der beiden Bereiche UE und des SE auseinandersetzen.

3.3.1 Goal-Scenario-Coupling

Die Idee der „Ziel-Szenario-Kopplung“ unterscheidet sich in einem wesentlichen Punkt von der klassischen Softwareentwicklung. Sie hinterfragt nicht mehr nur die Anforderungen eines Kunden, sondern fragt „Warum“ etwas benötigt wird. Die in diesem Prozess empfohlene Aufteilung in Ziele, Szenarien und lösungsorientierte Anforderungen (siehe Abbildung 53) ermöglicht eine sukzessive Verfeinerung bzw. Konkretisierung des zu implementierenden Systems. Darüber hinaus ermöglicht diese Form der Konkretisierung von einem Ziel über die Beschreibung der Zielerreichung durch ein Szenario bis hin zu der Ableitung der zu implementierenden Anforderungen

[14] Model-View-Controller: Aufteilung der interaktiven Benutzungsschnittstelle in drei eigenständige Teile, da diese oft von Änderungen betroffen sind und sich durch diese Aufteilung leichter warten lassen.

(Lösungsorientierte Anforderungen) eine gute Möglichkeit der Rückverfolgung in umgekehrter Richtung. Das nachfolgende Verfahren stammt überwiegend aus den Beschreibungen von Klaus Pohl (Pohl, 2008). Jedoch speziell die Kopplung von Zielen und Szenarien für die Ermittlung nichtfunktionaler Anforderungen entstammt aus den Arbeiten einer internationalen Arbeitsgruppe aus den Jahren 1996 bis 1999 (siehe dazu Kapitel 2.4.1).

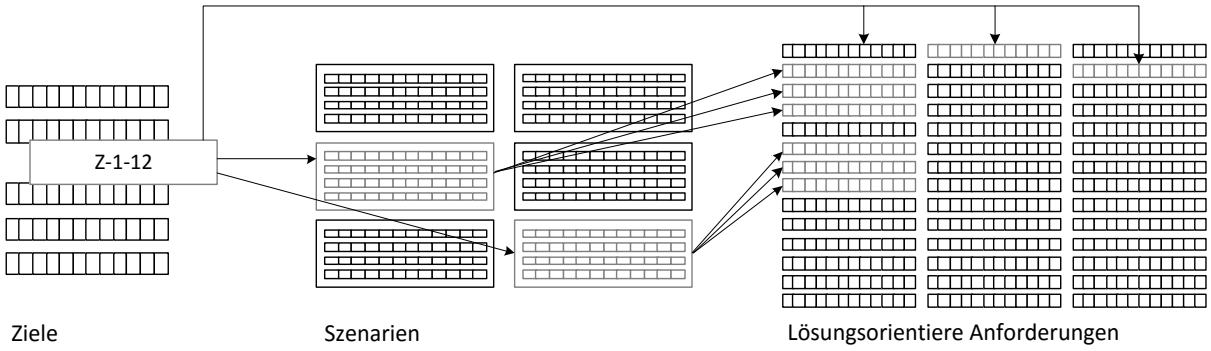


Abbildung 53: Ziel-Szenario-Kopplung (Pohl, 2008, S. 276)

Vision

Zu Anfang einer Entwicklung steht immer eine Vision. In jedem Entwicklungsvorhaben steckt der Wunsch nach einer Veränderung der Realität. Selbst bei komplexeren Vorhaben lässt sich der Kern der Veränderung kurz und prägnant formulieren. Die Vision steht über den beschriebenen Entitäten aus Abbildung 53 und beschreibt eine Art kurzer Gesamtzusammenfassung des Projektvorhabens. Anschließend wird diese Vision durch die Formulierung von Zielen konkretisiert.

Ziele

> *„Ziele werden als intentionale Beschreibungen charakteristischer Merkmale des zu entwickelnden Systems bzw. des zugehörigen Entwicklungsprozesses verstanden“.* (Pohl, 2008, S. 91)

Sie schaffen eine gedankliche Distanz zu ersten Lösungsansätzen und abstrahieren von der Nutzung des Systems. Ein Ziel kann als Begründung einer oder mehrerer Anforderungen angesehen werden. Trägt eine Anforderung nicht zu der Erfüllung eines Zieles bei, so kann es sein, dass die Anforderung überflüssig ist. In der ISO 9241-210:2011 wird ein Ziel als das „angestrebte Arbeitsergebnis“ definiert. Da Ziele miteinander in Beziehung stehen, können diese in hierarchischer Weise verfeinert werden. Ziele können sich dabei gegenseitig beeinflussen oder sogar in Konflikt zueinander stehen. Pohl schlägt für diesen Verfeinerungsprozess eine Und-/Oder-Dekomposition von Zielen vor. Bei einer Und-Dekomposition eines Zieles Z müssen alle Teilziele Z_1 bis Z_n (n>=2) erfüllt sein, um Z zu erfüllen. Bei einer Oder-Dekomposition von Z genügt die Erfüllung eines der Teilziele Z_1 bis Z_n. Darüber hinaus existieren Ziele, die sich gegenseitig widersprechen. Ein solcher Konflikt führt dazu, dass Ziele sich gegenseitig ausschließen können und somit nur eines der beiden Ziele erfüllt werden kann.

Für die Formulierung von Zielen schlägt Pohl die Einhaltung von sieben Regeln vor.

1. Ziele sollten so kurz und prägnant wie möglich formuliert sein.
2. Am besten werden Ziele in Aktivform beschrieben.
3. Ziele sollten überprüfbar sein.
4. Falls ein Ziel noch nicht überprüfbar ist, kann es in der Regel noch verfeinert werden.
5. Der Mehrwert des Zieles sollte erkennbar sein.
6. Jedes Ziel sollte eine kurze und verständliche Begründung haben.
7. Ziele sollten noch keine Lösungsansätze beinhalten.

Für die Verwendung von Zielen speziell im Bereich des Requirement-Engineerings wurden auch eine Reihe verschiedener Frameworks entwickelt, die es ermöglichen, Ziele in verschiedene Diagrammformen wie Netzdarstellungen und Bäume zu strukturieren. Bekannte Frameworks sind beispielsweise das KAOS- und das i*-Framework (Werneck, Oliveira & Leite, 2009). Und-/Oder-Bäume bzw. -Graphen stammen ursprünglich aus dem Bereich der Künstlichen Intelligenz (Bischhoff, 2007) und können ebenfalls gut zur Dekomposition von Zielen verwendet werden.

Szenarien

Wichtige Eigenschaften von Szenarien wurden in Kapitel 2.3.3.1 diskutiert. Bei den zielorientierten Ansätzen haben Szenarien den Zweck, die Ziele zu konkretisieren und der Vergegenwärtigung von Lösungsansätzen zu dienen. Umgekehrt können Szenarien jedoch auch bei der Aufdeckung noch unerkannter Ziele helfen, wie bereits Rolland (1998) aufzeigte. In den Jahren zwischen 1996 und 1999 beschäftigte sich eine internationale Arbeitsgruppe mit der Kopplung von Zielen und Szenarien in der Anforderungsanalyse. In dem Projekt CREWS (Cooperative Requirements Engineering with Scenarios) wurde zu diesem Zweck auch ein prototypisches Werkzeug entwickelt, welches in Kapitel 2.4.1 vorgestellt wurde.

In dem von Pohl vorgeschlagenem Ansatz können Szenarien als nützlicher Zwischenschritt zur Ableitung von implementierungsnahen Anforderungen dienen. Diese Idee passt gut zu der Aussage von Richter und Flückiger (2010), die ein Szenario als eine Reihe von zusammengehörigen Anforderungen aus Benutzersicht ansehen.

Lösungsorientierte Anforderungen

Eine stetige Verfeinerung der Ziele führt schließlich zu umsetzbaren Anforderungen. Pohl bezeichnet diese als lösungsorientierte Anforderungen.

> *„Lösungsorientierte Anforderungen spezifizieren alle für die technische Umsetzung des geplanten Systems relevanten Details im Hinblick auf die geforderte Funktionalität und Qualität des Systems.“* (Pohl, 2008, S. 183)

Eine lösungsorientierte Anforderung beschreibt die funktionale Eigenschaft des zu entwickelnden Systems. Der Übergang von Zielen zu Anforderungen kann dabei fließend sein.

3.3.2 Agile User Centered Design Process

Ein weiterer sehr interessanter Ansatz, das UE mit dem SE zu kombinieren, wird durch den Agilen User Centered Design Prozess vorgestellt (Paelke & Nebe, 2008). Der Entwicklungsprozess wurde für ein Projekt namens „Augmented Paper Map“ (APM) eingesetzt und bewertet. Interessant ist der Prozess, da er auf eine einfache Weise den Human-Centered-Designprozess mit dem Scrum-Prozess verbindet, indem er das Product-Backlog als Schnittstelle der beiden Prozesse verwendet (siehe Abbildung 54).

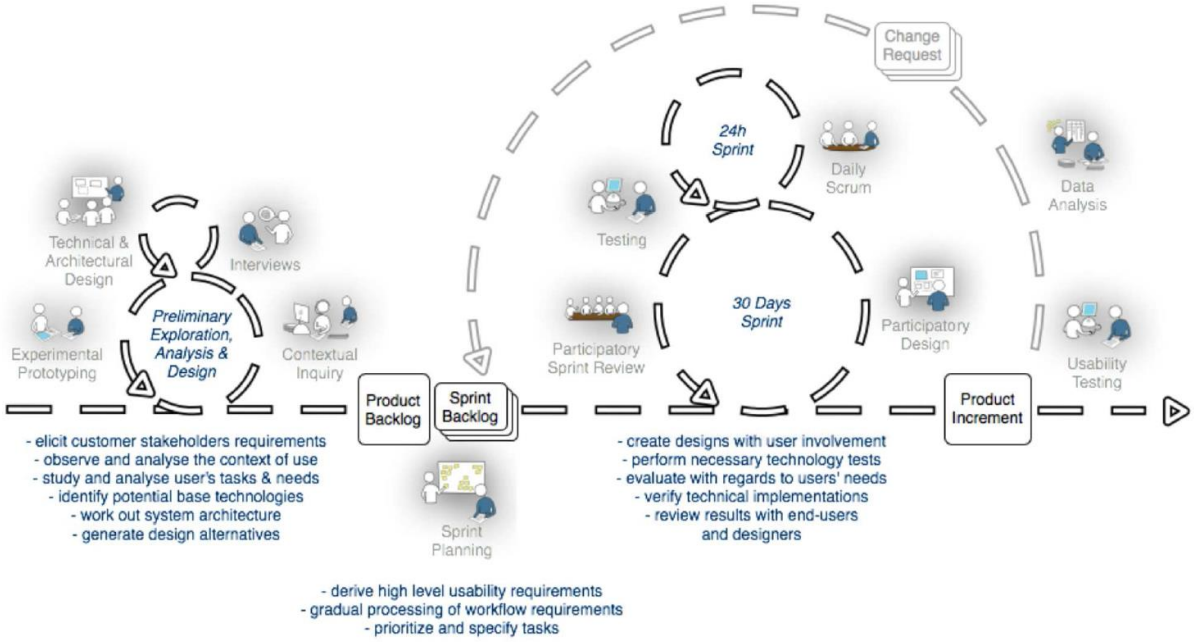


Abbildung 54: Der agile User Centered Design Prozess (Paelke & Nebe, 2008)

Das Anwendungsfeld des APM-Projektes konzentrierte sich zwar primär auf die Entwicklung von Mixed-Reality-Anwendungen, kann jedoch ohne weiteres auch auf klassische Software-Engineering-Projekte übertragen werden.

Paelke und Nebe propagieren dieses Vorgehen speziell bei MR-Projekten, da hier der Bedarf für ein „erkundungsorientiertes Design“ insbesondere ausgeprägt sei. Die Idee bei dem Prozess besteht darin, die Erkundungsphase des Scrum-Prozesses durch den HCD auszuweiten. Diese Phase dient normalerweise den Entwicklerteams dazu, sich eine Systemarchitektur zu überlegen, Kernpunkte zu erörtern und Machbarkeiten zu testen, während die Kunden-Stakeholder das Product-Backlog mit Items füllen. Im Vergleich zu herkömmlichen UE-Analysetechniken ist diese Phase bei Scrum sehr verkürzt. Scrum beinhaltet auch keine explizite Konstruktionsphase. Durch die Ausweitung der Erkundungsphase durch das HCD kann dieser dazu genutzt werden, Nutzungsanforderungen zu eruieren und alternative Schnittstellendesigns zu generieren, welche die Nutzungsanforderungen erfüllen. Paelke und Nebe schlagen als Methoden für die Analyse der Nutzungsanforderungen, Aufgaben, Ziele und Kontextinformationen und das bereits diskutierte Contextual Inquiry und Interviews vor. Die Anforderungen können dann durch die Analyse der Arbeitsabläufe aus Sicht der Benutzer abgeleitet werden und durch experimentelle Prototypen überprüft werden. Die Ergebnisse dieser Analyseaktivitäten sollten dann dem gesamten Team als Ausgangsinformationen kommuniziert werden. Gleichzeitig müssten jedoch auch MR-Experten potentielle Basistechnologien identifizieren und Architekturkonzepte entwerfen, die den Anwendungskontext erfüllen können. Die Ergebnisse dieser vorläufigen Analyse müssten dann ebenfalls in die Entwicklung der experimentellen Prototypen einfließen. Dabei ist es wichtig, dass die Benutzerziele dem gesamten Team bekannt sind. Um jedoch weiterhin „agil“ zu bleiben, sollte die Menge der dabei entstehenden Dokumentation kontrolliert werden. Es sollten leichtgewichtige Artefakte verwendet werden, die gemeinsam von allen Teammitgliedern verwendet und verstanden werden können. Das Product-Backlog beinhaltet weiterhin alle Systemanforderungen sowie organisatorische Aspekte und Aufgaben über Aufwandsabschätzungen. Das Backlog ist als Arbeitsdokument zu verstehen, welches während des gesamten Entwicklungsprozesses ständig angepasst wird. Jeder Sprint beginnt, wie bei Scrum üblich, mit einem „Sprint Planning Meeting“, um die Aufgaben für den nächsten Sprint zu definieren und so eine Teilmenge des Product-Backlogs als Sprint Backlog zu definieren. In Scrum ist das Design des Systems die Folge der Umsetzung. Nach Festlegung der ersten Systemarchitektur wird diese immer weiter ausgebaut und somit das Design verfeinert. Da bei MR-Systemen zu Projektbeginn jedoch häufig die Benutzungsschnittstelle noch unklar ist, behindert dies häufig die Umsetzung der Anwendung. Auch die benötigten Hardwarekomponenten sind in der Regel zu Projektbeginn noch nicht verfügbar. Aus diesem Grund sollte die Designphase dahingehend erweitert werden, zu Beginn des Prozesses iterativ günstig herzustellende Prototypen zu erzeugen. Sie ermöglichen die Entwicklung einer Vielzahl unterschiedlicher Umsetzungsvarianten. Diese sollten getestet und bewertet werden, um dann die vielversprechendsten Lösungen auszuwählen und iterativ zu verfeinern und weiterzuentwickeln. Um für den Prozess effektiv zu sein, ist es wichtig, dass die Design-Vorstellungen schnell und günstig erzeugt und auch problemlos verworfen werden können. In gewisser Weise verfolgt Scrum ebenfalls diesen Ansatz. Grundlegender Unterschied ist die Tatsache, dass die von Scrum erzeugten Prototypen aus implementierter Software bestehen. Daher empfehlen Paelke und Nebe bei MR-Technologien, erste Designentwürfe durch Prototyen, wie Zeichnungen, Papierprototypen und Mockups zu entwickeln und diese so häufig zu verfeinern, bis das Design ausreichend stabil erscheint, um sie in Teilen zu implementieren. Diese zusätzlichen Design-Darstellungen können leicht digitalisiert und dem Projekt-Repository hinzugefügt werden. Anforderungen, die sich aus dem HCD ergeben, sollten als übergeordnete Usability Anforderungen im Product-Backlog angesehen werden, die mit jeder Iteration weiter verfeinert werden, bis sie durch das Sprint-Backlog in den Scrum Prozess gelangen. Der Prozess unterteilt dabei in drei verschiedene Arten von Anforderungen (Zimmermann & Grötzbach, 2007):

- Usability Anforderungen
- Workflow-Anforderungen
- Interface-Anforderungen

Getestet wurde der Prozess bei der Entwicklung einer Augmented Paper Map. Ziel war die Kombination einer herkömmlichen Papierkarte mit den technischen Möglichkeiten von GPS-Geräten. Als Anwendungsszenario legte man sich auf die Navigation kleiner Segelbote fest. Als erster Schritt wurden dann Interviews und Aufgabenanalysen von UE- und UI-Experten durchgeführt. Die dabei entstandenen Usability Anforderungen wurden als erste grobe Spezifikation in das Product-Backlog eingetragen. Danach wurde jede dieser Usability Anforderungen einzeln ausgewählt und durch eine Kombination aus Aufgabenanalyse, Szenario-basiertem Design und Rapid-Prototyping verfeinert. Dadurch entstanden konkretere Einzelheiten, wie der zukünftige Workflow unterstützt werden

kann. Die entsprechenden Workflow-Anforderungen wurden ebenfalls in das Product-Backlog überführt und waren jetzt, mit nur kleinen Modifikationen in Sinne der DIN EN ISO 13407, in den Scrum Prozess überführbar. Dieses Verfahren kann über den gesamten Entwicklungsprozess angewendet werden, wodurch das Product-Backlog immer wieder mit neuen Informationen versorgt wird.

Als Ergebnis konnte festgehalten werden, dass der neue erweiterte Ansatz eine ganze Reihe neuer und innovativer Ideen hervorbrachte. Die Teilnehmer des Prozesses waren nach Aussage der Autoren überwiegend zufrieden, da der Mehraufwand minimal und dafür ein großer Nutzen zu erkennen war.

3.4 Fazit

Wie bereits erwähnt, haben trotz enormer Anstrengungen im Bereich der Softwareentwicklung im Jahr 2012 laut der Standish Group lediglich 14 % der Softwareprojekte, die nach dem klassischen Wasserfallmodell vorgegangen sind, erfolgreich abgeschlossen werden können (Standish Group, zit. nach Cohn, 2012). Der Rest hatte große Schwierigkeiten oder ist ganz gescheitert. In derselben Statistik wurde zugleich darauf hingewiesen, dass die Erfolgsquote bei Agilen Prozessen dreimal höher sei, als bei herkömmlichen Entwicklungsprozessen. Hier konnten 42 % der Softwareprojekte erfolgreich abgeschlossen werden (Cohn, 2012). Seit den Anfängen um 1970 mit dem Wasserfallmodell von Winston Royce (1970) über das Spiralmodell von Barry Boehm (1988) hin zu dem Scrum-Prozess von Ken Schwaber und Jeff Sutherland (2011) zeichnet sich ein Trend im Software-Engineering hin zu einfachen agilen und iterativen Prozessen ab. Agile Prozesse sind zeitlich gekapselte Iterationen, die evolutionäres Entwickeln, adaptives Planen und evolutionäre Produktlieferungen fördern (Paelke & Nebe, 2008). Typischerweise enthalten sie in der Regel wenige UE-Aktivitäten. Es existieren zwar eine ganze Reihe unterschiedlicher Prozesse entweder für das Software-Engineering oder das UE, jedoch auffällig wenige sind miteinander kombiniert. Vielmehr kann von einer Art Koexistenz zweier separater Prozessmodelle gesprochen werden, was zu einem erhöhten Verwaltung-, Organisations-, und Synchronisationsaufwand führt (Paelke & Nebe, 2008). Beide zuletzt vorgestellten Verfahren, die Methoden aus beiden Bereichen miteinander vereinen, nutzen im weitesten Sinne Anforderungen als Schnittstellenelement zwischen beiden Bereichen des UE und des SE. Auch der Lebenszyklus von Anforderungen (siehe Kapitel 3.2.3) deutet darauf hin, dass Anforderungen bei der Software-Entwicklung als eine Art Kernelement angesehen werden können und sich durch den gesamten Entwicklungsprozess von der Analyse bis zum Test ziehen. Jeder der vorgestellten SE-Prozesse verwendet Anforderungen, um mit deren Hilfe den Entwicklungsprozess zu koordinieren. Durch Anpassungen bei der Formulierung von Anforderungen in Form von User Stories (siehe Kapitel 3.1.5.2) oder Zielen (siehe Kapitel 3.3.1) können sie gut dazu verwendet werden, dem Kunden den abstrakten Mehrwert zu kommunizieren, der sich nach der Implementierung ergeben sollte. Im weiteren Entwicklungsprozess sollten sie als Ausgangsinformation weiterer Verfeinerungen dienen. Dabei ist es egal, ob sie nun als Ziele, Lasten, Epics oder User Stories bezeichnet werden. Wichtig ist nur, dass sie dem Benutzer den Nutzen bzw. Mehrwert der weiteren Anstrengungen verdeutlichen können und er sie auch akzeptiert. Die Verfeinerung dieser frühen Form von Anforderungen kann anschließend auf verschiedene Arten erreicht werden. Bei der Methode des Goal-Scenario-Coupling wurden Szenarien dazu verwendet, um das Design einzuleiten. Sie werden eingesetzt, um die zuvor formulierten Ziele zu konkretisieren und einen ersten Lösungsansatz zu entwickeln. Alternativ können jedoch auch Use Cases oder Prototypen dazu verwendet werden, um zu notieren, wie das zu entwickelnde System aussehen und die entsprechenden Ziele erreicht werden könnten.

Bei einem Vergleich von grundlegenden UE- und SE-Entwicklungsprozessen fällt auf, dass beide Ansätze ein ähnliches Vorgehen aufweisen. In beiden werden folgende Prozessschritte durchlaufen: Analyse, Design (Spezifikation), Entwicklung (Implementierung) und Evaluation (Test). Lediglich die Betrachtungsweise auf das zu lösende Problem unterscheidet sich. Die Methoden und Modelle des UEs haben ihren Schwerpunkt in der Betrachtung des Gesamtkontextes. Vor allem die generelle Arbeitsweise mit dem System bestehend aus Benutzer, Aufgaben, Arbeitsmittel und der Umgebung bilden eine essentielle Basis für den Erfolg der Entwicklung. Das SE sieht diese Betrachtungen zwar ebenfalls rudimentär vor, hat jedoch ganz klar seinen Schwerpunkt in der funktional-technischen Umsetzung des Problems. Erst durch eine effiziente Verschmelzung beider Kompetenzen aus SE und UE kann jedoch ein tatsächlich gebrauchstaugliches System entstehen. Die folgenden Kapitel weisen eine solche Verschmelzung von UE und SE auf. Erfahrungen positiver wie auch negativer Art werden aufgezeigt und diskutiert.

4 Eigener Ansatz zur Integration von UE und SE

Die Prozesse, Methoden und Ansätze aus den beiden vorhergehenden Kapiteln dienten im Rahmen eines *KoSSE-Verbundprojektes* der ersten Sondierung von UE-Methoden, die sich in den bestehenden SE-Prozess des Kooperationspartners integrieren lassen. Das Akronym KoSSE steht für *Kompetenzverbund Software Systems Engineering*. Partner des Verbundprojektes waren das Institut für Multimediale und Interaktive Systeme (IMIS) der Universität zu Lübeck und einem Software- und Beratungshaus. Nach ausführlicher Studie des Software-Engineering-Prozesses sowie der organisatorischen Strukturen des Kooperationspartners wurden die vorhergehend erwähnten UE-Methoden und Werkzeuge erprobt, bewertet und auf den Software-Engineering-Prozess des Software- und Beratungshauses abgebildet. Hierzu wurden die UE-Methoden in realen Entwicklungsvorhaben der Kooperationspartner angewandt, um anschließend ihren Nutzen beurteilen zu können. Dabei wurde gezielt typische Kunden des Kooperationspartners gewählt, um die daraus gewonnen Erkenntnisse bestmöglich übertragen zu können. Typische Kunden waren in den Praxiseinsätzen Bundesbehörden, Forschungseinrichtungen und Kommunen, die an dieser Stelle aus Gründen der Vertraulichkeit nicht weiter spezifiziert werden können.

Ein besonderes Augenmerk wurde bei der Abbildung der UE-Methoden auf die ersten Phasen des Software-Entwicklungsprozesses geworfen, da hier bereits zu Projektbeginn vom Kooperationspartner die größten Verbesserungspotentiale gesehen wurden. Ähnlich wie in den statistischen Erhebungen der Standish Group (Cohn, 2012) wurden auch von dem Kooperationspartner speziell unvollständige und sich ändernde Anforderungen und die mangelnde Einbeziehung von Benutzern als die Hauptgründe für optimierungswürdige Systementwicklungen identifiziert.

4.1 Das KoSSE-Verbundprojekt

Das von der EU und dem Land Schleswig Holstein geförderte Verbundprojekt KoSSE hatte das Ziel, die Methoden des UEs auf die generellen Belange von Softwareunternehmen zuzuschneiden und sie mit variabel praktizierten SE-Prozessen zu verknüpfen. Es sollten kundenorientierte Konzeptions-, Entwicklungs-, Test- und Einführungsprozesse sowie Systemarchitekturen definiert und im Unternehmen etabliert werden. Zur Unterstützung der veränderten Prozesse des Software-Engineerings sollte eine dazu passende Werkzeugunterstützung konzipiert werden, um sie in allen betroffenen Produkt- und Entwicklungsbereichen einführen zu können. Dabei kam es darauf an, Informationen über die Benutzer, deren Arbeitsabläufe, Bedürfnisse, Anforderungen, Aufgaben und Umgebung systematisch in die Softwareentwicklung einfließen und die technischen Möglichkeiten, Rahmenbedingungen und Grenzen in einer verständlichen Form an die Benutzer zurückfließen lassen zu können. Zwischen Benutzern und Entwicklern sollte auf diese Weise ein intensiverer und zielführenderer Informationsaustausch stattfinden. Das Software- und Beratungshaus war im Rahmen des KoSSE-Verbundprojektes der Pilotanwender für diese moderne Form anwendungs- und benutzerzentrierter Systementwicklung. Das auf drei Jahre angelegte Projekt unterteilte sich in verschiedene Abschnitte (siehe Abbildung 55).

Die erste Phase (Ist-Analyse) teilte sich in zwei Arbeitspakete. Zum einen sollte ein idealisierter UE-Prozess entwickelt werden. Zum anderen mussten die bestehenden Prozesse des Kooperationspartners für Konzeption, Entwicklung, Test und Einführung von Software aufwändig analysiert werden. In der zweiten Phase (Fahrplan Soll) sollte der idealisierte UE-Prozess auf die Prozesse des Software- und Beratungshauses abgebildet und angepasst werden. Die dritte Phase (Konzeption & Umsetzung) teilte sich in die Arbeitspakete zur Definition und Entwicklung des UE-Repositories inklusive Basisfunktionalität. Im zweiten Arbeitspaket sollten dann weitere UE-Werkzeuge auf Basis des UE-Repositories definiert werden. In den beiden parallel laufenden Arbeitspaketen sollte anschließend noch ein geeignetes Feedbacksystem konzipiert und entwickelt sowie auch ein Vorgehensmodell für die Qualifizierung der Software bzgl. der Bildschirmarbeitsverordnung definiert werden. Die letzten beiden Monate dienten der Berichterstattung für den Projektabschluss.

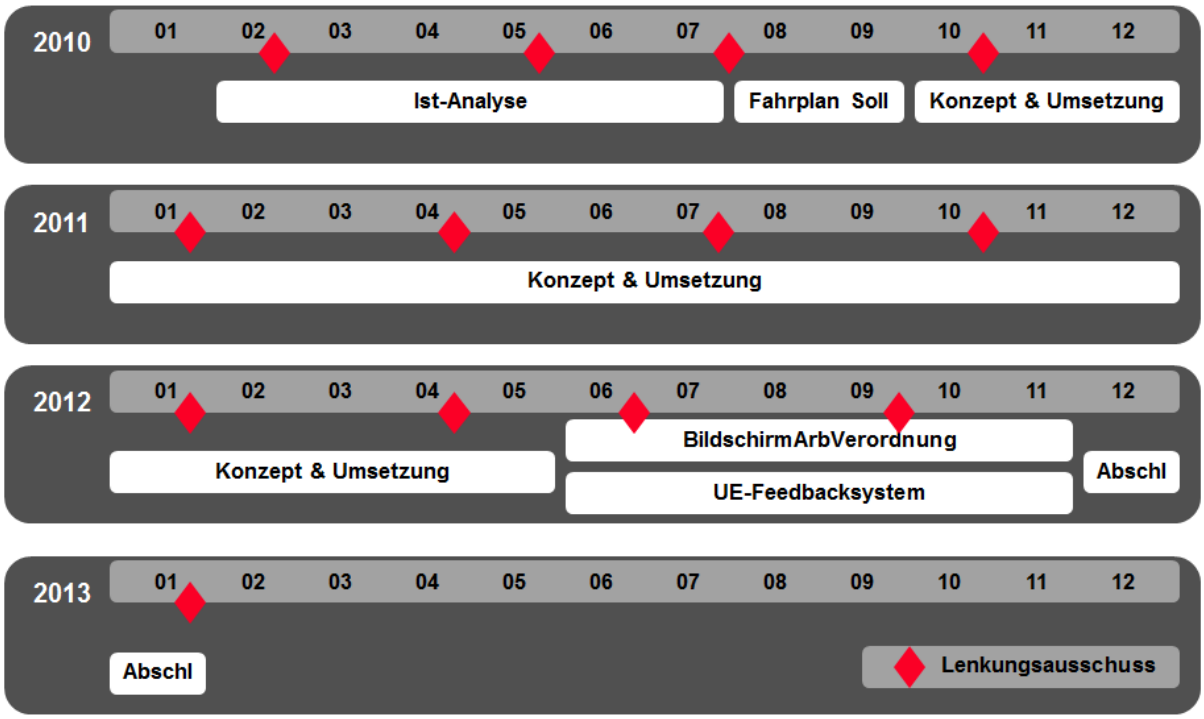


Abbildung 55: Phasenaufteilung des KoSSE-Verbundprojektes

Die Aufgaben innerhalb des Projektes wurden durch je einen Projektkoordinator auf Seiten von Unternehmen und Universität koordiniert. Seitens des Software- und Beratungshauses hatte diese Aufgabe der Qualitätsmanager des Unternehmens. Von Seiten der Universität zu Lübeck war dies meine Aufgabe. Die Bildung eines Projektteams oblag den jeweiligen Projektkoordinatoren. Das Projektteam der Universität bestand aus zwei wissenschaftlichen Mitarbeitern, aus studentischen Mitarbeitern sowie aus studentischen Abschlussarbeiten im Rahmen des Informatikstudiums. Bei dem Kooperationspartner bestand das Projektteam bedarfsgerecht aus allen wichtigen Organisationseinheiten der Organisation. In regelmäßigen Ausschusssitzungen wurden die wesentlichen durchgeführten und geplanten Aktivitäten, der Projektstand sowie strategische Fragen besprochen.

4.2 Der Kooperationspartner

Der Kooperationspartner und dessen Tochterunternehmen haben sich auf die Entwicklung und Einführung von Software für öffentliche Verwaltungen spezialisiert. Die Firma ist ein mittelständisches Unternehmen mit Hauptsitz in Lübeck. Weitere Standorte des Unternehmens sind Berlin, Düsseldorf und München. Bei der Software handelt es sich um eine umfangreiche Standardsoftware im Bereich Enterprise Ressource Planning (ERP) und E-Government für öffentliche Verwaltungen. Je nach strategischer Situation und Anforderungen der Bestands- und Neukunden sind kundenspezifische Anpassungen der Standardsoftware vorzunehmen, die in den Standard einfließen. Diese kundenspezifischen Anpassungen sind in Abbildung 56 als Teilmengen mit der Bezeichnung „Projekt x“ angedeutet. Jedes Projekt beschreibt dabei verschiedene Prozesse und Funktionalitäten, die mit der Software bearbeitet werden sollen. Abhängig von dem jeweiligen Projekt werden bereits Funktionalitäten durch die Standard-Software abgedeckt und befinden sich somit innerhalb der Standard-Software. Bei diesen müssen Funktionalitäten neu angepasst und umgesetzt werden. Da diese Anpassungen sich auf alle Kunden der Standardsoftware auswirken, ist hierbei besondere Vorsicht geboten. Erweiterungen, die keine bisherige Funktionalität der Standardsoftware betreffen, sind in Abbildung 56 als Projektbereiche aufgeführt, die die Standardsoftware übertreten.

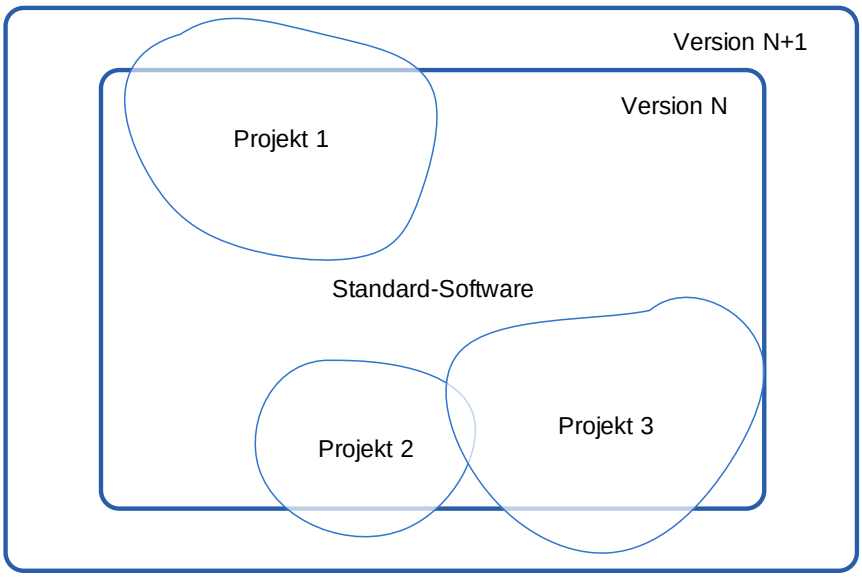


Abbildung 56: Standard-Software mit Anpassungen für den Kunden

Die Anpassungen dienen der Funktionalitätserweiterung der Standardsoftware. Sie fließen als Erweiterung in die neue Version mit ein. Diese Erweiterungen werden im Folgenden als *Konzeptumsetzungen* bezeichnet. Neben diesen fließen zudem *Interne Anliegen* und *Fehlerbehebungen* in die neue Version mit ein.

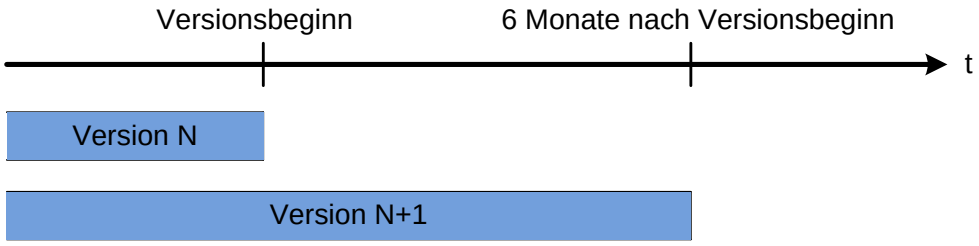


Abbildung 57: möglicher Versionszyklus der Software

Mit Freigabe einer Version N endet die Weiterentwicklung dieser Version, hier gibt es nur noch Fehlerbehebungen. Die Weiterentwicklung geschieht dann in Version N+1.

4.3 Die Ist-Analyse: Der SE-Prozess des Kooperationspartners

Bis zum Projektbeginn orientierte sich der Entwicklungsprozess bei dem Kooperationspartner weitestgehend an dem klassischen Wasserfallmodell (siehe Kapitel 3.1.1). Einzelne Kundenprojekte führten zwar zu einer evolutionären und iterativen Erweiterung der Funktionen der Gesamtsoftware, das einzelne Kundenprojekt durchlief jedoch den Prozess nur sequentiell, wie in Abbildung 58 schematisch dargestellt.

Im Folgenden werden die einzelnen Aktivitäten dieses Prozesses und die dabei gewonnenen Erfahrungen kurz aufgezeigt. Sie spiegelten den Ist-Zustand für diese Arbeit wider, der anschließend um die möglichen Methoden aus dem UE erweitert werden sollte.

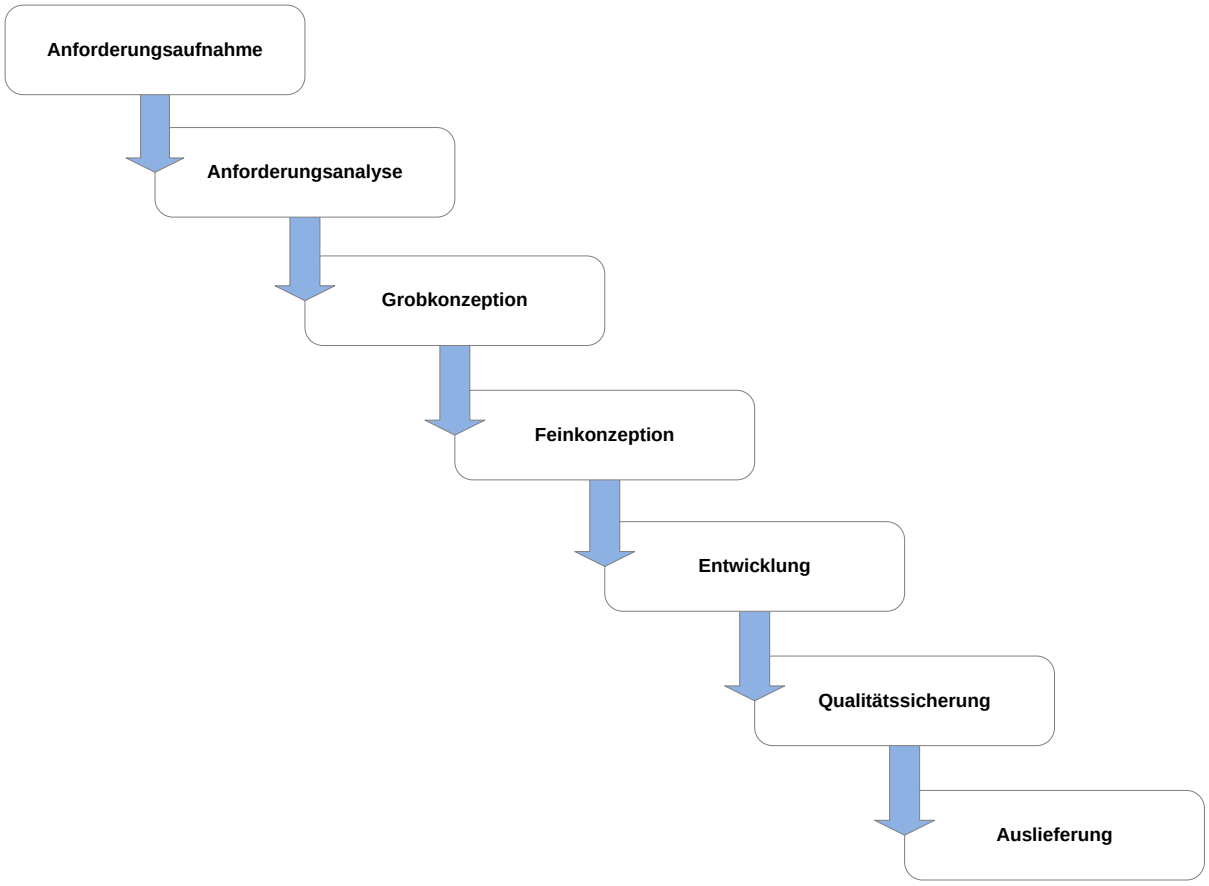


Abbildung 58: Ausgangssituation des bestehenden SE-Prozesses

4.3.1 Anforderungsaufnahme

In der Anforderungsaufnahme nimmt ein Kundenbetreuer die Ziele und Anforderungen des zu entwickelnden Systems bei dem Kunden auf und dokumentiert sie auf Basis einer Dokumentvorlage. Dies hat den Zweck, ein genaues Verständnis für die Ziele des Kunden zu erhalten. Der Kunde wird dabei informell bis zur endgültigen Ausformulierung der Anforderungen mit einbezogen. Er wird darüber hinaus bezüglich der Prozessgestaltung im Sinne einer dem Standard der Software entsprechenden Lösung beraten. Festgehalten werden, die in Abbildung 59 dargestellten Kapitel. Die Anforderungsaufnahme beinhaltet die vollständige Darstellung aller Ziele und Anforderungen des Kunden in Prosaform. Darüber hinaus werden mögliche Fehlersituationen, Ausnahmen und Sonderfälle behandelt und dokumentiert und es müssen Schnittstellen zu bestehenden Systemen und Artefakten berücksichtigt werden. Beispiele mit konkreten Zahlen und Werten aller Anwendungsfälle sollen den Entwicklern als Verständnishilfe dienen. Zusätzlich wird durch die Beispiele erreicht, Verständnisfehler zwischen Kundenbetreuer und Kunden zu bereinigen. Das Mengengerüst beschreibt, welche Zahl von Grund- und/oder Bewegungsdaten (z.B. Partner, Adressen, Abrechnungsobjekte, usw.) zu erwarten sind. Darüber hinaus soll hier auch die erwartete Häufigkeit der Ausführung der zu entwickelnden Funktionalität erhoben werden.

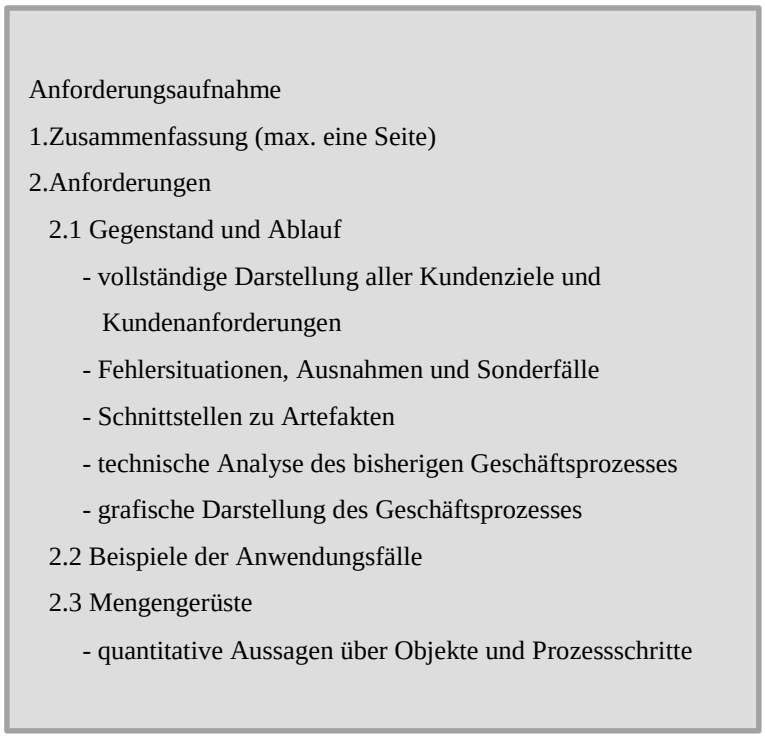


Abbildung 59: Struktur für Dokument bei Anforderungsaufnahmen

Der Kundenbetreuer hält dann Rücksprache mit einem koordinierenden Entwickler. Weites Spektrum von „Ablieferung“ der fertigen Anforderungsdokumentation bis hin zu mehreren Gesprächen oder Telefonaten zwischen Kundenbetreuer und koordinierendem Entwickler. Bereits vertraglich festgehaltene Vereinbarungen (z.B. Ausschreibungsunterlagen) werden als weiterer Bestandteil in das Anforderungsdokument mit aufgenommen. Diese werden, deutlich markiert vom Rest der Anforderungsaufnahme (z.B. durch Kursivschrift), mit in das Dokument aufgenommen.

Der Kunde muss anschließend die aufgenommenen Anforderungen noch als vollständig und korrekt bestätigen.

4.3.2 Anforderungsanalyse

Der zweite Prozessschritt "Analyse der Anforderungen" der Anforderungsanalyse ist ausschließlich intern. Ab diesem Schritt wird der Kunde nicht mehr in die weitere Konzeption einbezogen (das fertige Feinkonzept bekommt er jedoch wieder zwecks Abnahme und Beauftragung der Umsetzung). Beteiligte Personen sind der Kundenbetreuer, der koordinierende Entwickler (KENT), der Produktmanager und je nach Bedarf weitere Entwickler (ENT).

Der Kundenbetreuer hat nun verschiedene Aufgaben. Er muss Kundenformulierungen auf interne Begriffe abbilden, Anforderungen auf bestehende bzw. bereits vorhandene Softwarefunktionen oder -module übertragen sowie neue Softwarefunktionen oder -module aufzeigen und begründen. Ihm obliegt die Aufgabe der Abbildung der Prozessgestaltung der Software auf die organisatorische Struktur des Kunden. Zudem bewertet er die zu entwickelnden Themen (bzw. Funktionalitäten).

Die Erstellung der Analyse geschieht durch den Kundenbetreuer – KENT und Kundenbetreuer geben fachlichen Input, das reine Formulieren im Dokument obliegt dem Kundenbetreuer. Der koordinierende Entwickler bewertet die Machbarkeit der verschiedenen Anforderungen und versucht den daraus resultierenden Aufwand abzuschätzen.

Durch den Produktmanager kann es dazu kommen, dass die von dem Kunden formulierten Anforderungen durch interne Anforderungen ergänzt werden. Das ist dann der Fall, wenn sich beispielsweise eine bestimmte Kundenanforderung durch einen vertretbaren Aufwand zu einer neuen und weiterführenden Kernfunktionalität ausgebaut

werden kann. Weiterhin blickt der Produktmanager auf strategische Produktziele, welche durch die Erweiterungen erreicht werden können.

Mögliche Ausgänge bei der Analyse der Anforderungen sind:

1. Die Anforderung kann über bestehende Konfigurationen gelöst werden. Dies ist ein eher selten eintretender Fall.
2. Die Anforderung ist nicht mit den technischen Gegebenheiten vereinbar. Das bedeutet die Anforderung kann nicht realisiert werden.
3. Die Anforderung ist nicht mit den strategischen Produktzielen vereinbar. In diesem Fall würde keine Software-seitige Lösung realisiert werden. Es müssen in der Regel organisatorische Lösungen beim Kunden gefunden werden.
4. Die letzte Möglichkeit ist es, die gewünschte Anforderung zu programmieren. Dies hat dann zur Folge, dass die gewünschte Anforderung auch in eine nachfolgende Version der Standardsoftware übernommen werden würde.

4.3.3 Grobkonzeption

Durch die Grobkonzeption soll vor einer detaillierten Spezifikation der zu entwickelnden Softwarefunktionen zunächst das generelle Produktkonzept für die geplante Erweiterung der Software festlegt werden.

Es wird passend zu den vom Kunden geforderten Anforderungen eine grobe Beschreibung der Umsetzung formuliert. In dieser Phase sind der Kundenbetreuer, der koordinierende Entwickler, der Produktmanager und je nach Bedarf fachspezifische Entwickler beteiligt.

Der koordinierende Entwickler ist für die grundlegende Machbarkeit sowie die Einhaltung des Software-Styles bei der Entwicklung der neuen Konzeptumsetzungen verantwortlich. Er schätzt nach Rücksprache mit den fachspezifischen Entwicklern den zu erwartenden Aufwand ab, um diesen im weiteren Verlauf vertraglich festhalten zu können. Der Produktmanager hat die Aufgabe, interne Anliegen an geeigneter Stelle mit in die Grobkonzeption einzubringen. Er lenkt damit die Weiterentwicklung.

Dem Kundenbetreuer obliegt neben der eigentlichen Konzeptionsarbeit die Aufgabe, die diskutierten Inhalte der an der Konzeptionsphase beteiligten Personen zusammenzutragen und in Prosaform in der Gesamtdokumentation zusammenzuführen.

4.3.4 Feinkonzeption

In der Phase der Feinkonzeption werden die in der Grobkonzeption vorgegebenen Aspekte der neuen Softwarelösung weiter spezifiziert. Hierbei sind der Kundenbetreuer, der koordinierende Entwickler und bei Bedarf weitere fachspezifische Entwickler beteiligt.

Der Kundenbetreuer trägt wiederum die besprochenen Spezifikationen in einer Gesamtdokumentation zusammen. Durch die Feinkonzeption kann in dieser Phase auch eine detailliertere Aufwandsabschätzung abgegeben werden.

Das vom Kundenbetreuer verfasste Gesamtdokument „Fachliches Feinkonzept“ gilt in seiner Endfassung als Vertragsbestandteil und muss daher vom Kunden abgenommen werden. Die Abnahme des Kunden sowie die Beauftragung der Umsetzung ist Voraussetzung für den offiziellen Beginn der notwendigen Entwicklungsarbeiten.

4.3.5 Entwicklung

Der Software als Standardprodukt liegt ein einheitliches und für alle Kunden gleichermaßen gültiges Datenmodell zugrunde. Die notwendigen Änderungen und Ergänzungen an diesem Datenmodell werden gesammelt und fließen im Halbjahresrhythmus in eine neue Datenmodell- und Programmversion ein.

Bei der Umsetzung eines Feinkonzepts werden in einem ersten Schritt die für dieses Konzept notwendigen Änderungen und Ergänzungen in der noch nicht veröffentlichten Version des Datenmodells vorgenommen. Danach beginnen die Entwickler mit der Programmierung der jeweiligen Programmfunktionalitäten in der neuen, noch nicht veröffentlichten Version.

Zur Verifikation des Programmcodes werden parallel zur eigentlichen Programmiertätigkeit Unit-Tests erstellt. Sie dienen der wiederholbaren Überprüfung der Korrektheit einzelner Klassen und Funktionen und werden durch einen automatischen Prozess täglich mehrfach ausgeführt. Als Versionsverwaltung und Repository werden gängige CVS (Concurrent Versions System) verwendet.

Für die Dialoggestaltung existiert ein Styleguide, dessen Verwendung jedoch weiter optimiert werden könnte. Es fehlt beispielsweise an verbindlichen Gestaltungsrichtlinien für eine Reihe von in den Anwendungen verwendeten Komponenten (z.B. Tabellen, Baumstrukturen). Dies hat zur Folge, dass viele Entwickler sich bei der Umsetzung der benötigten Komponenten an fachlich oder technisch „ähnlichen", bereits existierenden Anwendungen orientieren, wodurch eine systemweite Konsistenz für den Benutzer verloren gehen kann.

In der Entwicklungsphase wird, überwiegend durch die Entwickler, ein Online-Benutzerhandbuch fortgeschrieben. Es dient der fachlichen Dokumentation von Softwarefunktionen sowie den zugehörigen modulübergreifenden Zusammenhängen im Softwaresystem und wird zusammen mit der Software an die Kunden ausgeliefert.

Alle Programmneuheiten, die in eine neue Version einfließen, werden in dieser Phase in einem zentralen Dokument gesammelt und mit Erscheinen der neuen Version zusammen mit einer Updateanleitung an die Kunden ausgeliefert. Eine technische Dokumentation existiert nach bisherigem Kenntnisstand nicht.

4.3.6 Qualitätssicherung

Das Testen der neu entwickelten und der geänderten Softwarebestandteile obliegt der Qualitätssicherung. Diese überprüft alle in eine neue Version einfließenden Änderungen und Ergänzungen (aus allen Feinkonzepten) auf ihre korrekte Umsetzung, also auf die Übereinstimmung von konzeptioneller Beschreibung und Umsetzung in der Software.

Durchschnittlich fließen in eine neue Version 150 bis 200 Entwicklungsprojekte ein, die jeweils als eigener Vorgang in einem zentralen Workflowsystem dokumentiert sind. Zu diesen Entwicklungsvorhaben gehören neben einer Reihe von Feinkonzepten auch kleine Kundenaufträge, für die eine Konzeption nicht erforderlich war („Leistungsbeschreibungen") sowie Mangelbehebungen und interne Optimierungen.

Die Testphase der Qualitätssicherung beginnt einige Wochen vor einer Versionsfreigabe und wird durch die Kundenbetreuer durchgeführt, die in dieser Zeit von ihrer sonstigen Betreuungstätigkeit freigestellt werden. Die verschiedenen Entwicklungsvorhaben werden je nach Knowhow auf die einzelnen Kundenbetreuer verteilt. Für die Qualitätssicherungsphase stehen den Kundenbetreuern verschiedene Testdatenbanken in unterschiedlichen Konfigurationsvarianten zur Verfügung.

Für die Tests existieren verschiedene Verfahren:

- Die *Basistests* prüfen grundlegende Funktionalitäten einer neuen oder geänderten Anwendung. Hier wird z.B. getestet, ob bei Formularen die Tab-Funktion die richtige Reihenfolge der Eingabefelder durchläuft, ob die Druck-Funktion korrekt arbeitet oder die Reihenfolge von Menüpunkten eingehalten wurde.
- Der *vorgangsbezogene Test* gleicht die Leistungsbeschreibung mit der Software und dem Online-Benutzerhandbuch ab bzw. prüft, ob ein von den Kunden an den Support gemeldeter Mangel behoben ist.
- Der *konzeptbezogene Test* gleicht die in einem fachlichen Feinkonzept formulierte Umsetzungsbeschreibung mit der Software und dem Online-Benutzerhandbuch ab. In der Regel ist für ein fachliches Feinkonzept zusätzlich ein Testkonzept erstellt worden, in dem die unterschiedlichen Testfälle, die notwendige Eingabe und das erwartete Ergebnis strukturiert und systematisch abgearbeitet werden können.
- Die *Integrationstests* prüfen anhand vorgegebener Testanleitungen das korrekte Zusammenspiel unterschiedlicher Module der Software. In diesen Testanleitungen sind notwendige Eingabe, erwartetes Ergebnis und tatsächliches Ergebnis einander gegenübergestellt.

Abweichungen zwischen Leistungsbeschreibung, fachlichem Feinkonzept oder erwartetem Ergebnis in der Testanleitung einerseits und der tatsächlichen Systemfunktionalität andererseits werden als Mangel angesehen. Der Tester klassifiziert sie und berichtet sie strukturiert über das Workflowsystem an den betroffenen Entwickler. Nach Behebung des Mangels prüft der Tester die korrekte Mangelbehebung.

4.3.7 Einführung

In der Regel stammt ein fachliches Feinkonzept aus dem Kontext eines Software-Einführungsprojektes bei einem Kunden. Die für das Projekt tätigen Kundenbetreuer haben das fachliche Feinkonzept selbst erstellt und zumindest Teile der Software-Tests selbst durchgeführt.

Nach Freigabe einer neuen Software-Version, wird die Software für alle Kunden auf einem Server für den Download bereitgestellt. Nach Installation der neuen Version auf der Hardware des Kunden übernimmt das Projektteam gemeinsam mit Vertretern des Kunden die weitere Einrichtung sowie die ggf. notwendigen Schulungsaktivitäten für den produktiven Einsatz des erweiterten Softwaresystems.

4.3.8 Erkannte Schwachstellen des bisherigen SE-Prozesses

Der SE-Prozess orientiert sich stark an einem stringenten Wasserfallmodell. Die dabei entstehende Dokumentation wird überwiegend mit gängigen Textprogrammen erstellt. Dies erschwert eine interne Kooperation unter den Mitarbeitern. Der bisherige Prozess sieht keine durchgängigen Methoden der UE-zentrierten Konzeption und Entwicklung vor. In einem einwöchigen Workshop wurde zwar eine Reihe von Persona mit typischen Anwendern der Software erstellt, diese fanden bisher jedoch wenig Einfluss in der Entwicklung. Nach Fertigstellung des Grobkonzeptes findet in dem Prozess bisher kaum Rücksprache mit den Kunden statt, wodurch es nach eigenen Angaben immer wieder zu „negativen Überraschungen“ bei der Auslieferung der Software kam. Die Grobkonzepte, aus denen Kunden die zu erwartenden Gestaltungslösungen hätten ableiten können, verdeutlichten die entsprechenden Lösungsansätze nur sehr abstrakt. Zudem wurde bisher nur sehr wenig formativ evaluiert. Anregungen der Kunden konnten bisher nur selten berücksichtigt werden.

4.4 Entwicklung eines idealisierten Entwicklungsprozesses

Für die Entwicklung eines idealisierten UE-Entwicklungsprozesses stand bereits zu Beginn des Projektes fest, dass sich dieser auf lange Sicht grob an die Struktur und die Grundsätze des HCD der DIN EN ISO 9241-210:2011 (siehe Kapitel 2.2.1) orientieren sollte.

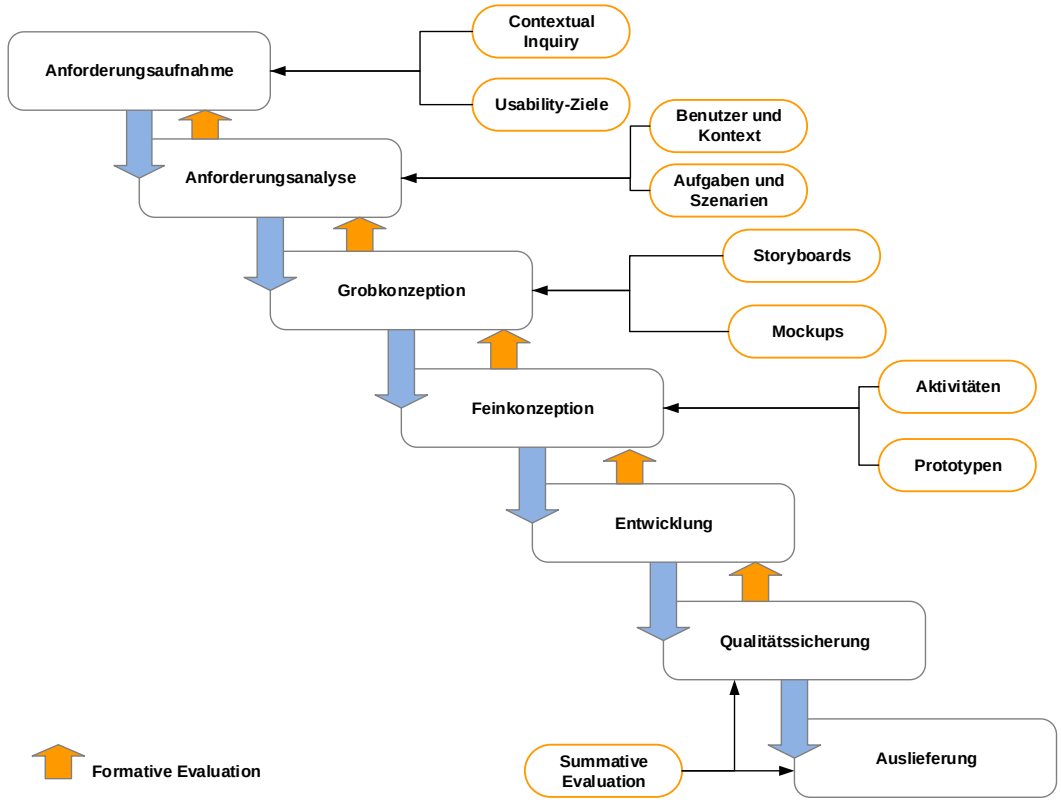


Abbildung 60: Erste Version des idealisierten Entwicklungsprozesses

Um jedoch die UE-Methoden zu erproben, mussten diese zu Anfang an dem bestehenden Prozess erprobt werden. Dabei kamen folgende Methoden zum Einsatz.

4.4.1 Formulierung von Usability Zielen

Eine erstaunliche Verbesserung bei der Erfassung von Anforderungen des Kunden ergaben die Erkenntnisse bei der „Formulierung von Zielen“ des Kunden, wie in Kapitel 3.3.1 beschrieben. Allein das methodische Erfragen dieser Ziele führte sofort zu einer konstruktiveren Haltung des Kunden. Bei vorherigen Anforderungsaufnahmen formulierten Kunden häufig die aufzunehmenden Änderungswünsche an die Software in einer Form, die wenig durchdacht war. In einigen bestehenden Anforderungsaufnahmen konnte man Sätze lesen wie: „Der Kunde x wünscht sich an Stelle y eine neue Funktion z. Dies konnte dazu führen, dass Funktionalitäten implementiert wurden, die wenig gewinnbringend waren oder sogar den eigentlichen Anforderungen des Kunden nicht entsprachen.

Durch die neue Methodik, den Kunden nach seinen „eigentlichen“ Zielen bei der Softwareerweiterung zu befragen, ergab in vielen Fällen einen komplett anders gearteten Lösungsansatz. Der Kunde wurde dadurch zu einem konstruktiven Überdenken seines selbst erdachten Lösungsansatzes eines bestimmten Anwendungsfalls gebracht. Auch der Kundenbetreuer, der für die Erhebung und Dokumentation der Anforderungen des Kunden verantwortlich ist, konnte durch das Verständnis der tatsächlichen Intention des Kunden die zu erörternde Problemstellung besser einschätzen. Häufig war diese Methodik der Auslöser für eine rege Diskussion zwischen Kundenbetreuer und Kunde, die zu einem besser durchdachten Ergebnis der zu erfassenden Problemstellung führten.

Diese Methodik führte dazu, dass bereits fertige Feinkonzepte (Pflichtenhefte) überarbeitet und die zu implementierende Funktionalitäten auf Grundlage der neuen Zielformulierung umgestaltet wurden.

Für die Dokumentation ergab sich bisher meist eine textuelle Form. In aktuellen Tests wird erprobt, ob eine werkzeugunterstützte Dekomposition von Zielen bzw. Aufgaben in Form eines Baumes bei dem Kunden akzeptiert wird. Der Vorteil einer Baumdarstellung ist seine leichte Verständlichkeit und die gute Übersicht von Aufgaben und deren Zielen.

4.4.2 Dekomposition von Aufgaben

Die Methode der Aufgabendekomposition in Form von HTAs (siehe Kapitel 2.3.1.2.1) erprobten wir mit zwei komplexeren Entwicklungsvorhaben unseres Kooperationspartners. Da uns in dieser Zeit kaum Werkzeuge bekannt waren, behalfen wir uns mit dem Mindmap-Programm „FreeMind“, um die Methode zu erproben. Aus verschiedenen Erfahrungsberichten von Mitarbeitern des Kooperationspartners leiteten wir einige Stärken und Schwächen für diese Form der Aufgabenanalyse ab. So wurde beispielsweise von Mitarbeitern des Kooperationspartners angemerkt, dass es zu Anfang schwierig sei, von der Software zu abstrahieren und nicht auf die Ebene der Interaktionsfolge des Programms zu denken. Speziell diese Anmerkung zeigte uns jedoch einen wichtigen Vorteil dieser Form der Aufgabenanalyse. So sollte speziell in der Designphase als erstes nach einem Weg gesucht werden, wie eine Aufgabe idealerweise erfüllt werden kann, bevor im nächsten Schritt dazu übergegangen wird, wie ein entsprechendes Werkzeug bei der Erfüllung dieser Aufgabe unterstützend wirken kann. Manche Probanden vermissten die Möglichkeit, bei der von uns durchgeführten Form mit dem nicht dafür ausgelegten Programm FreeMind, die ausführende Rolle (bzw. Personas) der Aufgabe angeben zu können. Manche Aufgabenbeschreibungen bedurften zusätzlicher Hintergrundinformationen, die bei der Ausformulierung den Rahmen der Übersichtlichkeit eines solchen Diagrammes überschritten. Auch der ständige Wechsel zwischen Tastatur und Maus bei der Erstellung der Aufgabendekomposition wurde bemängelt. In vielen Aufgabenbeschreibungen stellten wir die Notwendigkeit fest, die für die Aufgabe verwendeten Arbeitsobjekte erfassen und eventuell referenzieren zu können. Zudem wurde die Erfassung der Verarbeitungsreihenfolge von Teilaufgaben vermisst. Dies war in erster Linie jedoch der Wahl des von uns verwendeten Erfassungswerkzeuges geschuldet.

4.4.3 Formulieren von Aufgaben und Szenarien

Die Analyse der Aufgaben ist eng mit der Formulierung von Zielen verknüpft: Welche Aufgaben und Teilaufgaben muss ein Benutzer in welcher Reihenfolge und unter welchen Bedingungen erledigen, um ein Ziel zu erreichen?

Bisherige Anforderungsaufnahmen der Kooperationspartner wurden häufig ausschließlich mit den „Entscheidern“ auf Kundenseite durchgeführt. Dadurch kam es zu Entscheidungen, die an dem zukünftigen Benutzer vorbeigehen konnten. Wichtig in dieser Phase ist jedoch ein fundiertes Verständnis der künftigen Benutzer, ihrer Tätigkeiten und Bedürfnisse. Um dieses zu erreichen, erweiterten wir den bisherigen Prozess der Anforderungsaufnahme, um die Methodik des sogenannten *Contextual Inquiry* (siehe Kapitel 2.2.3). Hierbei untersucht der Kundenbetreuer die Bedürfnisse der Benutzer, indem er diese bei ihren Tätigkeiten beobachtet und sie dazu befragt. Mit dieser Form der Kontextanalyse sollen ausgewählte Problemstellungen beantwortet werden. Die Fragen und Anwendungsszenarien zielen auf den im Vorweg mit dem Entscheider erörterten Anwendungsfall, der zu ändern ist, ab. Die Kontextanalyse fokussiert die Tätigkeit der späteren Benutzer und das Umfeld der Anwendung. Die sich daraus ergebenen Erkenntnisse über

- Aufgaben und Verantwortlichkeiten
- Typische Rollenverteilungen
- Kommunikations- und andere Arbeitsmittel
- Kommunikationszweck und Inhalte
- Häufigkeit, Frequenz, Intensität und Dauer der Durchführung
- Ausnahmesituationen, Fehler und Spezialfälle

werden durch den Kundenbetreuer in Form von Notizen dokumentiert und anschließend als Szenarien ausformuliert.

Für eine strukturierte Wiederverwendbarkeit und dauerhafte Speicherung der erstellten Szenarien ergaben unsere Praxistests eine gute Möglichkeit, die Szenarien den vorher erstellten Aufgabenbäumen zuzuordnen (siehe Abbildung 61). Hierbei sollte immer auch die Kategorisierung in Haupt-, Alternativ- und Ausnahmeszenario geachtet werden, wie in Kapitel 2.3.3.1 beschrieben.

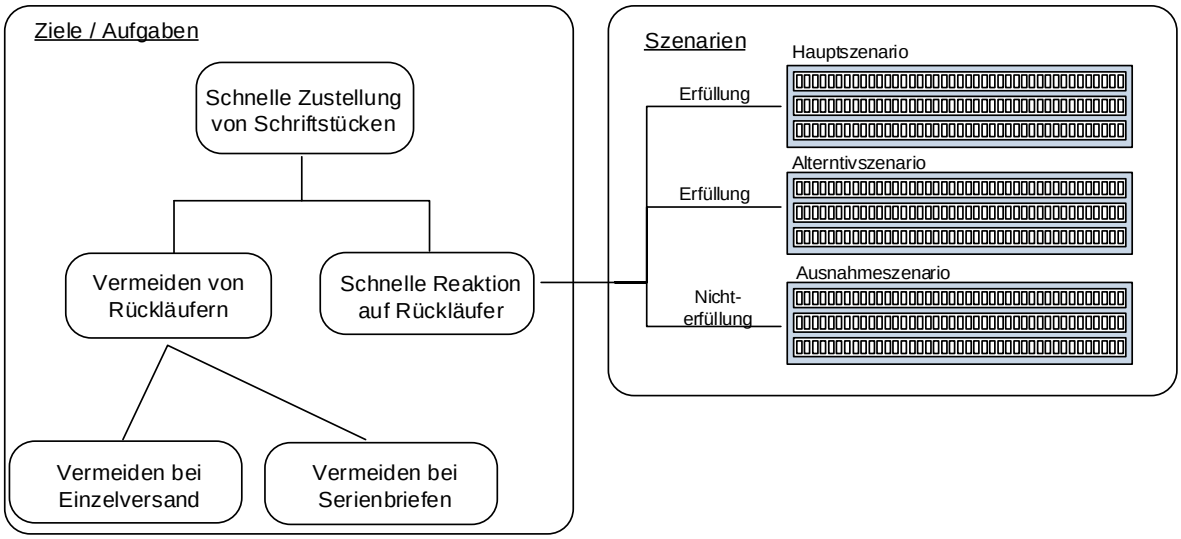


Abbildung 61: Ausschnitt des Aufgabenbaums eines kommunalen Kunden mit Szenarienzuordnung

Ziel bei der Verwendung von Szenarien ist es, ein Verständnis der künftigen Benutzer, ihrer Tätigkeiten und Bedürfnisse auf Entwicklerseite der Kooperationspartner zu erreichen. Bisher angefertigte Szenarien unterschiedlicher Anwendungsfälle realer Kunden wurden auf Entwicklerseite der Kooperationspartner als sehr hilfreiches Mittel aufgenommen, um neben den funktionalen Anforderungen des Kunden auch ein umfassenderes Bild über die Arbeitssituation des Kunden zu erlangen. Damit eignen sich Szenarien sehr gut, um sich über problembezogene Anforderungen austauschen zu können.

4.4.4 Nutzen von Storyboards

Als weitere Ergänzung der sehr positiv aufgenommen Szenarien erwies sich in unseren Praxistests die Verwendung von Screenshots (Bildschirmfotos). Die mit Bildern angereicherte Version von Szenarien, auch als „Storyboard“ bezeichnet, dient einer verbesserten Kommunikation zwischen Auftraggebern, Benutzern und Entwicklern. Ein Storyboard zeigt mithilfe der Benutzungsschnittstelle, wie die Anwendung genutzt wird und stellt somit wichtige Aspekte der Anwendung bildlich dar. Es dient damit der Kommunikation zwischen allen Beteiligten und als Visualisierung von Szenarien. Um ein noch exakteres Verständnis für die Verwendung der Anwendung des Benutzers zu erhalten, wurden die Screenshots noch mit „Interaktionspfeilen“ ergänzt (siehe dazu rechte Seite von Abbildung 62). Sie ermöglichen der Entwicklerseite die genaue Abfolge des Arbeitsprozesses eines Benutzers zu verstehen. Zudem können somit auf einfache Art und Weise störende Aspekte in der Anwendung benannt und aufgezeigt werden.

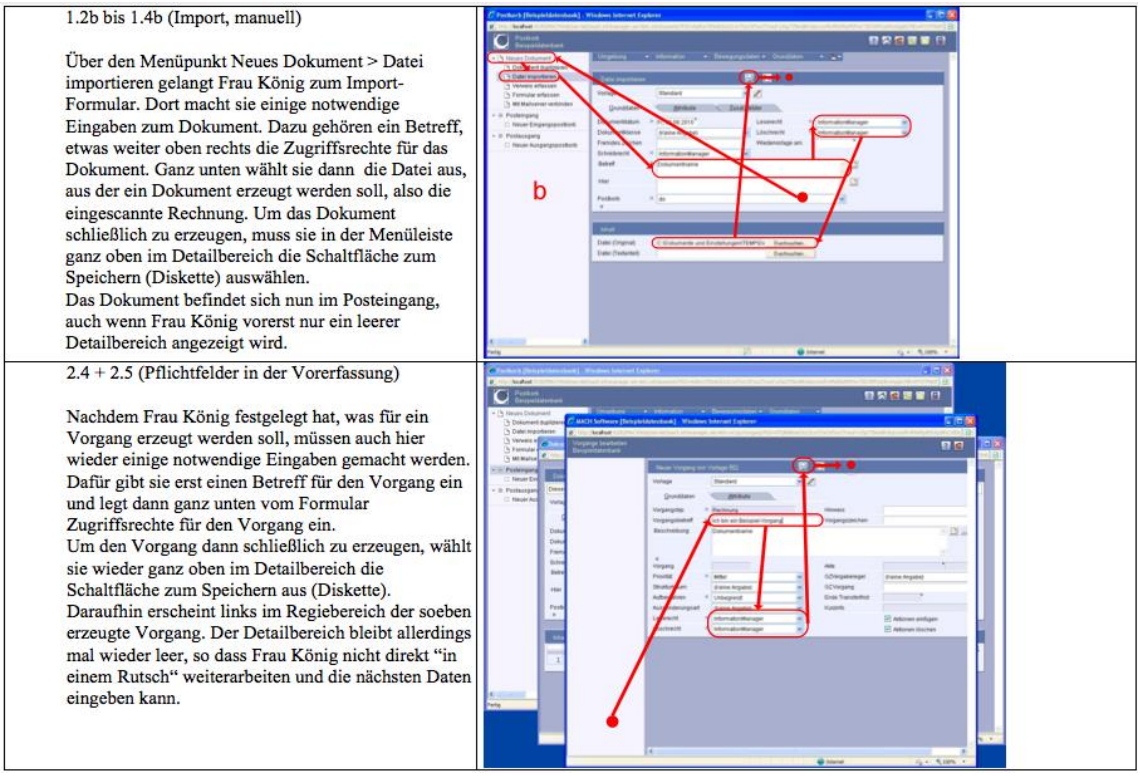


Abbildung 62: Storyboard im Praxistest

Neben der Beschreibung der tatsächlichen Arbeitssituation eines Benutzers kann ein Storyboard auch gut dazu verwendet werden, Konzeptideen der Kundenbetreuer oder Entwickler des Kooperationspartners dem Kunden zu verdeutlichen.

4.4.5 Formulierung von Aktivitäten

Da in den verschiedenen Storyboards häufig die Situation eintrat, dass alternative Interaktionspfade verfolgt werden konnten, bot sich folgende Idee an:

Aktivitätsdiagramme können dazu dienen, Storyboards strukturiert darzustellen. Durch die Verknüpfung einer Aktivität eines Aktivitätsdiagramms mit einem Abschnitt eines Storyboards (Szenario + Screenshot + Interaktionsfolgemarkierung auf dem Screenshot) können Storyboard-Abschnitte leicht und übersichtlich dargestellt werden. Dieses Verfahren wurde in einem Praxistest im Rahmen einer Studienarbeit erprobt (siehe Abbildung 63). Weiterer Vorteil dieses Verfahrens ist die genaue Beschreibung der Prozessfolge, welche gut bei der späteren Implementierung genutzt werden kann und somit mögliche Fehldeutungen ausschließt.

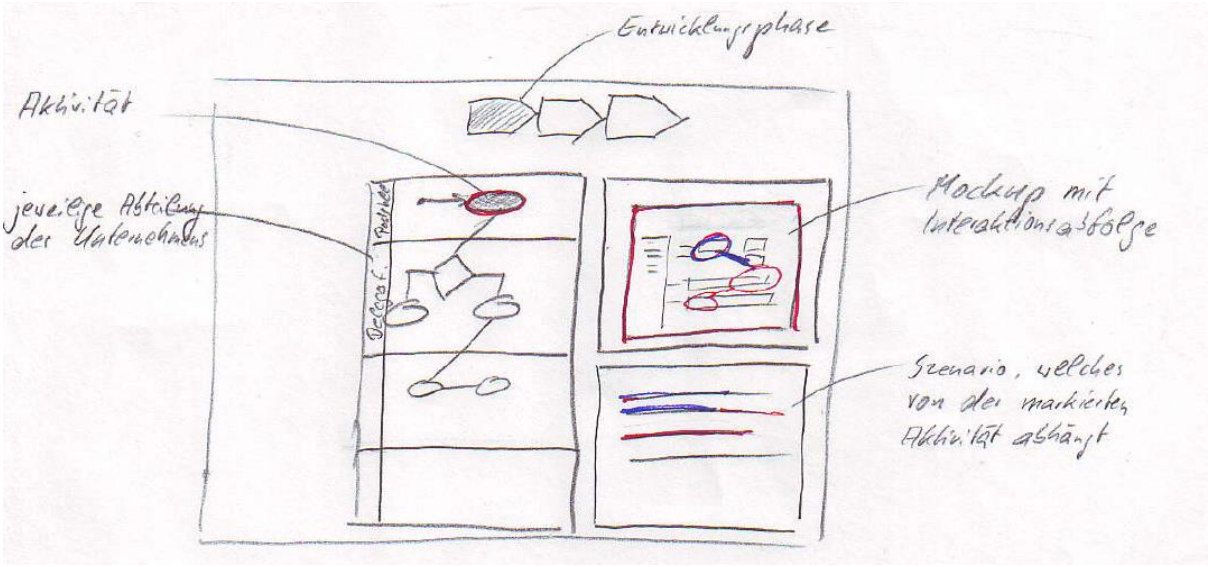


Abbildung 63: Mockup-Skizze für die Verknüpfung von Aktivitäten mit Storyboard-Abschnitten

Für die Umsetzung dieser Idee in eine Werkzeugunterstützung würde dies bedeuten, dass durch das Markieren einer Aktivität in dem Aktivitätsdiagramm zwei kontextabhängige Editierfenster zur Verfügung stehen würden (siehe Abbildung 63). Das erste Editierfenster (oben rechts) zeigt einen Screenshot der entsprechenden Anwendung mit der Möglichkeit, in dieses auch eine Interaktionsfolge zu zeichnen. Das zweite Editierfenster (unten rechts) ermöglicht die Dokumentation des Szenarios zu der entsprechenden Aktivität. Durch zusätzliche farbliche Markierung könnten Textabschnitte aus dem Szenario einem Interaktionsschritt aus dem ersten Editierfenster zugeordnet werden. Das Aktivitätsdiagramm auf der linken Seite von Abbildung 63 bietet einen weiteren Vorteil. In diesem können zusätzlich Organisationstrukturen abgebildet werden. So kann jede Aktivität der entsprechenden Abteilung der Organisation zugeordnet werden.

Zu Testzwecken wurde basierend auf der Skizze aus Abbildung 63 eine eigenständige Flash-Anwendung entwickelt, die Verknüpfungen zwischen Textabschnitten eines Szenarios und einer Interaktionsfolge auf einem Mockup erlaubte. Mittels dieser Anwendung wurde anschließend der Nutzen überprüft. Auf der linken Seite konnte das Szenario erfasst werden. In dem Beispiel aus Abbildung 64 ist dieses in Form einer Interaktionsfolge als Liste zu sehen. Auf der rechten Seite können beliebige Screenshots der Anwendungen oder Mockups eingefügt werden, mit denen die verschiedenen Interaktionsschritte aus dem Szenario durchgeführt werden können. Anschließend können beliebige Ausschnitte des Szenarios markiert und durch Platzierung eines roten Rechteckes durch Drag & Drop auf dem Screenshot verknüpft werden. Auf diese Weise kann zum einen überprüft werden, ob alle Interaktionsschritte tatsächlich durch die Anwendung (Screenshot auf der rechten Seite) realisiert werden können, und zum anderen, welche Interaktionspfade durch den späteren Anwender vollzogen werden müssen.

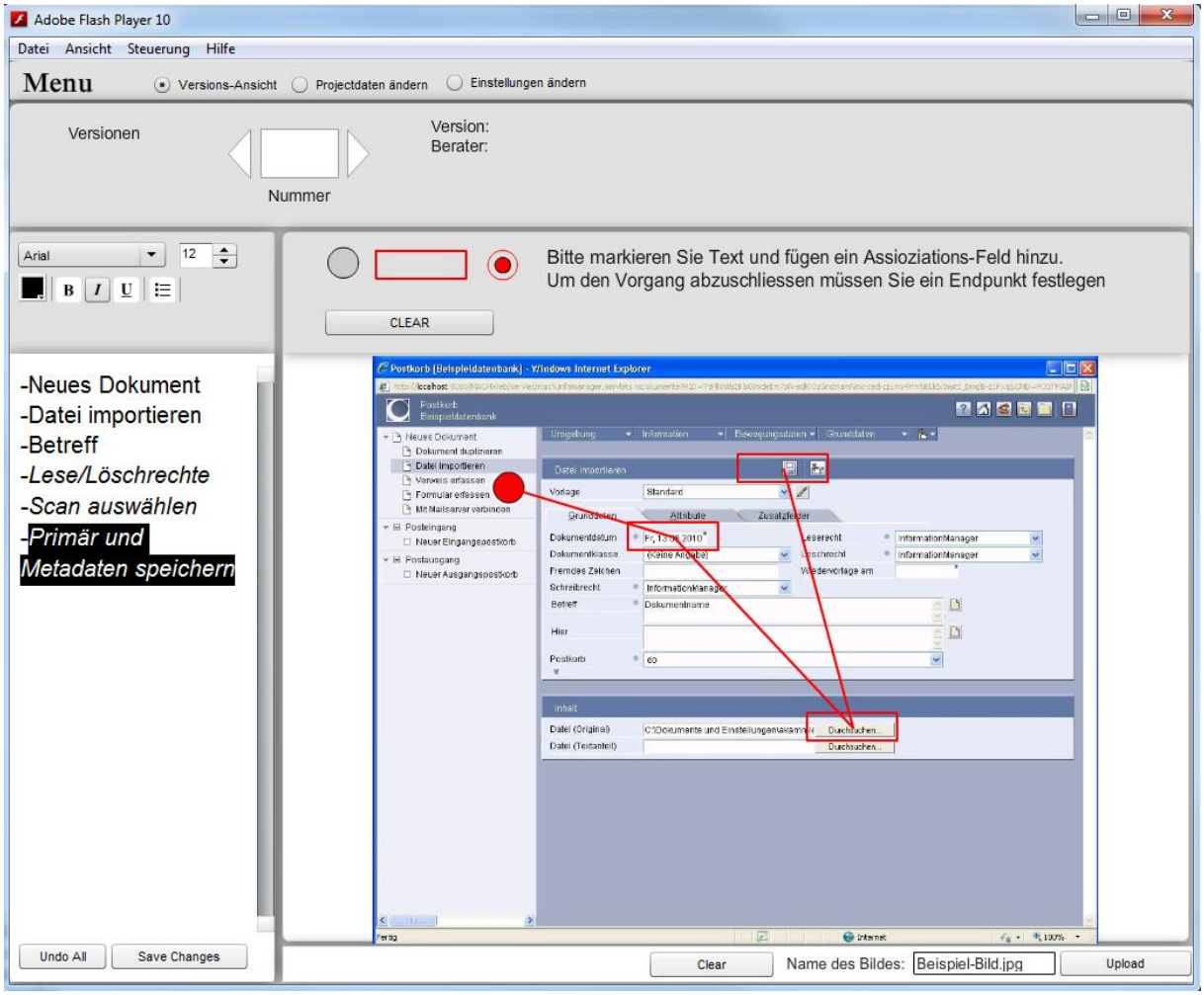


Abbildung 64: Screenshot der Flash-Anwendung für die Verknüpfung von Szenarien und Interaktionsfolge

Darüber hinaus versuchten wir die Methoden des 6-Ebenen Modells aus Kapitel 2.3.1.2 in die Praxis umzusetzen. Hierzu bedienten wir uns der klassischen Aktivitätsdiagramme. Die Idee bestand in der Schachtelung von Aktivitätsdiagrammen. So konnte beispielsweise eine Teilaktivität aus der intentionalen Ebene durch ein weiteres Aktivitätsdiagramm aus der pragmatischen Ebene detailliert werden. Dies lässt sich beliebig fortführen. Im Hinblick auf die zu entwickelnde Werkzeugunterstützung erwies sich die Schachtelung von Aktivitäten als sehr nützlich. In Abbildung 65 ist eine derartige Verschachtelung aus einem realen Praxistest abgebildet. Ähnlich der Baumdarstellung von Aufgaben (siehe HTA in Kapitel 2.3.1.2.1) kann jede Aktivität in Teilaktivitäten zerlegt werden. Dadurch wird eine Aktivität durch die Erfüllung ihrer Teilaktivitäten beschrieben.

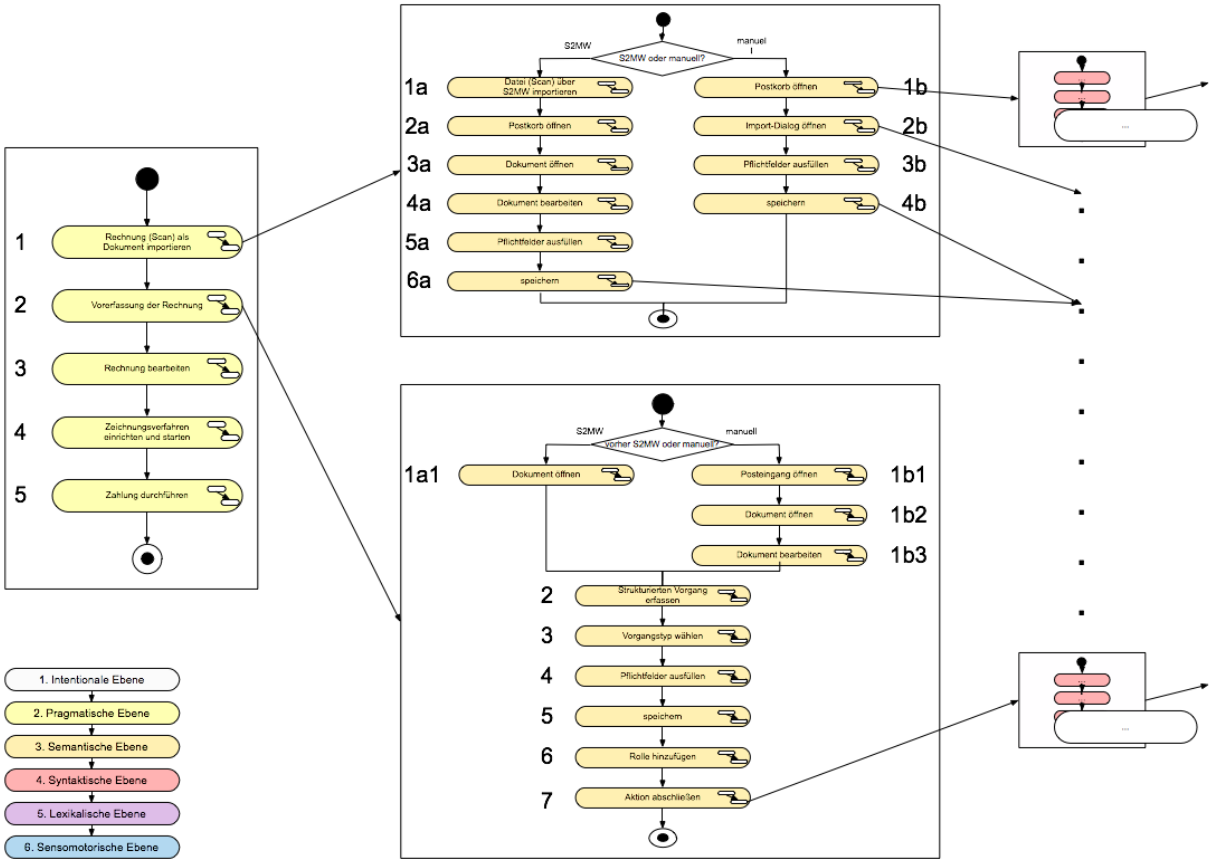


Abbildung 65: Die Umsetzung des 6-Ebenen-Modells im Praxistest

Das bisherige Verfahren des Software-Entwicklungsprozesses bei den Kooperationspartnern sah es nicht vor, die genaue Prozessfolge einer zu implementierenden Funktionalität bereits in der Konzeption zu dokumentieren. Daher konnte der Kunden manchmal nur einen vagen Eindruck von der neuen Implementierung bekommen, was für manche Überraschung im Nachhinein sorgen konnte.

4.4.6 Analyse der Organisation

Der strukturelle Aufbau der Organisation ergab sich bei den bisherigen Untersuchungen häufig implizit durch die narrative Beschreibung des Kontextes. Bei manchen Kunden des Kooperationspartners hatte sich jedoch auch gezeigt, dass die Erstellung eines Organigramms und die dazugehörige Aufgabenaufteilung des Kundenunternehmens als guter Einstieg dienten, um einen Überblick über die jeweilige Organisation zu erhalten. Das in Abbildung 66 entstandene Diagramm aus einer Personalabteilung einer öffentlichen Institution wurde dazu verwendet, um bei der später durchgeführten Tätigkeitsanalyse die Verantwortlichkeitsbereiche den verschiedenen Abteilungen zuordnen zu können. Auf der linken Seite der Abbildung ist die Dekomposition einer externen Aufgabe (siehe Kapitel 2.3.1.2) angedeutet. Auf der rechten Seite ist die Organisationsstruktur in Form eines Organigramms zu sehen (siehe Kapitel 2.3.1.3). Durch die Zuordnung der Aufgaben zu einzelnen Organisationseinheiten kann anschließend die in Kapitel 2.3.1.3 beschriebene Aufgabensynthese durchgeführt werden.

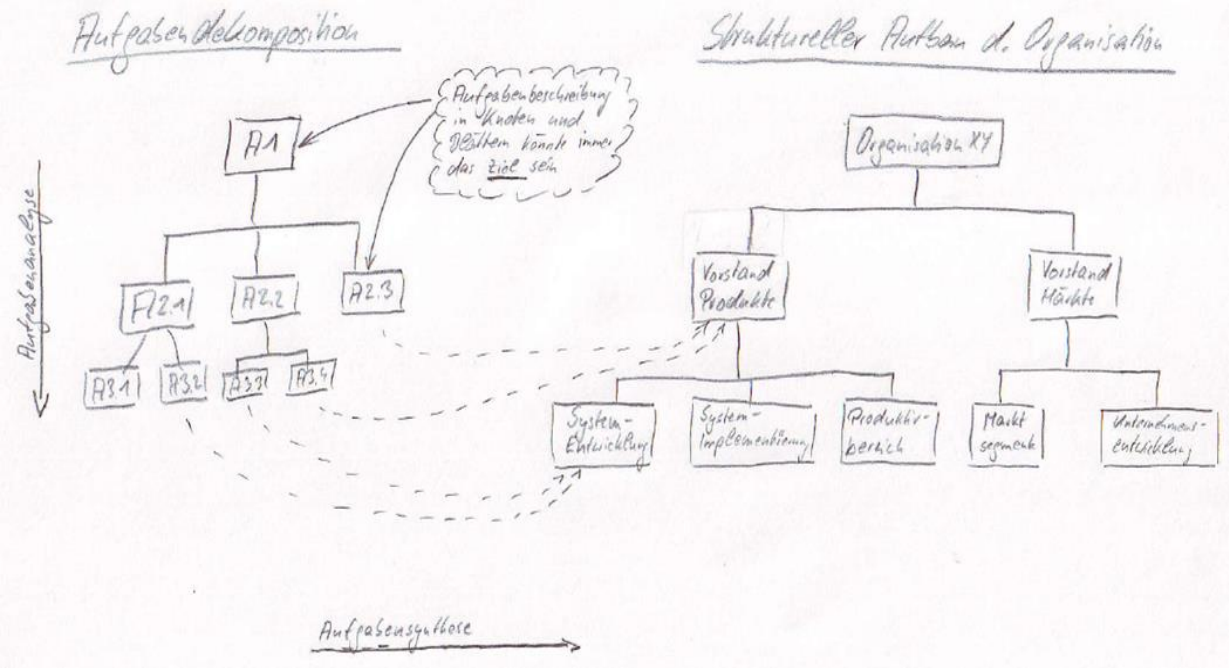


Abbildung 66: Organisationsanalyse mit einer Aufgabenanalyse gekoppelt

Diese Art der Informationserhebung ist für eine genaue Organisationsanalyse wichtig. Erst sie ermöglicht es, Abläufe und Prozesse der Organisation genau zu untersuchen und diesen dann die verwendeten Arbeitsmittel zuzuordnen.

4.4.7 Analyse der Benutzer- und Persona-Erstellung

Bei den bisherigen Probeläufen der Anforderungsaufnahme mit den Methoden der Zieldekomposition und der Szenarien-Beschreibung ergaben sich Benutzerprofile bisher lediglich implizit. Dieses Vorgehen wurde in den folgenden Probeläufen durch explizites Erfragen von benutzerspezifischen Eigenschaften ergänzt. Aus den erhobenen Benutzerinformationen wurden anschließend verschiedene Personas erstellt. Folgende problembezogene Benutzereigenschaften wurden erfragt:

Aufgabenbereich	Welche Aufgabenbereiche haben die Benutzer? Diese Frage ergibt sich häufig durch Zuordnung des Benutzers zu den entsprechenden Soll-Szenarien.
Sichten	Welche Sichten und Zugriffsrechte auf die zu verarbeitende oder entstehende Information sollen die Benutzer erhalten?
Kenntnisse	Welchen anwendungsbezogenen Wissenshintergrund besitzen die Benutzer? Mit welchem Verfahren lösen sie bislang ihre Aufgabe?
Erfahrungen	Welche Erfahrungen in der Nutzung einer bestimmten Arbeitsweise oder Anwendungssystems besitzen die Benutzer?
Fertigkeiten	Welche Routinen und Automatismen im Umgang mit Arbeitsmitteln besitzen die Benutzer?
Erwartungen	Welche Funktionalität, welche Eigenschaften und welches Verhalten erwarten die Benutzer von der Neuentwicklung? Hier soll explizit nach den Zielen und Wünschen der Benutzer zu einer bestimmten Teilanwendung gefragt werden.

Die erhobenen Informationen sollten anschließend in kleinen Arbeitsgruppen ausgearbeitet werden. Nach anfänglichen Schwierigkeiten in der Ausformulierung, wurde ein mehrtägiges Gruppentraining für die Erstellung von Personas durchgeführt. Aus diesem gingen am Ende dann etwa zehn verschiedene Benutzerbeschreibungen in Form einer Persona hervor. Neben den wichtigsten Primary Personas entstanden auch eine Reihe von Secondary und Negative Personas. Um möglichst vielen firmeninternen Mitarbeitern einen Zugang zu den Benutzerbeschreibungen zu ermöglichen, wurden diese in Sitzungs- und Aufenthaltsräumen ausgehängt. Für zwei der Persona-Beschreibungen wurden sogar Schaufensterpuppen entsprechend der Beschreibung verkleidet und aufgestellt. Die Resonanz war überwiegend positiv. Lediglich der Aufwand für die Erstellung wurde als sehr groß bewertet.

4.4.8 Herstellen von Mockups und Prototypen

„Bilder sagen mehr als tausend Worte“. Auch die bisherigen Erfahrungen im Rahmen des Verbundprojektes zeigen, dass dieses Volkssprichwort auch für die Softwareentwicklung von Bedeutung ist. Durch die Ergänzung einiger Konzepte der Kooperationspartner mit verschiedenen Arten von Mockups, konnten so manche Missverständnisse zwischen Kunden- und Entwicklerseite umgangen werden. Es zeigte sich ein gewisser Mehraufwand bei der Erstellung der Mockups. Dieser relativiert sich schnell, sieht man die positive Resonanz nicht nur auf Seiten des Kunden, sondern auf der der Entwickler.

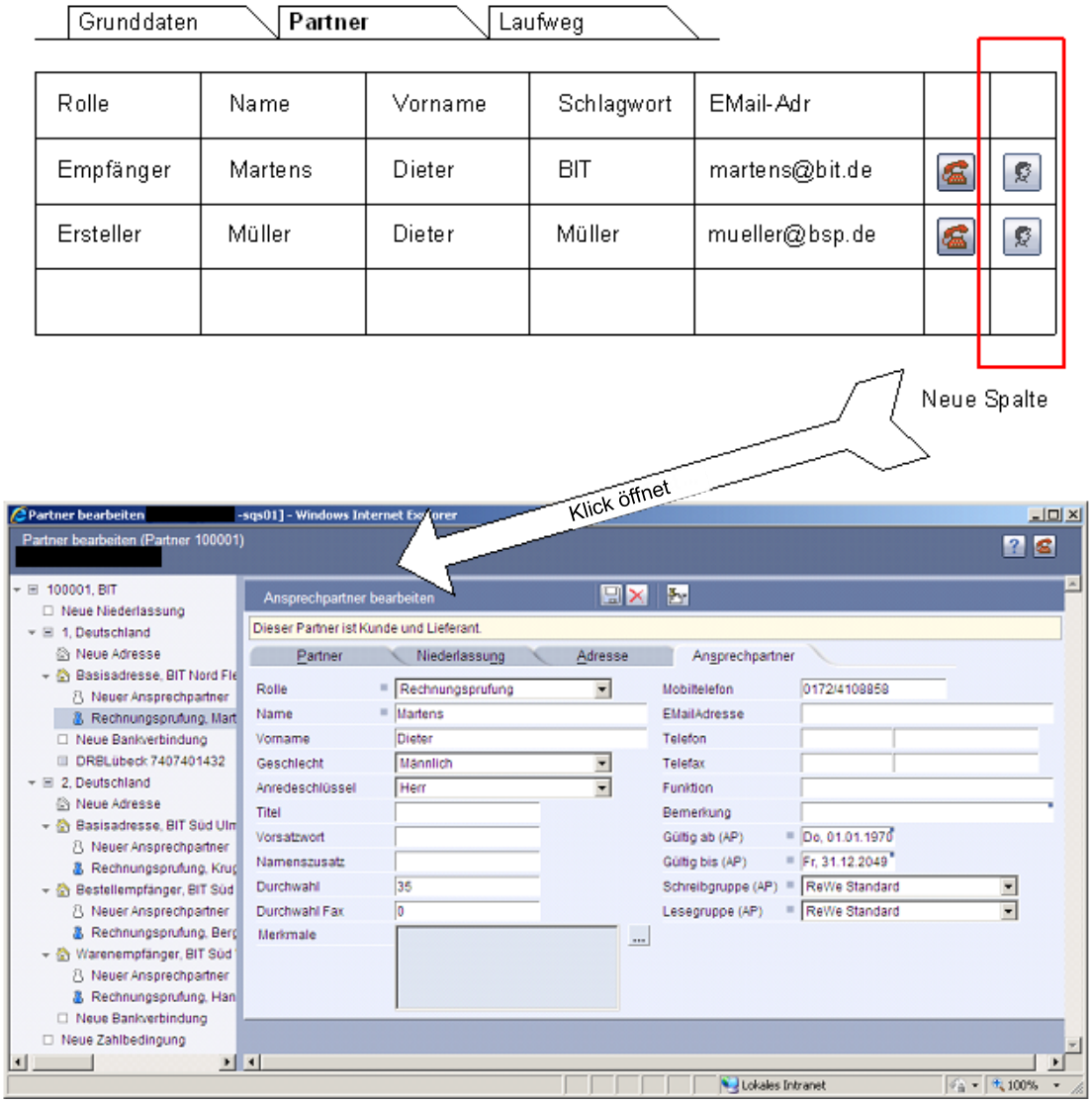


Abbildung 67: Mockup aus einem Konzept des Kooperationspartners

In bisherigen Untersuchungen wurden viele Arten von Mockups verwendet: Skizzen, Visio-Zeichnungen, manipulierte Screenshots. Abbildung 67 zeigt einen Mockup des Anwendungsbereichs „Partner bearbeiten“ der ERP-

Software eines Kooperationspartners, der um eine neue Funktionalität erweitert wurde. Dieses ergänzt um Soll-Szenarien, welche die neue Funktionalität noch textuell beschreiben, erwies sich als hervorragendes Mittel, um dem Kunden einen guten Eindruck der Neuentwicklung zu vermitteln.

4.4.9 Durchführung von formativen Evaluationen

Es ist praktisch unmöglich, auf Anhieb ein System zu entwickeln, dass allen Anforderungen genügt. Es ist notwendig, das System aus verschiedenen Perspektiven zu beleuchten. Hierzu ist es nötig, auf Entwicklerseite ein fundiertes Verständnis des künftigen Benutzers, seiner Tätigkeiten und Bedürfnisse zu bilden, jedoch auch mögliche Systemimplementierungen dem Kunden frühzeitig zu kommunizieren, um von ihm ein konkretes Feedback zu erhalten. Dieser Prozess muss bei der Softwareentwicklung mehrfach durchlaufen werden, damit das Ergebnis am Ende möglichst allen Anforderungen genügt. Dazu ist es wichtig, eine gemeinsame Sprachebene zwischen Entwickler, Auftraggeber, Fachvertreter und Benutzer zu verwenden. Alle untersuchten Methoden dienten dazu, diese gemeinsame Sprache methodisch zu verbessern und so den Austausch zwischen den involvierten Personengruppen zu erleichtern. Somit dienen die oben vorgestellten Usability Methoden dem Ziel der formativen Evaluation, sich frühzeitig mit dem Kunden auszutauschen und sein fachspezifisches Anwendungswissen in den Entwicklungsprozess einfließen zu lassen.

Im Hinblick auf die geplante Werkzeugunterstützung ergaben unsere Erfahrungen die Notwendigkeit der Schaffung einer kollaborativen Plattform, um den Informationsaustausch zwischen allen Beteiligten stärker zu unterstützen.

4.4.10 Summative Evaluation

Für die Überprüfung summativer Evaluationsmethode führten wir eine Kundenbefragung bei zwei verschiedenen Kundenbehörden mit insgesamt 11 Personen durch. Dabei konnte in unserem Fall der Vorteil ausgenutzt werden, dass sich die Kundenprojekte des Kooperationspartners meist auf die Weiterentwicklung des bestehenden Systems beziehen. Es konnte also eine summative Evaluation zu einer bereits existierenden Teilanwendung durchgeführt werden, deren Ergebnisse direkt als Ausgangssituation für eine Neuentwicklung genutzt werden konnte. Als Fragebogen verwendeten wir ISONORM 9241/110 zur Beurteilung von Software auf Grundlage der Internationalen Ergonomie-Norm DIN EN ISO 9241-110 von Prümper und Anft (2009). Als Ergebnis erhielten wir das in Abbildung 68 dargestellte Diagramm. Das Ergebnis zeigt, dass speziell die Punkte der Fehlertoleranz, der Steuerbarkeit und der Selbstbeschreibungsfähigkeit in der Teilanwendung optimiert werden konnten. Eine detaillierte Analyse und Interpretation der erfassten Daten kann in der Abschlussarbeit von Beholz (2011) nachgelesen werden. Im Rahmen jener Arbeit wurde der Schwerpunkt der Betrachtung auf die generelle Verwendung von Fragebögen für Evaluationen gelegt.

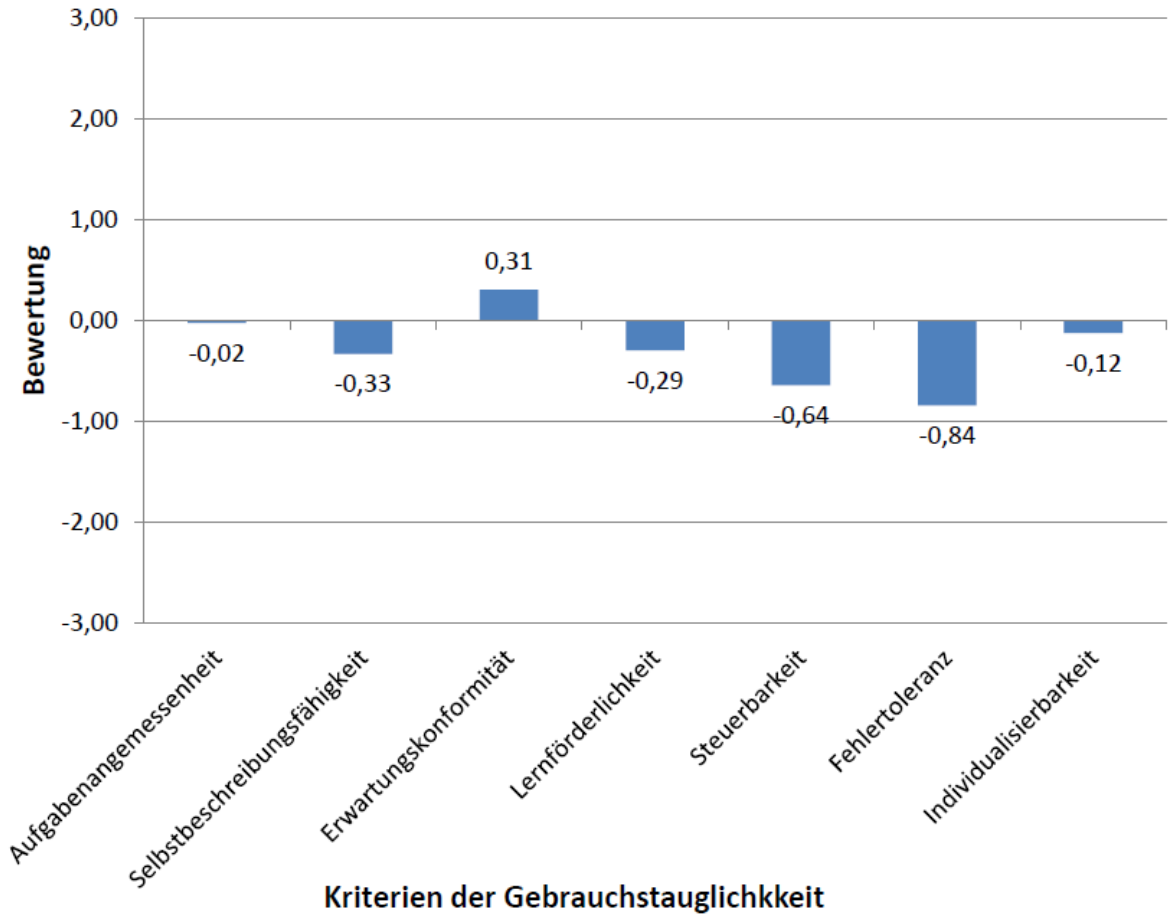


Abbildung 68: Ergebnis des ISONORM 9241/110 Fragebogens zur Gebrauchstauglichkeit

Neben dem eigentlichen Resultat der Umfrage, in welchen Punkten der Gebrauchstauglichkeit die existierende Teilanwendung idealerweise verbessert werden könnte, zeigte uns der Praxistest weitere wichtige Punkte auf. Die Durchführung summativer Evaluationen beansprucht einen enormen Zeitaufwand. Nach Aushändigung der 20 ausgedruckten Fragebögen dauerte es über drei Wochen, bis zumindest elf Fragebögen wieder beantwortet bei uns eingingen.

4.4.11 Der idealisierte UE-Prozesses

Alle oben genannten und im Rahmen des Verbundprojektes erprobten Usability-Methoden und Usability-Werkzeuge erwiesen sich als hilfreiche Mittel, um den bisherigen Entwicklungsprozess der Kooperationspartner in relevanter Weise zu ergänzen. Die Erweiterung der bisher erstellten Softwarekonzepte der Kooperationspartner um die oben aufgezeigten Usability-Methoden wurde sowohl von Entwicklerseite als auch von den Kunden als gewinnbringende Ergänzung bewertet. Als Ergebnis zeigte sich jedoch, dass das bisher stark sequenziell orientierte Vorgehensmodell zu sehr den Grundsatz des iterativen Vorgehens brach, wie in der DIN EN ISO 9241-210:2011 gefordert. Die Firmenleitung entschied sich für eine grundlegende Änderung des gesamten Entwicklungsprozesses. Der Prozess sollte sich zukünftig am Scrum-Prozessrahmenwerk orientieren, wie er in Kapitel 3.1.5.1 und 3.1.5.2 beschrieben ist. Dies führte zu einer kompletten Überarbeitung des bisherigen Ansatzes, der anhand des in Kapitel 3.3.2 von Paelke und Nebe (2008) beschriebenen Vorgehens inspiriert wurde. Die Grundidee bestand darin, den HCD-Prozess an geeigneter Stelle mit dem Scrum-Prozess zu verbinden. In mehreren Arbeitsgruppen mit den Verantwortlichen und durch den theoretischen Background entschieden wir uns für den Ansatz, wie er in Abbildung 69 angeführt ist.

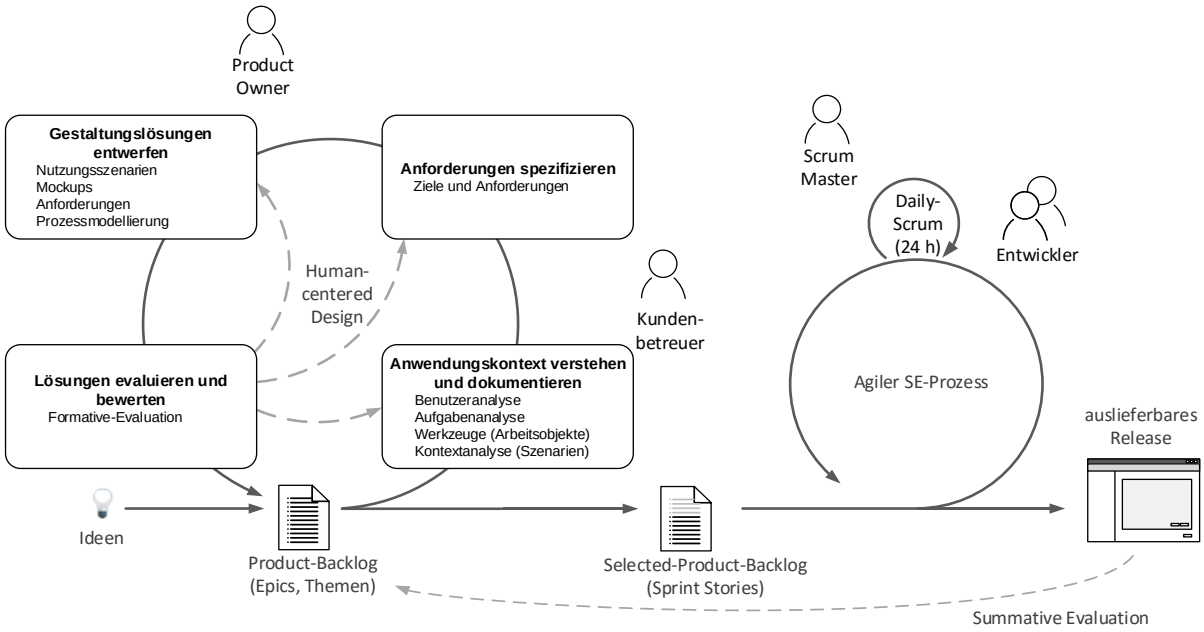


Abbildung 69: Idealisierter agiler UE-Prozess

Hauptverantwortlicher für das Product-Backlog ist nach wie vor der Product-Owner. Dieser hat weiterhin die Aufgabe, die Einträge des Product-Backlogs zu priorisieren und zu verwalten. Durch den vorgeschalteten HCD-Prozess können Gestaltungslösungen bereits im Vorweg entwickelt und bewertet werden, bevor sie dann in den Selected-Product-Backlog überführt werden. Als eine der vielversprechendsten Methoden für den Detaillierungsprozesses der Backlog-Items (Anforderungen) innerhalb des HCD-Prozesses ergaben die Vorstudien eine Kombination aus dem Szenario-based Design von Carroll und Rosson (2002) kombiniert mit dem Ansatz von Klaus Pohl (2008), diese Szenarien mit Zielen (bzw. abstrakten Anforderungen, wie Epics und Themen) zu koppeln. Ausgehend von der Befragung der Kunden und deren angestrebter Ziele, wie in Kapitel 3.3.1 und 4.4.1 beschrieben, sollten im ersten Prozessschritt durch indikative Szenarien der bestehende Anwendungskontext beschrieben werden. Weiterhin sollten in dieser Phase noch problemrelevante Informationen zu den Benutzern, ihren Aufgaben, den von ihnen eingesetzten Arbeitsobjekten sowie organisatorische Strukturen erhoben werden. Hierbei entstand die Idee, diese Informationserhebung später mittels einzelner Werkzeugmodule zu unterstützen. So könnte beispielsweise eines dieser Werkzeugmodule gezielt die Methode der Persona-Erstellung nach Cooper (2007) bereitstellen oder bei der Analyse einer Aufgabe unterstützen. Zudem sollten die auf diese Weise erhobenen Informationen in den Szenarien referenziert werden können, um bei Bedarf neben der textuellen Information noch zusätzliche Details erhalten zu können. Kundenberater und Product-Owner müssen in dieser Phase gut miteinander kooperieren können, da aus organisatorischen Gründen der Kundenberater die einzige Person mit direktem Kundenkontakt darstellt.

In der nächsten Phase können dann die indikativen Szenarien dazu genutzt werden, die sich aus den Problembeschreibungen ergebenden Nutzungsanforderungen zu spezifizieren. Diese können nach den Regeln von Pohl wie Ziele beschrieben werden. Diese Form der Zielbeschreibung überschneidet sich stark mit den Forderungen aus Scrum, in Anforderungen stets den Mehrwert für den Benutzer zu verdeutlichen. Wichtig ist es an dieser Stelle zu berücksichtigen, dass die Nutzungsanforderungen noch keine Lösungsansätze beinhalten. Dies geschieht erst im nächsten Prozessschritt.

Bei der Erarbeitung von Gestaltungslösungen werden aus den einzelnen Nutzungsanforderungen, im Sinne der Ziel-Szenario-Kopplung nach Pohl (2008), erste Gestaltungslösungen entwickelt. Abhängig von der jeweiligen Iteration des HCD-Prozesses sollte die Gestaltungslösung immer weiter detailliert werden. Hier hatte sich die Unterteilung in Aktivitäts-, Informations- und Interaktionsszenarien nach der Idee von Rosson und Carroll (2002) bewährt, um überflüssige Detailierungsarbeiten zu vermeiden.

Abhängig vom Detaillierungsgrad sollten im nächsten Schritt die Gestaltungslösungen bewertet werden. Wie bereits erwähnt war die Resonanz zu den prototypischen Gestaltungslösungen in Form von Szenarien sowohl auf

Kunden- als auch auf Entwicklerseite sehr positiv. Zudem wurde basierend auf den Erkenntnissen einer vom Institut für Multimediale und Interaktive Systeme durchgeführten Abschlussarbeit (Beholz, 2011) der zusätzliche Nutzen von grafischen Visualisierungen innerhalb der Szenarien erprobt und für effektiv befunden. Wichtig bei der Evaluierung der Gestaltungslösung ist die Verständlichkeit des Lösungsansatzes im Gesamtkontext. Im Hinblick auf die zu entwickelnde Werkzeugunterstützung wurde daher angedacht die Szenarien-Beschreibungen möglichst kollaborativ bearbeiten und bewerten zu können. Auch die Entwickler sollten bereits in der Konzeptionsphase Aussagen über technische Machbarkeiten treffen können. Basierend auf dieser Erkenntnis entschieden wir uns für eine web-basierte Umsetzung des neu zu entwickelnden Systems.

Sobald eine Gestaltungslösung sowohl entwicklungsintern als auch von den Kunden akzeptiert wird, meist erst nach mehreren Iterationen der Verbesserung, können die technisch umzusetzenden Anforderungen abgeleitet werden.

Das Entwicklungsteam, welches sich im Anschluss mit der Umsetzung dieser technischen Anforderungen auseinandersetzen muss, sollte zu jeder Anforderung zurückverfolgen können, welches Konzept (Szenario) dahinter steht. Ziele, Szenarien und die technischen Anforderungen sollten in geeigneter Weise miteinander verknüpft werden können.

Nach einem Sprint ist es wichtig, das jeweilige Produktinkrement summativ zu evaluieren. Da es gerade in größeren Entwicklungsprojekten manchmal schwer sein kann, Benutzer aktiv bei der Beurteilung des Produktinkrementes zu beteiligen, entstand die Idee, sich dieses wichtige Feedback durch Fragebögen einzuholen. Diese Form der Befragung würde es erlauben, auch Benutzer zu beteiligen, die dem Sprint Review nicht beiwohnen können. Zudem wären die Aussagen objektiver da eine Befragung in Form von Fragebögen mehr Benutzer erlauben würde.

Neben den bereits geschilderten Erkenntnissen, zeigte sich zudem:

- Zielformulierung und Aufgabenformulierung gehen ineinander über
- Nötige Kontextinformationen werden gut durch Szenarien abgedeckt
- Die Qualität der Kontextinformationen steht und fällt mit der Formulierung der Szenarien; daher müssen die Analysten (bzw. Kundenbetreuer) mit dem Umgang und der Verwendung dieser unbedingt geschult werden.
- Szenarien werden sowohl von Kunden als auch Entwicklern sehr positiv aufgenommen
- Zusätzlicher Mehraufwand resultiert in gebrauchstauglicherer Software
- Bisher verwendete Textdokumente reichen zu dieser Art der Informationserhebung nicht mehr aus

4.5 Konzeption und Entwicklung des UE-Repository

Neben dem idealisierten Entwicklungsprozess sollte auf den zuvor gewonnen Erkenntnissen ein UE-Repository entwickelt werden, welches die UE-Methoden innerhalb dieses Prozesses werkzeugseitig unterstützt. Aus der Bezeichnung „**Us**ability-**E**ngineering-**R**epository“ entstand dann der Name „UsER“ für das System.

Leitbild für die Konzeption der Werkzeugunterstützung ist die Metapher der Bearbeitung eines Dokuments (siehe Abbildung 70, links), da alle beteiligten Personen (Kundenbetreuer, Entwickler, Kunde) das Denken in Dokumenten gewohnt sind. Auch die ISO Norm 9241-210 aus Kapitel 2.2.1, empfiehlt die Anfertigung eines „Arbeitsdokumentes“ für die Erfassung entwicklungsrelevanter Informationen. Dieses Dokument könnte im gesamten Entwicklungsprozess dazu verwendet werden, es kontinuierlich zu verfeinern und zu erweitern. Dokumente können ausgedruckt werden und als Vertragsbestandteil dienen. Daher sollen alle Beteiligten so weit wie möglich den Eindruck erhalten, an einem Dokument zu arbeiten.

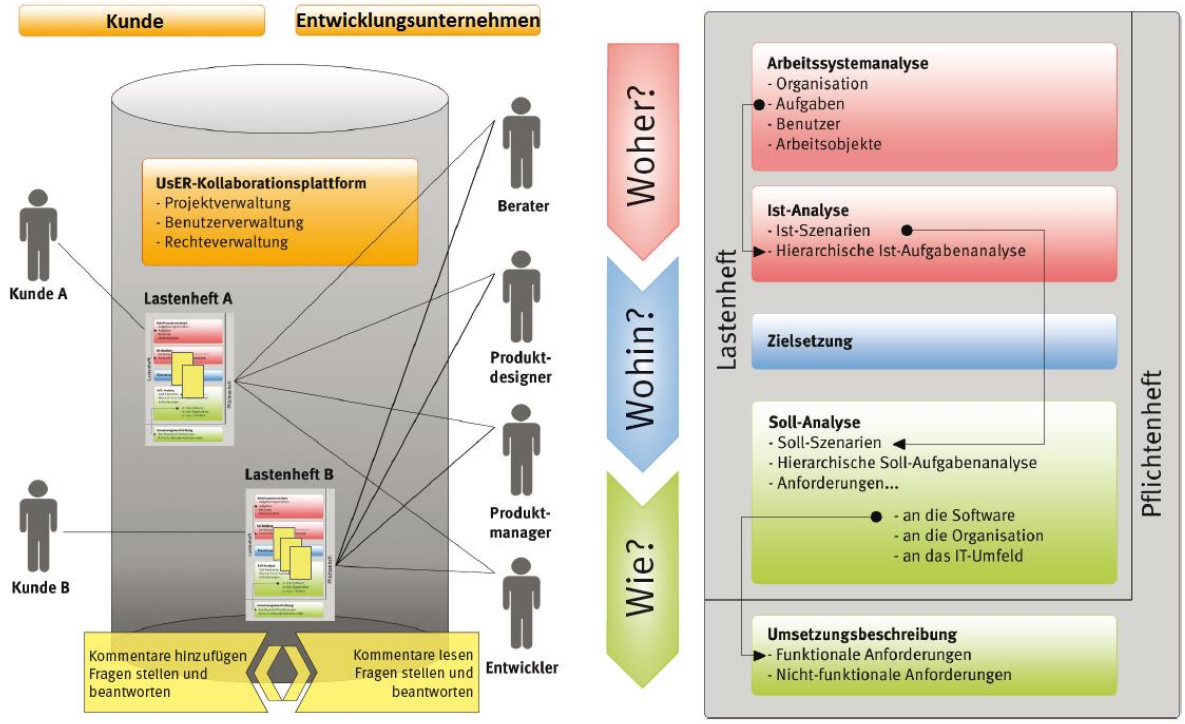


Abbildung 70: Grundidee der Kollaborationsplattform UsER

Die optische Anmutung und die Art der Bedienung des Systems sollen diese Metapher stützen. Durch die zentrale Verwaltung dieser Dokumente und die Möglichkeit, alle in einem Dokument vorkommenden Komponenten zu annotieren, wird dem Aspekt der Kollaboration Rechnung getragen.

Basierend auf den Erfahrungen aus den vorausgegangenen Vorarbeiten hat das Software- und Beratungshaus zusammen mit dem IMIS für das Dokument „Lasten-/ Pflichtenheft“ eine neue Gliederungsstruktur entwickelt, welche die wesentlichen Methoden des UE abbildet (siehe Abbildung 70, rechts). Das zu konzipierende Werkzeug UsER unterstützt die Informationsverarbeitung in den einzelnen Phasen der Entwicklung durch verschiedene methodengestützte Module und aufgabenbezogene Ansichten. Die erfassten Informationen können als Lasten-/ Pflichtenheft exportiert werden, welches die bereits in der Praxis etablierte Gliederungsstruktur nutzt.

4.5.1 Ableitung der Nutzungsanforderungen aus der Kontextanalyse

Aus den bisherigen Analysen ergeben sich die folgenden Nutzungsanforderungen für das geplante UE-Repository UsER:

- Entwicklungsprojekte sollen in Form von Dokumenten vom System repräsentiert werden. Diese Dokumente sollten ausgedruckt werden können.
- An einzelnen Projekten sollte kollaborativ gearbeitet werden können. Insbesondere Kunden und Benutzer sollten aktiv einbezogen werden können.
- Usability Methoden werden interaktiv durch einzelne Werkzeugmodule unterstützt
- Dokumentierte Informationen sollten formativ evaluiert bzw. bewertet werden können, indem bereits während der Entwicklung Konzepte und entwickelte Komponenten wiederholt zur Beurteilung vorgelegt werden können.

4.5.2 Grobkonzeption des UE-Repositories

Bei der Entwicklung des UsER-Systems wurde nach dem idealisierten Entwicklungsprozess vorgegangen, wie er in Kapitel 4.4.11 beschrieben ist. Die nachfolgend vorgestellten Konzepte sind daher das Resultat mehrfacher Iterationen dieses Prozesses. Einige der ersten Entwürfe und Prototypen sind im Anhang dieser Arbeit zu finden.

UsER hat vier unterschiedliche Ansichten (siehe Abbildung 71). Die *Login-Ansicht* dient der Authentifizierung eines Benutzers, um sich im System anzumelden. Hier können auch weitere allgemeine Information zu dem System wie das Impressum und Informationen zu den einzelnen Modulen stehen. Ein Entwicklungsvorhaben wird in UsER als einzelnes Projekt verwaltet.

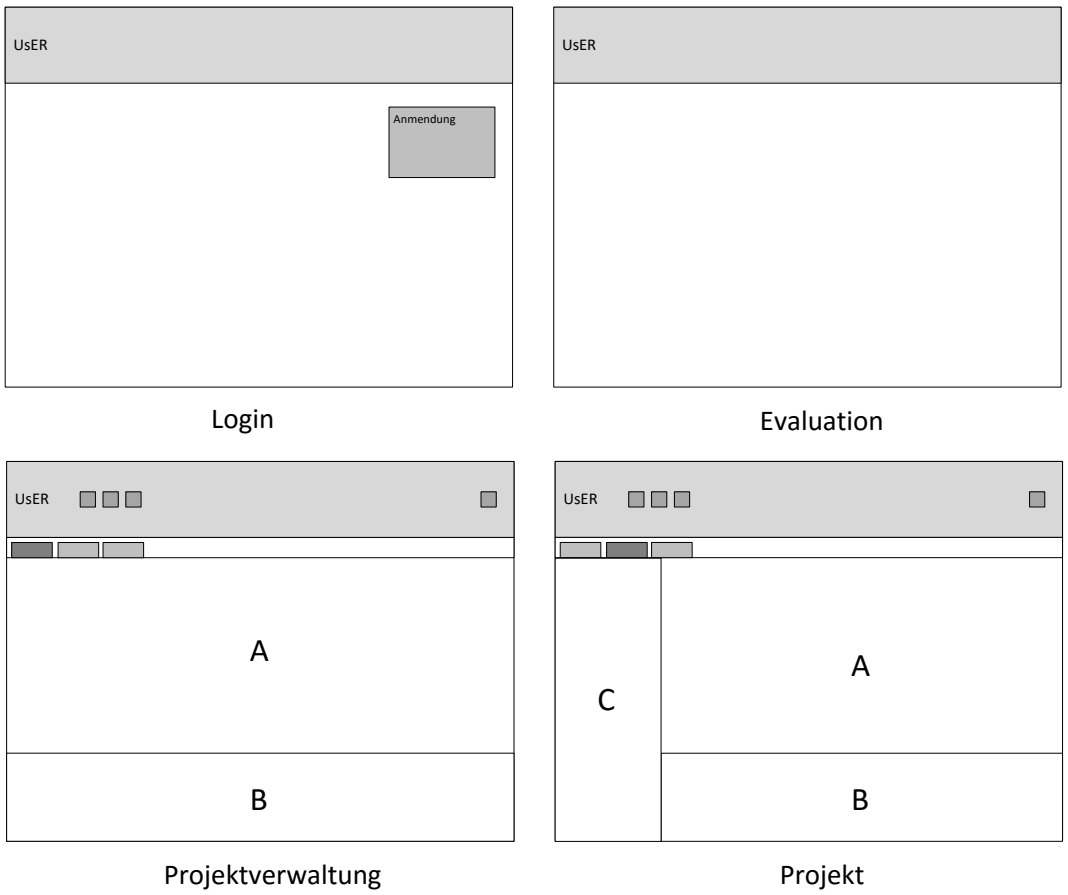


Abbildung 71: generelle Aufteilung von UsER

In der *Projektverwaltung*, werden diese in Form einer Liste von Projekten angezeigt. Die Projektverwaltung unterteilt sich in die Übersicht (A) mit der Liste der Projekte und einem Detailbereich (B), der bei Selektion eines Projektes zusätzliche Informationen zu diesem selektierten Projekt anzeigt. Der Detailbereich kann zum Editieren des jeweiligen Projektes genutzt werden. Durch einen Doppelklick auf ein Element der Projektansicht öffnet sich das jeweilige *Projekt* (siehe Abbildung 71 unten rechts). Die Projektansicht besitzt einen zusätzlichen Regiebereich (C) mit der Kapitelstruktur des jeweiligen Projektes. In ihm kann die Kapitelstruktur eines Projektes beliebig aufgebaut werden. Nach dem Öffnen eines Kapitels sollen sich alle Module auf konsistente Weise in die Bereich A und B aufteilen. Die beiden Bereiche sollen sich dabei wie bei der Projektverwaltung verhalten. Der obere Bereich A zeigt den jeweiligen Inhalt an. Der Bereich B ermöglicht das Editieren zusätzlicher Informationen der in Bereich A ausgewählten Entität. Die *Evaluations-Ansicht* ist eine separate Ansicht, die für Probanden einer Evaluation vorgesehen ist. Auf diese werden Probanden über eine dynamisch erzeugte URL geleitet, um an einer für sie freigeschalteten Umfrage eines Projektes teilnehmen zu können. Über diese Ansicht bekommen die Probanden nicht das jeweilige Projekt zu sehen. Die Ergebnisse der Evaluation werden jedoch innerhalb des Projektes angezeigt.

4.5.3 Projektverwaltung

Nach der erfolgreichen Authentifizierung gelangt der Benutzer auf die Projektverwaltung der Anwendung (siehe Abbildung 72). Basisfunktionen, die von allen Modulen der Anwendung gebraucht werden, wie das Speichern, Drucken, Kommentieren, Abmelden und weitere Systemeinstellungen befinden sich im *Hauptmenü*. In dem darunter liegenden Teil der Anwendung können die verschiedenen Ansichten über Tabs organisiert werden. Jeder

Teilbereich ist in einem einzelnen Karteireiter organisiert. Der erste Karteireiter beinhaltet immer die Projektübersicht, in der alle Projekte angezeigt werden und für die ein Benutzer freigeschaltet ist. Durch diese Form der Strukturierung kann ein Benutzer beliebig viele Projekte gleichzeitig öffnen und an diesen arbeiten.

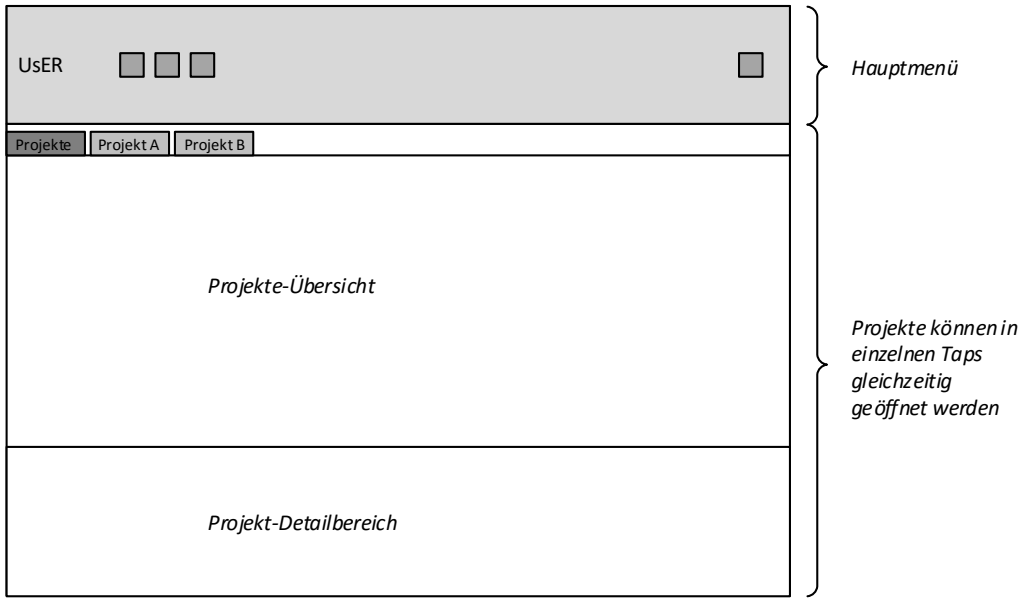


Abbildung 72: Projektverwaltung von UsER

In der Projektverwaltung können neue Projekte angelegt oder existierende Projekte gelöscht werden. Die Projektverwaltung unterteilt sich in einen oberen und einen unteren Bereich. Im oberen Bereich der *Projekte-Übersicht* werden alle freigeschalteten Projekte eines Benutzers in einer Liste angezeigt. Durch Selektion eines Projektes erscheint im unteren Drittel der *Projekt-Detailbereich*, der zusätzliche Inhalte anzeigt und das Editieren der Informationen eines selektierten Projektes ermöglicht. Bereits geöffnete Projekte erscheinen in einem eigenen Karteireiter. Mit einem Doppelklick in der Projekte-Übersicht, ist der Benutzer in der Lage, ein Projekt zu öffnen. Dadurch wird entweder ein neuer Karteireiter geöffnet oder, falls das Projekt bereits geöffnet war, zu dem entsprechenden Karteireiter gewechselt.

4.5.4 Projektansicht

Ein geöffnetes Projekt erscheint für einen Benutzer immer in einem eigenen Karteireiter. Auf der linken Seite in der Projektansicht kann im Regiebereich, wie man ihn auch von gängigen Texteditoren kennt, zu jedem Projekt eine beliebige Dokumentenstruktur angelegt werden. Dazu muss der Benutzer eines der im Folgenden vorgestellten Module wählen und durch Drag & Drop an die gewünschte Stelle im Regiebereich ziehen. Dies ermöglicht den Projektbeteiligten eine beliebige für ihre Bedürfnisse logische und sequenzielle Abfolge einer Projektbeschreibung. Durch diese Form der Strukturierung kann ein Unternehmen alle möglichen Arten eines Spezifikationsdokumentes verwenden, wie sie beispielsweise in Kapitel 3.2.1.1 beschrieben wurden. Soll eine bestimmte Struktur innerhalb eines Unternehmens immer wieder verwendet werden, kann diese unter den Vorlagen gespeichert werden.

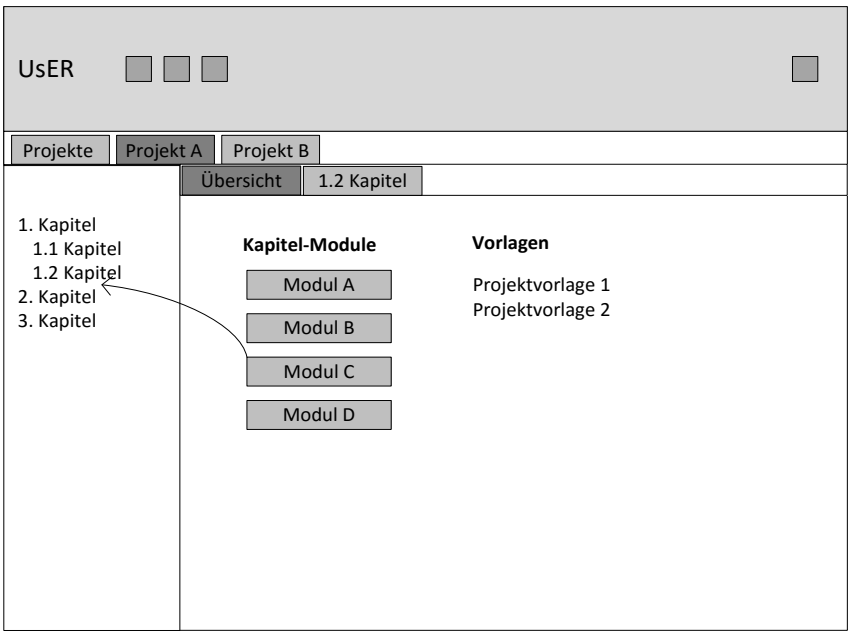


Abbildung 73: Projektansicht von UsER

Um innerhalb eines Dokumentes mehrere Kapitel öffnen zu können, werden diese wieder in Form von Karteireitern angeordnet. Der vorderste Reiter *Übersicht* beinhaltet immer die bereits erwähnte Modulauswahl und die möglichen Vorlagen.

4.5.5 Modulansicht

Da zu diesem Zeitpunkt noch keine klare Vorstellung bestand, welche und wie viele Module im Rahmen dieses Projektes in das System integriert werden, musste eine möglichst flexible Aufteilung gefunden werden. Nach mehreren Iterationen und möglichen Lösungsansätzen fiel die Entscheidung für eine Unterteilung in einen oberen und einen unteren Bereich für alle Module. Der obere Bereich sollte die darzustellenden Informationen eines Kapitels enthalten. Der untere Bereich soll zusätzliche Detailinformationen beinhalten, abhängig von der im Kapitelbereich selektierten Entität.

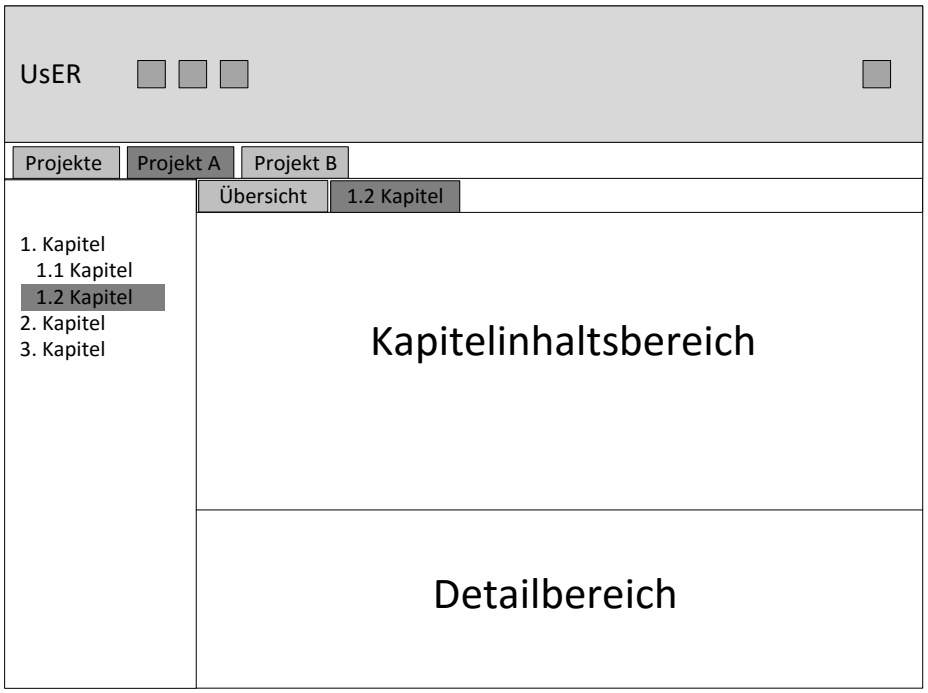


Abbildung 74: Generelle Modulaufteilung

4.5.6 Evaluation der vorgestellten Konzepte

Um die Konzepte für UsER bewerten und iterativ verfeinern zu können, wurden in dieser Phase des Projektes Szenarien und Walkthroughs für die formative Evaluation verwendet. Diese wurden in den regelmäßig stattfindenden Team-Meetings als Ausgangslage für weitere Entwicklungen verwendet. Sich daraus ergebende Verbesserungen wurden direkt in die nächste Iteration einbezogen. Im Rahmen einer Masterarbeit (Roenspieß, 2011) wurde der Stand der oben vorgestellten Konzepte durch acht Mitarbeiter des Kooperationspartners bewertet (siehe Abbildung 75). Ihnen wurden die Konzepte in Form von Anwendungsfällen und Mockups (Storyboards) sowie eines Fragebogens vorgelegt. In den Anwendungsfällen wurde den Probanden aufgezeigt, wie sie ein Projekt erstellen, bearbeiten oder neue Inhalte hinzufügen. Für alle Arbeitsschritte zeigten die skizzenhaften Mockups die dafür nötige Interaktion mit dem System auf.

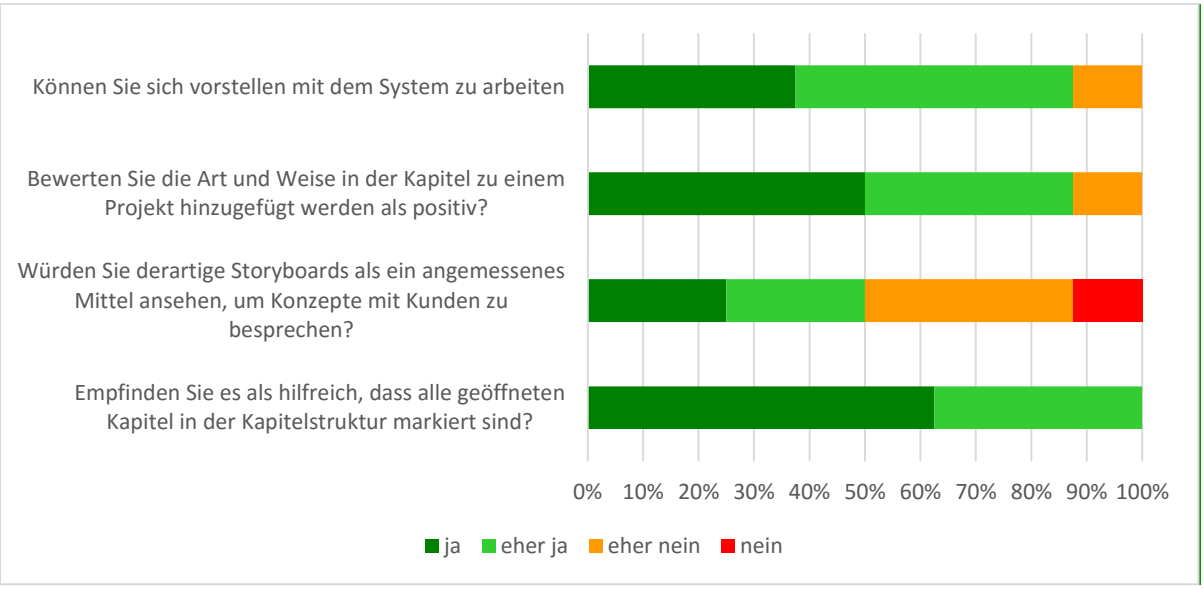


Abbildung 75: Evaluation des UE-Repository Konzeptes

Nur ein einziger Proband konnte sich „eher nicht“ vorstellen, mit dem System zu arbeiten, der Rest bewertete das Konzept überwiegend positiv. Dies zeigte uns zu diesem Zeitpunkt, dass die oben beschriebenen Grundzüge der Konzeption den Zielsetzungen entsprachen. Veränderungen im Detail können sich dann auch noch in den folgenden Iterationen ergeben. Das Hinzufügen von Modulen bzw. Kapiteln in den Regiebereich wurde ebenfalls nur von einem Probanden als „eher nicht positiv“ bewertet. Dieser gab in der Bewertung jedoch auch an, dass er die Funktionsweise und die entsprechenden Elemente nicht ganz verstanden habe. Diese Funktion sollte also auch in das zu entwickelnde System übernommen werden. Eine unerwartet skeptische Resonanz ergaben die eingesetzten Storyboards für die Evaluation aus der dritten Frage. Die für diese Evaluation angefertigten Storyboards in den Fragebögen wurden nur von der Hälfte der befragten Personen als angemessen angesehen. Da diese Funktion jedoch nur optional von UsER angeboten werden soll, änderte dies nicht die bisherigen Konzepte des Systems. Sehr positiv wurde als letztes noch eine Funktion der Übersicht bewertet, die dem Benutzer von UsER im Regiebereich anzeigt, welche Kapitel aktuell geöffnet sind.

4.5.7 Auswahl der Frameworks

Etwa zeitgleich zur Konzepterstellung beschäftigten wir uns mit der Auswahl der technologischen Plattformen für das UsER-System. Um die geplante Kollaboration zwischen Kunde und Entwicklung zu ermöglichen, sollte UsER als Webanwendung realisiert werden, ohne dabei auf die Anmutung einer nativen Anwendung zu verzichten. Im Gegensatz zu den ebenfalls angedachten Frameworks Wicket und Java-Server-Pages erlaubt GWT die Erstellung dynamischer Webanwendungen ohne tiefergehende Kenntnisse von Skriptsprachen. Auch zukünftige Erweiterungen können komplett in Java entwickelt werden. Insgesamt kamen die folgenden Technologien zum Einsatz:

Maven

Maven ist eine Projekt Management Software, die sich um das automatisierte Herunterladen und Aktualisieren von Abhängigkeiten kümmert. Zudem bietet es Möglichkeiten, den Build-Prozess des Projekts leicht konfigurieren und automatisieren zu können.

Hibernate

Für das objektrelationale Mapping der Entitäten aus der Datenbank fiel die Entscheidung im Rahmen des Projektes auf die Verwendung des Frameworks Hibernate. Dieses ermöglicht das Speichern des Zustandes eines Objektes aus der JVM (Java Virtual Machine) in einer relationalen Datenbank.

Hibernate Envers

Hibernate Envers dient dazu, bestimmte Entitäten und ihre Beziehungen zu versionieren. Diese werden im Java Code mittels einer Annotation gekennzeichnet. Das Framework sorgt dann dafür, dass beim Speichern von veränderten Entitäten von diesen eine neue Version gespeichert wird. Ältere Versionen von Entitäten bleiben somit mit Bezug auf ihren Änderungszeitpunkt und den ändernden Benutzern erhalten.

Spring

Spring ist ein Open Source Framework mit dem Ziel, die Entwicklung in Java Projekten zu vereinfachen. Die Komplexität und die unterschiedlichen Konzepte vieler Java EE- und Open Source APIs führen oft zu fehlerhaften Abläufen und langen Projektlaufzeiten. Spring vereinfacht Entwicklern die Benutzung von APIs und stellt ein klar strukturiertes Programmiermodell zur Verfügung. Es übernimmt das Management verschiedener Ressourcen wie z.B. der Datenbankanbindung. Zudem kümmert sich Spring um das Verknüpfen der einzelnen Schichten des Projekts. Mittels Dependency können einzelne Schichten unkompliziert ausgetauscht oder neu implementiert werden.

GWT

Das Google Web Toolkit (GWT) ist ein Entwicklungswerkzeug, um mit Java robuste Web Applikationen zu erstellen. Die entwickelte Applikation wird dabei in eine Server- und eine Clientkomponenten unterteilt. Die Clientkomponente wird zwar ebenfalls in Java entwickelt, wird jedoch von GWT in JavaScript Code kompiliert. Diese läuft somit eigenständig im Browser des Benutzers und tauscht lediglich Daten per Remote Procedure Call (RPC) mit dem Server aus.

GXT

GXT oder auch ExtGWT ist eine Erweiterung von GWT, die es ermöglicht, Rich Internet Applications (RIA) wie mit dem ExtJS Framework zu entwickeln. Dabei wird die Applikation jedoch in Java und nicht in JavaScript entwickelt. Durch das GWT wird der Java Code in entsprechenden JavaScript Code kompiliert. Das GXT Framework bietet wie auch das ExtJS Framework viele verwendbare Widgets wie z.B. Buttons, Nachrichtenfenster oder Tabellen, und Funktionalitäten wie z.B. Animationen, Drag & Drop oder automatisierte Layouts.

4.5.8 Architektur

Die Software-Architektur von UsER entspricht weitestgehend der einer klassischen Software-Architekturaufteilung, wie beispielsweise von Oates (2007) beschrieben. Auf Grund des gewählten Frameworks besitzt UsER noch eine zusätzliche Datentransportschicht zwischen der Präsentationsschicht des Clients und der Serviceschicht des Servers.

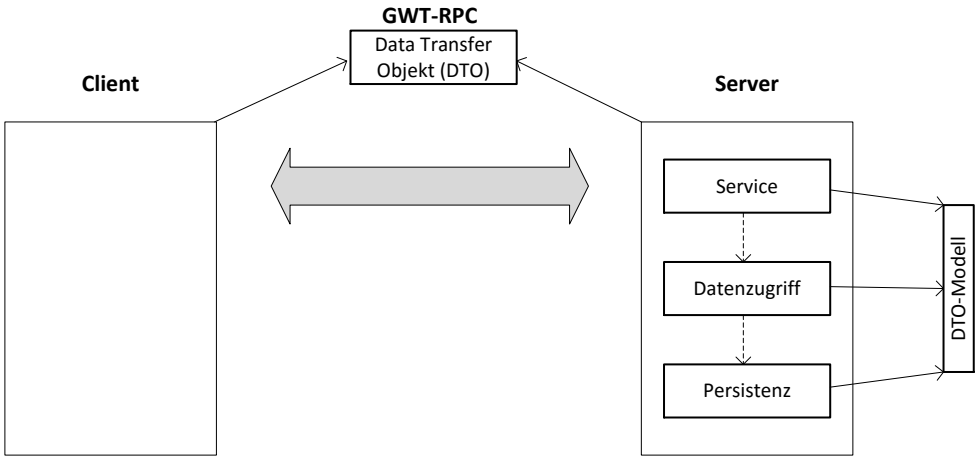


Abbildung 76: Client Server Aufteilung von UsER

Die in UsER erstellten Daten werden in einer MySQL-Datenbank gespeichert. Diese befindet sich auf einem separaten Datenbankserver, der aus Gründen der Übersichtlichkeit in Abbildung 76 nicht mit aufgeführt ist.

Daten, die in der Datenbank gespeichert werden sollen, liegen in UsER als Java-Objekte in der *Persistenz-Schicht*. Die Umwandlung dieser Java-Objekte (DTO-Modell) in das Datenbankschema übernimmt in UsER das Hibernate-Framework. Dieses sorgt für die objektrelationale Abbildung der Java-Objekte in eine relationale Datenbank. Dabei entspricht ein Java-Objekt bzw. eine Klasse einer Tabelle in der Datenbank. Eine Instanz des Objektes ist ein Eintrag bzw. eine Zeile in der Tabelle. Die Spalten der Tabelle (Attributen) entsprechen den Eigenschaften des Java-Objektes. Primitive Datentypen wie Integer oder Strings werden von Hibernate in den jeweiligen Datentyp des Datenbankdialektes umgewandelt. Komplexe Datentypen werden als Relation in der Datenbank gespeichert. Das Hibernate-Framework ermöglicht darüber hinaus den Austausch einer ganzen Reihe von unterstützten Datenbanken wie beispielsweise Oracle oder den Microsoft SQL Server.

Die *Datenzugriff-Schicht* besitzt für jede Klasse, die in der Datenbank gespeichert werden soll ein Data-Access-Objekt (DAO). Die DAOs kümmern sich um alle Datenbankzugriffe wie Erstellen, Lesen, Aktualisieren und Löschen. Oates bezeichnet diese Funktionen auch als CRUD[15]-Operationen. In einer DAO sollte sich keine weitere Geschäftslogik befinden. In UsER werden diese Basisfunktionen durch die Klasse *BaseDao* an alle anderen DAOs vererbt. Dies verhindert mögliche Fehler und hilft darüber hinaus einen redundanten Code zu erstellen.

In der *Service-Schicht* befindet sich die gesamte Geschäftslogik von UsER. Sie verwendet die Objekte aus der DAO-Schicht. Sie kapselt Operationen, die über das Erstellen, Lesen, Aktualisieren und Schreiben hinausgehen. Sie verhindert Code-Duplikate in der Präsentations-Schicht. Auch Funktionen wie beispielsweise das Versenden von e-Mails können von ihr übernommen werden.

Die Kommunikation zwischen Server und Client geschieht in dem gewählten GWT-Framework durch sogenannte *Data Transfer Objekte* (DTO). Ein solches Objekt beinhaltet Informationen, die vom Client zum Server geschickt werden sollen oder von diesem geholt werden. Mit Hilfe des Remote Procedure Call (RPC) können diese aus Programmsicht als Java-Objekte verschickt werden. RPC sorgt für die Serialisierung der Objekte über das XMLHttpRequest. Sie können in serialisierter Form hin und her geschickt werden. So schickt GWT die Parameter einer Funktion, die auf Clientseite aufgerufen wird, durch einen RPC in serialisierter Form an den Server. Auf Serverseite werden die Parameter wieder deserialisiert und von der entsprechenden Funktion dann ausgeführt. Der Rückgabewert wird anschließend über Response an den Client zurückgeschickt.

Die wahre Stärke des Google Web Toolkit (GWT) kommt auf *Client-Seite* zum Tragen. Hier bietet GWT eine ganze Reihe an grafischen Visualisierungsmöglichkeiten, die sonst nur native Frameworks wie Swing oder SWT[16]

[15] Create, Read, Update, Delete

[16] Standard Widget Toolkit

unterstützen. GWT ermöglicht die Programmierung der Präsentationsschicht in Java. Die Umwandlung der einzelnen Komponenten in HTML, CSS und JavaScript wird von GWT übernommen. In UsER ist noch eine weitere Bibliothek der Firma Sencha integriert. Diese ermöglichte UsER in Bezug auf Erscheinung und Verhalten wie eine Desktopanwendung erscheinen zu lassen.

4.5.9 Datenmodell

Das Datenmodell aus Abbildung 77 ist in fünf verschiedene Bereiche aufgeteilt. Es handelt sich hierbei um das Datenmodell vom 01.06.2011, welches sukzessiv in den weiteren Iterationen erweitert wurde. Die im grauen Rechteck zusammengefassten Entitäten unterliegen einer ständigen Versionierung der darin enthaltenen Daten. So können in der zu entwickelnden Anwendung Änderungen eines Benutzers chronologisch rückverfolgt werden. Die Versionierung der Entitäten wird durch das Hibernate-Envers Framework innerhalb des Systems realisiert. Die fünf Bereiche unterteilen sich in die Benutzerverwaltung (lila), die modulübergreifenden Kernanwendungen (rot) sowie einige Entity-Relationship-Beziehungen der Module Arbeitssystemanalyse (gelb), des Szenarien-Moduls (grün) sowie ein Modul für die Anforderungsanalyse.

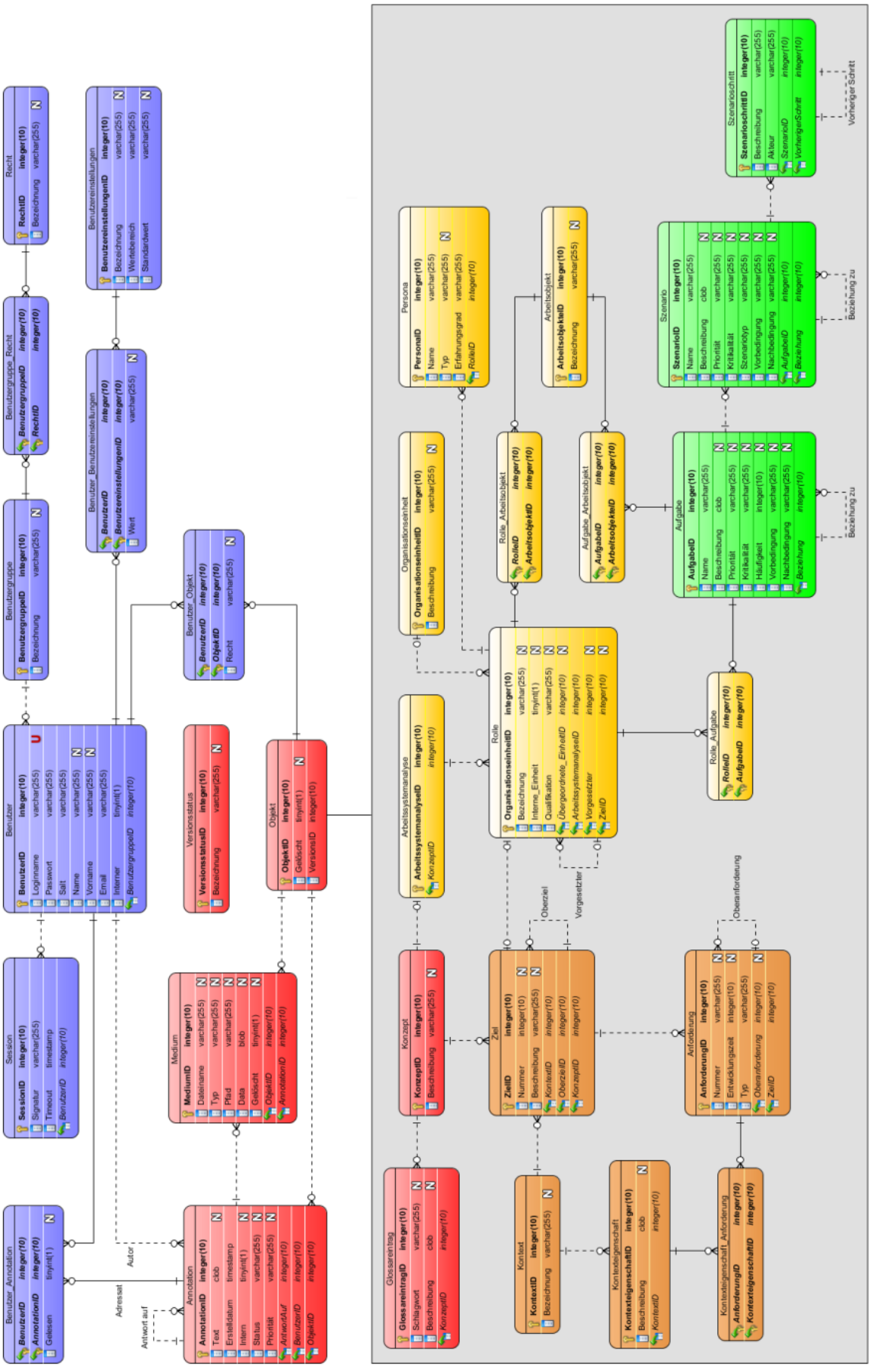


Abbildung 77: Datenmodell der Kernkomponente von UsER (Stand 01.06.2011)

5 Realisierung des UE-Werkzeugs UsER

Aufbauend auf dem oben beschriebenem Kernsystem und dem zuvor vorgestellten idealisierten Entwicklungsprozess sollen im Folgenden Werkzeugmodule für das UsER-System entwickelt werden, die den idealisierten Entwicklungsprozess unterstützen. Ein Großteil der dabei entstandenen Module wurde im Rahmen der von mir betreuten studentischen Abschlussarbeiten entwickelt. Insgesamt kamen im Rahmen dieses Projektes 15 Abschlussarbeiten zustande. Zudem arbeiteten eine Reihe wissenschaftlicher Hilfskräfte an der Entstehung der im Folgenden beschriebenen UE-Werkzeugmodule. In wöchentlich stattfindenden Teammeetings wurden Gestaltungslösungen bewertet, Probleme diskutiert und die sich daraus ergebenden Anforderungen abgeleitet. Auf diese Weise durchlief jedes der entstandenen Module eine Reihe von Iterationen. Zu Beginn des Projektes wurde die gesamte Verwaltung und Dokumentation mit unterschiedlichen Anwendungen wie Microsoft Word für die Dokumentation oder der freien Kollaborationssoftware Retrospectiva für die Ticketverwaltung verwendet. Für die Erstellung von Mockups setzten wir einfache Grafikprogramme wie Microsoft Paint oder Adobe Photoshop ein. Im Zeitverlauf dieser Arbeit konnte die gesamte Projektverwaltung mit dem entstandenen System UsER organisiert werden. Ziel war die Erstellung eines in sich geschlossenen Systems mit beliebig kombinierbaren UE-Werkzeugen, die es Entwicklungsunternehmen erlaubt, flexibel die Funktionen zur benutzerzentrierten strukturierten Erhebung, Ablage und Organisation von Informationen für die Entwicklung interaktiver Anwendungssysteme zu nutzen. Abbildung 78 veranschaulicht den in Kapitel 4.4.11 vorgestellten idealisierten Entwicklungsprozess mit den angedachten UE-Methoden innerhalb der jeweiligen Prozessschritte. Die blauen Symbole links neben den Methoden symbolisieren die jeweiligen UE-Werkzeuge, die im Rahmen dieser Arbeit entstanden sind (Kammler, Roenspieß & Herczeg, 2012):

 Benutzeranalyse	In diesem Modul können über den kompletten Lebenszyklus wiederverwendbare abstrakte Benutzerklassen, Stereotypen und konkrete Personas modelliert werden. Die dabei erhobenen Benutzerziele (Cooper et al., 2007) gehen in die Liste der Requirements ein.
 Aufgabenanalyse	Dieses Modul ermöglicht die Dekomposition externer organisatorischer Aufgaben einer Rolle oder Stelle in interne Aufgaben und die Zuordnung von Attributen wie Häufigkeit, Priorität, Kritikalität, etc.
 Organisationsanalyse	Für betrieblich orientierte Softwareentwicklungen bietet dieses Modul eine hierarchische Darstellung der Aufbauorganisation in Form von Organisationseinheiten und Stellen. Diese können bei Bedarf durch die Beschreibung unterschiedlicher Rollen und dazugehöriger Aufgaben detailliert werden.
 Anwendungsfall (Szenarien)	Je nach Projektfortschritt können Anwendungsfälle abstrakt in Form von Szenarien mit Bildern - z.B. aus der integrierten Mockup-Komponente - beschrieben werden. Durch ein Klassifikationsschema können die Szenarien nach Systemkomponenten und Benutzerzielen organisiert werden. Das Modul ermöglicht darüber hinaus ein mit den Anforderungen gekoppeltes Bewertungsverfahren als Unterstützung des Erfüllungsgrades von Anforderungen und unterstützt damit den iterativen Verfeinerungsprozess von Lösungsansätzen aktiv.
 Anforderungen	Die Erfassung und Bearbeitung von Anforderungen ist aus jedem Modul heraus möglich. Dieses Modul unterstützt den gesamten Lebenszyklus einer Anforderung von der Anforderungsaufnahme bis zur Implementierung. Eine Exportfunktion kann zur Integration in IDEs verwendet werden.
 Prozessbeschreibung	Ähnlich den aus der UML bekannten Aktivitätsdiagrammen unterstützt dieses Modul den BPMN-Standard zur strukturellen Beschreibung von Prozessen und Anwendungsfällen.

	Arbeitsobjektanalyse	In einem (Arbeits-) Prozess vorkommende (Arbeits-)Objekte können hier angezeigt und einem Projekt zugeordnet werden.
	Text	Dieses Modul erlaubt, innerhalb eines Projekts beliebige Rich-Text-Dokumente zu ergänzen und zu vernetzen.
	Evaluation	Sie ermöglicht die Neuerstellung, Wiederverwendung und Auswertung digitaler Evaluationsbögen.

Ein mögliches Anwendungsszenario dieser Module wäre dann der entwickelte idealisierte Entwicklungsprozess, wie er in Abbildung 78 dargestellt ist.

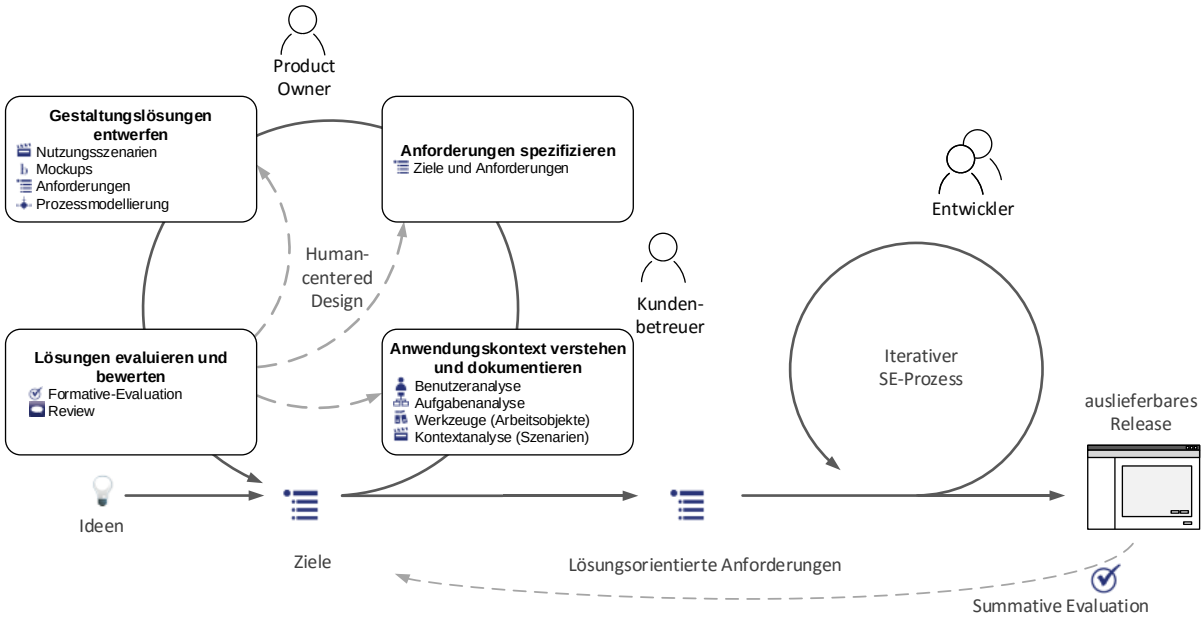


Abbildung 78: Idealisierter UE-Prozess mit UE-Werkzeugen

Auf der linken Seite der Abbildung ist schematisch das UE in Form des HCD-Prozesses nach der DIN EN ISO 9241-210, aus Kapitel 2.2.1, zu sehen. Auf der rechten Seite ist das eigentliche SE dargestellt, angelehnt an das iterative Vorgehen der agilen Softwareentwicklung (siehe Kapitel 3.1.5). Die Anforderungen (bzw. Ziele und lösungsorientierten Anforderungen) bilden die Schnittstelle zwischen den beiden Verfahren. Sie sind das Ergebnis des UE und können von der anschließenden Softwareentwicklungsphase als Ausgangspunkt für einen Entwicklungszyklus genutzt werden. Ziel der SE-Phase sollte, wie im agilen Prozessrahmenwerk (Beck & Grenning, 2001) beschrieben, ein potentiell auslieferbares Produktinkrement sein. Auf diese Weise ist der Kunde in der Lage, anhand jedes dieser Produktinkremente den Fortschritt zu beurteilen und so frühzeitig auf mögliche Fehlentwicklungen Einfluss zu nehmen.

Jede Gestaltungsaktivität des HCD-Prozesses wird durch eine Reihe unterschiedlicher Methodenmodule unterstützt, die als blaue Symbole in den jeweiligen Prozessschritten angedeutet sind. Sie beschreiben die Werkzeugmodule des UsER-Systems. Alle mit diesen Modulen erhobenen Informationen können wiederum mit Informationen anderer Module verknüpft werden, wodurch ein Netz aus semantischen Verknüpfungen entsteht. Dies ermöglicht eine flexible Arbeitsteilung an unterschiedlichen, entwicklungsrelevanten Informationen. Das Entwicklungsprojekt selbst wird innerhalb von UsER in einem klassischen linearen Arbeitsdokument verwaltet, wie in der ISO 9241-210 beschrieben. Jedes Kapitel dieses Entwicklungsdokumentes bietet die Funktionalität eines ausgewählten Moduls. Dies gewährleistet die Abbildbarkeit auf viele gängige Entwicklungsprozesse. Aufgrund der Entscheidung, UsER als Webapplikation zu realisieren und wegen des einfachen Rechtemanagements, können Projekte von Entwicklungsfirmen sowohl intern als auch extern genutzt werden. Dies ermöglicht die Einbindung von Kunden in einen Entwicklungsprozess. Im Folgenden soll die Funktionsweise des UsER-Systems erläutert

werden, indem der oben aufgeführte Prozess mit den verschiedenen Modulen des UsER-Systems durchlaufen wird.

5.1 Planung des Human-centered Design-Prozesses

Entsprechend der ISO 9241-210 sollen den einzelnen zusätzlichen Aktivitäten des HCD Prozesses in dieser Phase des Entwicklungsprozesses feste Zeiten und Ressourcen zugeordnet werden. Insbesondere die Integration der zusätzlichen Aktivitäten in den SE-Prozess sowie die Iterationen und der vermehrte Informationsaustausch mit den beteiligten Stakeholdern sind einzuplanen. Die folgenden Unterkapitel zeigen, wie das UsER-System das Projekt bereits in dieser Phase unterstützen kann.

5.1.1 Projektverwaltung

Bei der Entwicklung des Systems wurde nach einer Möglichkeit gesucht, beliebig strukturierte Entwicklungsprozesse methodisch und werkzeugseitig unterstützen zu können. Dazu werden Entwicklungsvorhaben in UsER als Projekte verwaltet. In der Projektübersicht ist eine tabellarische Aufstellung aller in einem Unternehmen vorkommenden Projekte aufgeführt. Durch das Markieren eines Projektes wird ein Detailbereich (unteres Viertel in Abbildung 79) eingeblendet, der die Manipulation der wichtigsten Daten wie den Namen des Projektes, sowie eine Beschreibung und den Namen der Kundenfirma ermöglicht. Zusätzliche Informationen zum Zeitpunkt der Erstellung und zu dem Autor werden automatisch im Hintergrund gespeichert. Zudem können hier Teilnehmer eines Projektes verwaltet und benutzerspezifische Kommentare eingesehen werden.

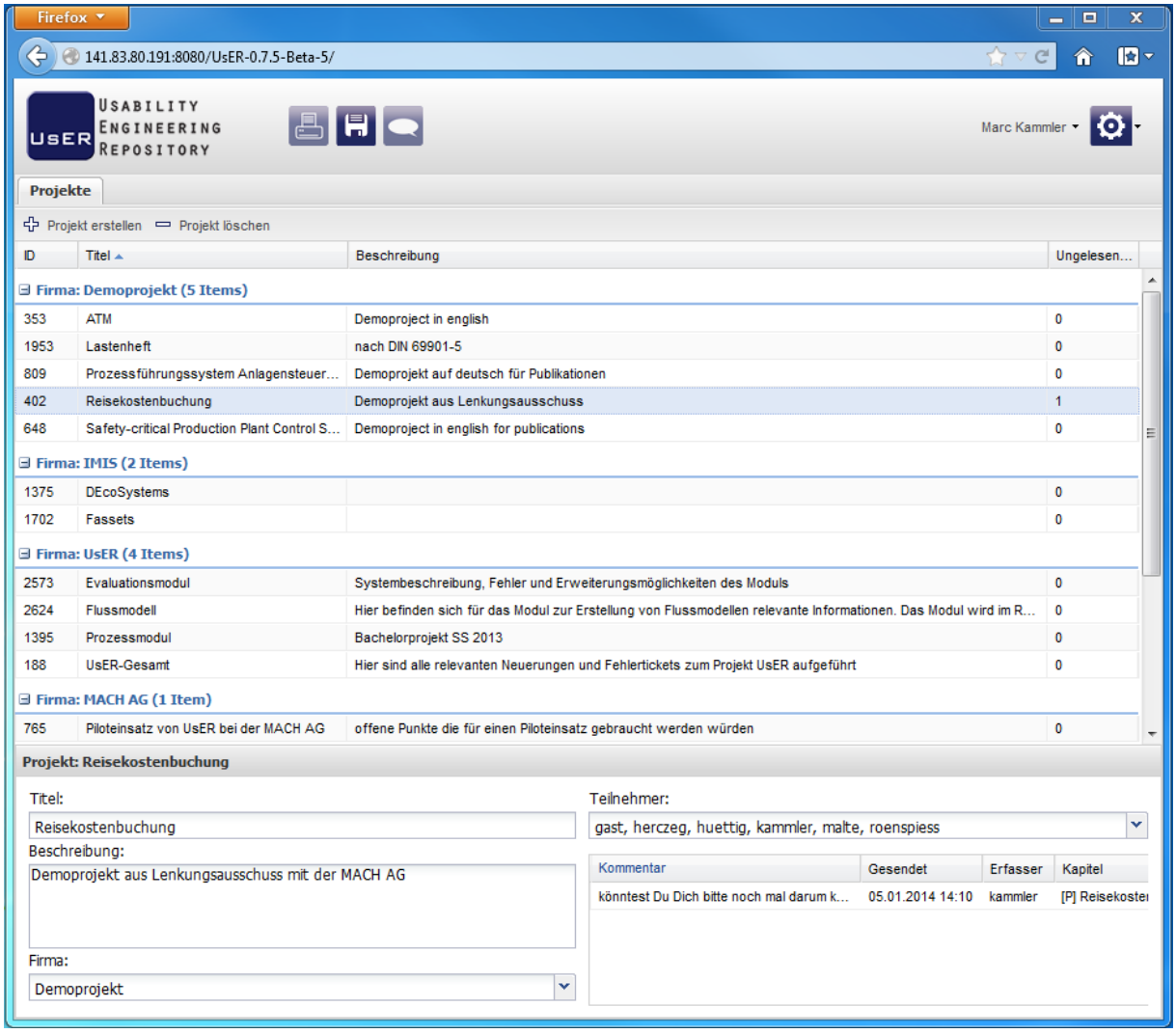


Abbildung 79: Projektübersicht von UsER

Über ein einfaches Rechtemanagement, können im System registrierte Benutzer einem Projekt zugeordnet werden. Anhand des Dropdown-Menüs *Teilnehmer* (oben rechts im Detailbereich) können Teilnehmer dem ausgewählten Projekt zugeordnet oder wieder entfernt werden. Die auf diese Weise ausgewählten Teilnehmer (siehe Abbildung 80) werden durch ein Komma getrennt anschließend in dem Menü angezeigt. Die Projekte sind ausschließlich für eingetragene Teilnehmer sichtbar.

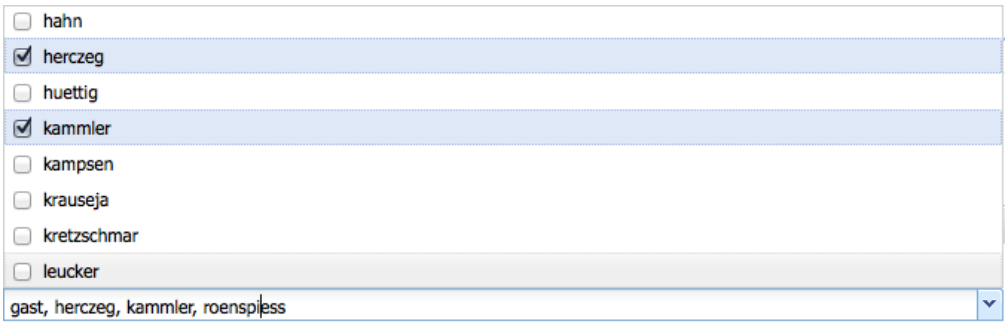


Abbildung 80: Auswahl der zu einem Projekt gehörigen Teilnehmer

Um den vermehrten Informationsaustausch zwischen den Projektteilnehmern (Stakeholdern) zu erleichtern, besitzt das UsER-System eine Kommentarfunktion. Diese wird in den folgenden Kapiteln eingehender beschrieben. In der Projektansicht werden diese Nachrichten bereits als Vorschau angezeigt. Im Detailbereich unten rechts werden alle an einen Benutzer gerichteten Nachrichten in einer Tabelle aufgeführt (siehe Abbildung 81).

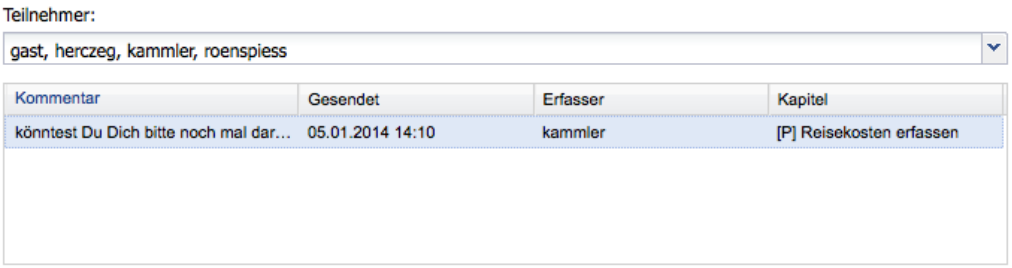


Abbildung 81: Nachrichten beteiligter Stakeholder

Durch einen Doppelklick auf ein Projekt wird dieses, wie in Kapitel 4.5.5 beschrieben, in einem neuen Karteireiter geöffnet.

5.1.2 Projektansicht

Projekte können als Dokument verstanden werden, wobei jedes Dokument mit einer beliebigen Kapitelstruktur aufgebaut werden kann (siehe Abbildung 82). Darüber hinaus können Projekte jedoch auch als *Kapitel-Sammlungen* bzw. *Templates* gespeichert werden, damit eine etablierte Struktur, beispielsweise für ein Lastenheft oder ein Pflichtenheft, in mehreren Entwicklungsvorhaben verwendet werden kann. Um eine hohe Flexibilität zu erreichen, wurde das System modular aufgebaut. Dies ermöglicht eine grobe Festlegung der durchzuführenden Aktivitäten, wie es in der ISO 9241-210 gefordert wird. Die technische Umsetzung dieses modularen Aufbaus entstand im Rahmen einer Diplomarbeit (Wollesen, 2012).

Jedes Modul ist als eigenständige Methoden- oder Funktionskomponente für die durchzuführende Gestaltungsaktivität des HCD zu verstehen. Die mit den Modulen erhobenen Informationen, wie beispielsweise spätere Benutzer oder die von ihnen zu erledigenden Aufgaben, werden innerhalb eines Projektes als eindeutige Entitäten behandelt. UsER ermöglicht eine flexible Vernetzung dieser Entitäten. Diese Art der Verknüpfung kann innerhalb eines Dokumentes modulübergreifend durchgeführt werden.

Ein Entwicklungs-Vorhaben (Projekt) wird für die bessere Übersichtlichkeit jedoch als lineares Dokument dargestellt (siehe Abbildung 82 links). Die einzelnen Kapitel dieses Dokumentes entsprechen dabei den jeweiligen Modulen, die in beliebiger Weise – via Drag & Drop – in dem Dokument angeordnet werden können. Dies ermöglicht beispielsweise die Strukturierung des Dokuments in Form eines Lasten- oder Pflichtenhefts oder jeder

anderen Struktur, die eine Organisation nutzen möchte. Die Struktur eines solchen Dokuments kann auch als Template gespeichert werden, um sie wiederverwenden zu können.

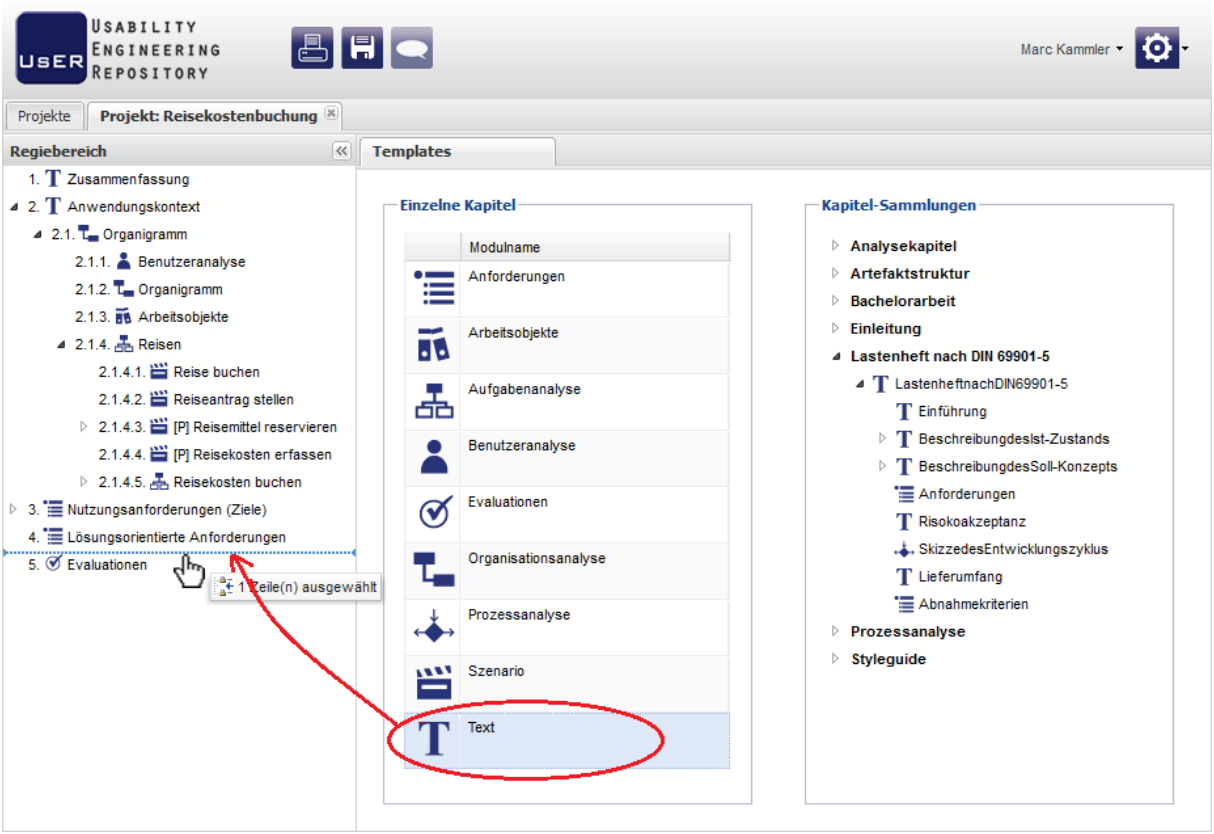


Abbildung 82: Beliebig strukturierbares Spezifikationsdokument

Die bisher realisierten Module unterstützen die Analyse, das Design, die Evaluation wie auch die Anforderungsverwaltung. Das System kann jedoch beliebig um weitere Module ergänzt werden. Die Schnittstelle zur klassischen Software-Entwicklung in der Lösungsansätze in die Entwicklung überführt werden können, wird über ein Modul zur Anforderungsverwaltung ermöglicht (Paelke & Nebe, 2008). Im Gegensatz zu bisherigen Anforderungsmanagement-Tools wie Redmine oder Trac ermöglicht UsER jederzeit die Rückverfolgung aller für die Entscheidung relevanten Informationen aus Analyse, Design oder Evaluation. So kann ein Entwickler ausgehend von der zu implementierenden Anforderung jederzeit die Ergebnisse der jeweiligen Entwicklungsentscheidung einsehen.

5.2 Verstehen und Dokumentieren des Anwendungskontextes

Nach der Empfehlung der ISO 9241-210 soll in diesem Schritt der Nutzungskontext verstanden und in Form eines Arbeitsdokumentes dokumentiert werden. Dies ist somit die Phase der Entwicklung, in der das zu lösende Problem verstanden und dokumentiert werden sollte. Dazu sind Informationen zu den Benutzern und anderen Interessensgruppen, den Zielen und Arbeitsaufgaben der Benutzer als auch zu den Materialien notwendig, die zur Erfüllung einer Arbeitsaufgabe dienen. In frühen Phasen des Entwicklungsvorhabens können diese Dokumentationen noch nicht erschöpfend sein, sondern erst in weiteren Iterationen fortlaufend verfeinert werden (ISO 9241-210:2011).

5.2.1 Benutzer-Analyse

Für das Verständnis des Nutzungskontextes, in der eine zu entwickelnde Software eingesetzt werden soll, könnte beispielsweise mit der Analyse der Stakeholder begonnen werden. Das dafür vorgesehene Benutzeranalyse-Modul ermöglicht das Anlegen von Benutzerklassen (siehe Abbildung 83) in Form von organisatorischen Rollen oder Zielgruppen. In dem Beispiel aus Abbildung 83 sind die exemplarischen Rollen „Reisender“, „Entscheider“, „Reisesachbearbeiter“ und „Buchhalter“ angelegt worden. Unterhalb der jeweiligen Rollen können dann einzelne Benutzer aufgeführt werden, welche die Rolle weiter konkretisieren. Abbildung 83 zeigt die tabellarische Übersicht

der in einem Projekt vorkommenden Rollen und Benutzer. Neben dem fiktiven Namen werden in der Übersicht vor allem Informationen für die Verwaltung und Erstellung der jeweiligen Benutzer aufgeführt. Es wird sofort ersichtlich, ob sich eine Benutzerbeschreibung noch in der Bearbeitung befindet oder schon freigegeben ist. Auch der jeweils Verantwortliche ist in der Spalte *Pate* schnell ersichtlich. Durch das Selektieren eines Benutzers in der tabellarischen Übersicht werden – wie auch in allen anderen Modulen – weitere Detailinformationen zu dem jeweiligen Benutzer im unteren Detailbereich Anwendung angezeigt.

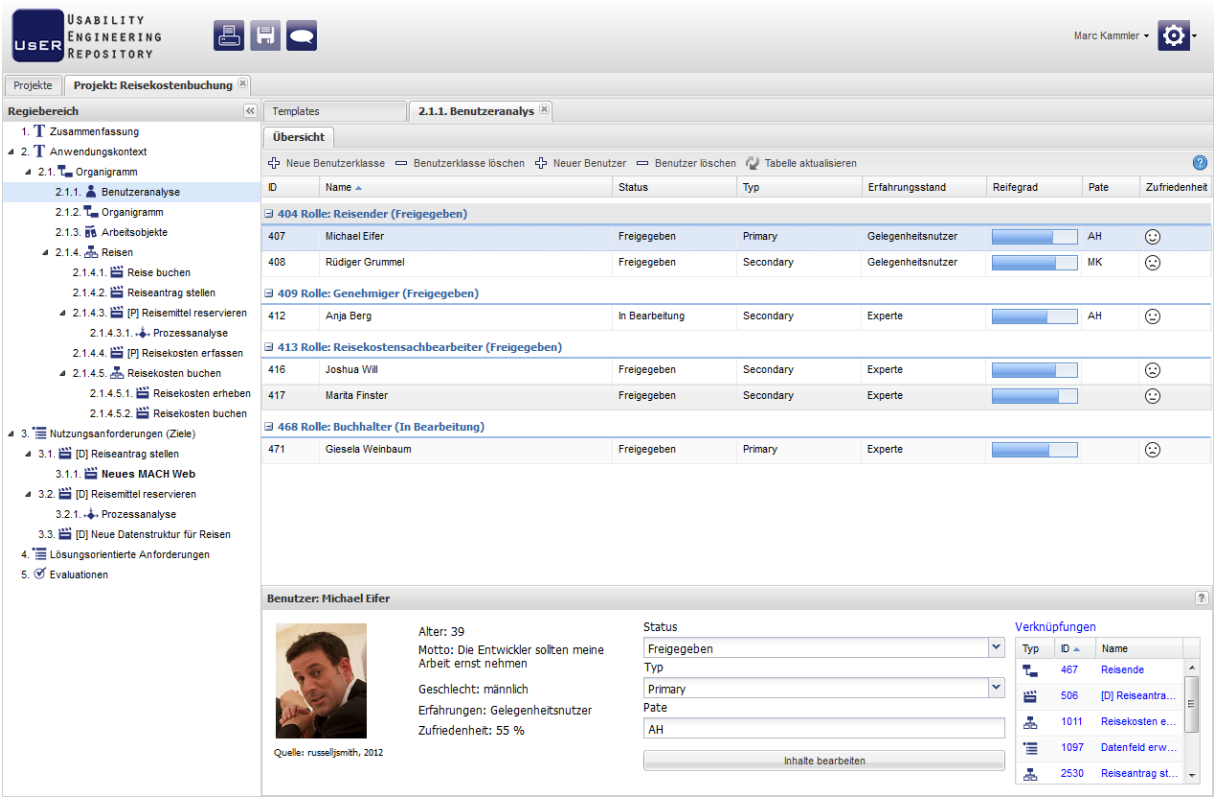


Abbildung 83: Übersicht der Benutzer und ihrer Rollen innerhalb eines Projektes

Einmal zu einer Benutzerklasse angelegte Informationen können immer wieder dazu verwendet werden, detailreichere Beschreibungen der Benutzer anzulegen. Alle Informationen aus der Benutzerklasse werden dabei vererbt, können dort jedoch dann auch weiter manipuliert oder bearbeitet werden. Um einer Rolle – wie in dem unteren Beispiel dem „Entscheider“ – einen weiteren Benutzer hinzuzufügen, wird über das Kontextmenü die Funktion *Neuer Benutzer* aufgerufen (siehe Abbildung 84).

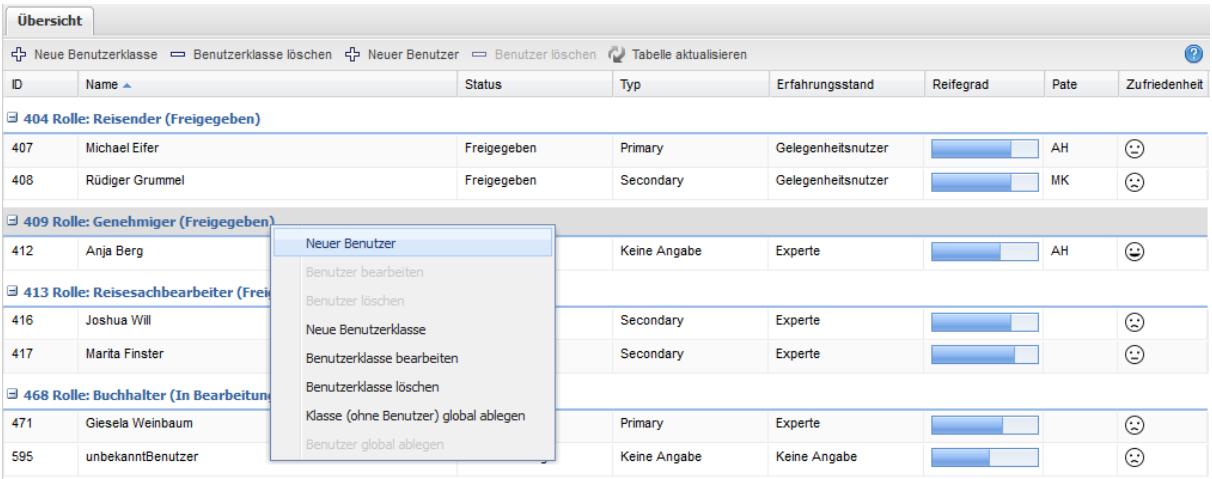


Abbildung 84: Einer Rolle einen neuen Benutzer hinzufügen

Durch die Möglichkeit der Vererbung von Informationen, könnte es beispielsweise von der Größe eines Entwicklungsvorhabens (Pruitt & Grudin, 2003) abhängig gemacht werden, wie detailliert die Benutzerbeschreibung ausfällt. Um jedoch die Detaillierung einer Benutzerbeschreibung zu forcieren, wurde ein Fortschrittsbalken eingebaut, der den Reifegrad des jeweiligen Benutzers anzeigt. Dieser zeigt den Verantwortlichen an, wie viele Informationen noch fehlen, um den Mindestanforderungen einer Persona zu entsprechen.

Für die Erstellung einer Benutzerbeschreibung ist es wichtig, reale Eigenschaften zu verwenden (Cooper et al., 2007). Aus diesem Grund ermöglicht das Modul die Erstellung sogenannter Verhaltensvariablen. Dies sind für das Entwicklungsvorhaben wichtige Eigenschaften der Benutzer, wie beispielsweise die Information, ob ein Anwenderkreis „einfach und übersichtlich“ strukturierte Anwendungen bevorzugt, oder wie vertraut der Umgang mit dem Internet z.B. für Web-basierte Softwarelösungen ist (siehe Abbildung 85). Die Antworten der Benutzer lassen sich dann auf einer bidirektionalen Skala verteilen, wodurch sich Trends bzw. Cluster abzeichnen, die dazu genutzt werden können, Eigenschaften der fiktiven Benutzer auf Grundlage realer Daten zu beschreiben. Es ist geplant diese Befragung zukünftig halbautomatisch durchzuführen.

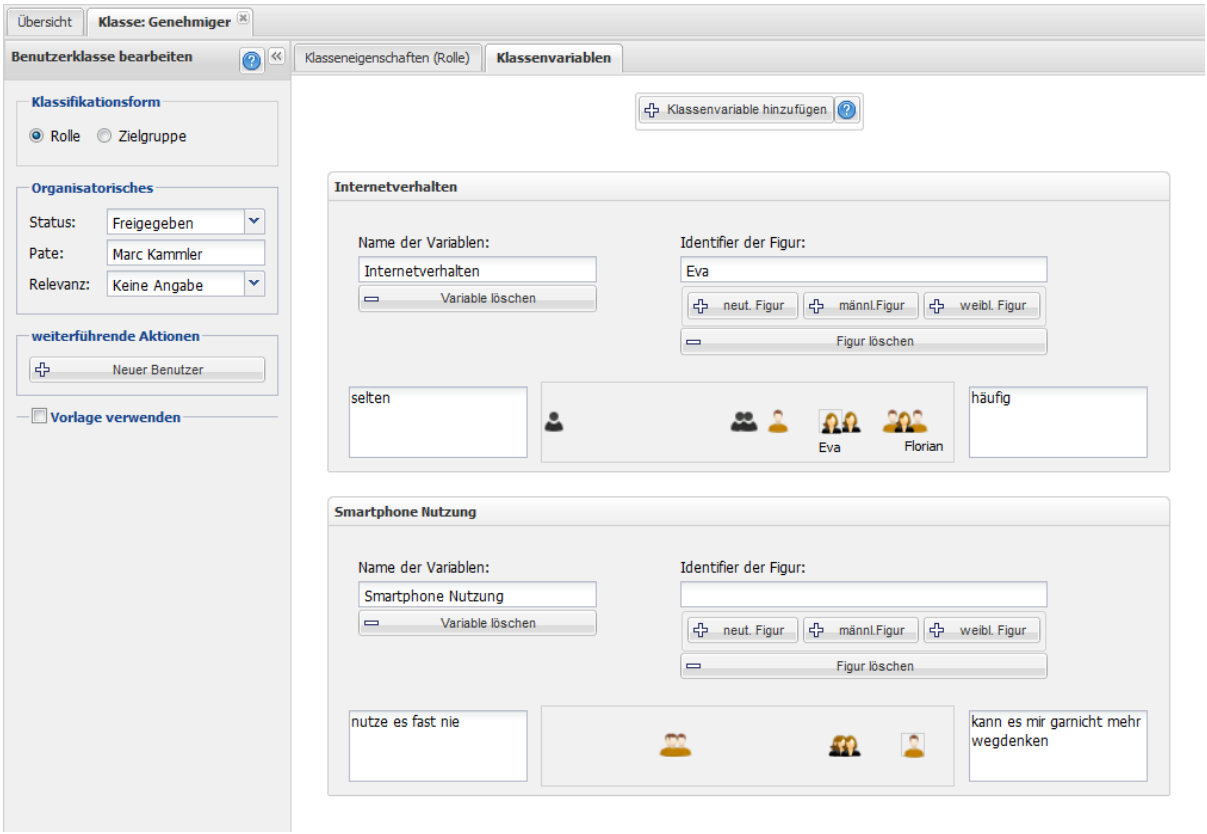


Abbildung 85: Persona Eigenschaften durch Verhaltensvariablen dargestellt

Benutzerbeschreibungen können projektspezifisch bearbeitet und angepasst werden. Alternativ können diese auch projektübergreifend zugänglich gemacht werden, um sie in mehreren Entwicklungsvorhaben (Projekten) verwenden zu können. Diese Funktion ist in der tabellarischen Übersicht über das Kontextmenü erreichbar.

Ebenfalls durch Cooper inspiriert ist die Idee, persönliche und fachliche Ziele der Benutzer mit aufzunehmen, damit diese dann auch während der Spezifikation entsprechend berücksichtigt werden. Diese werden automatisch von UsER in dem Modul zur Verwaltung der Anforderungen angezeigt (siehe Kapitel 5.3.1). Für die Entwicklung einer gebrauchstauglichen Software ist es darüber hinaus noch unabdingbar, die Aufgaben der Benutzer zu ermitteln (Constantine & Lockwood, 1999; Herczeg, 2009). In der Ansicht zur detaillierten Benutzerbeschreibung können zu diesem Zweck Aufgaben auf einem abstrakten Beschreibungslevel, im betrieblichen Umfeld auch als „externe Aufgaben“ (Herczeg, 2009) bezeichnet, aufgenommen werden. Um sowohl Ziele, Anforderungen, Aufgaben als auch weitere relevante Informationen mit einem Benutzer verknüpfen zu können, kann über die Funktion *Verknüpfungen* jedes beliebige Element eines Projektes mit dem jeweiligen Benutzer verbunden werden. Diese werden anschließend in Form einer Liste (siehe Abbildung 86) angezeigt.

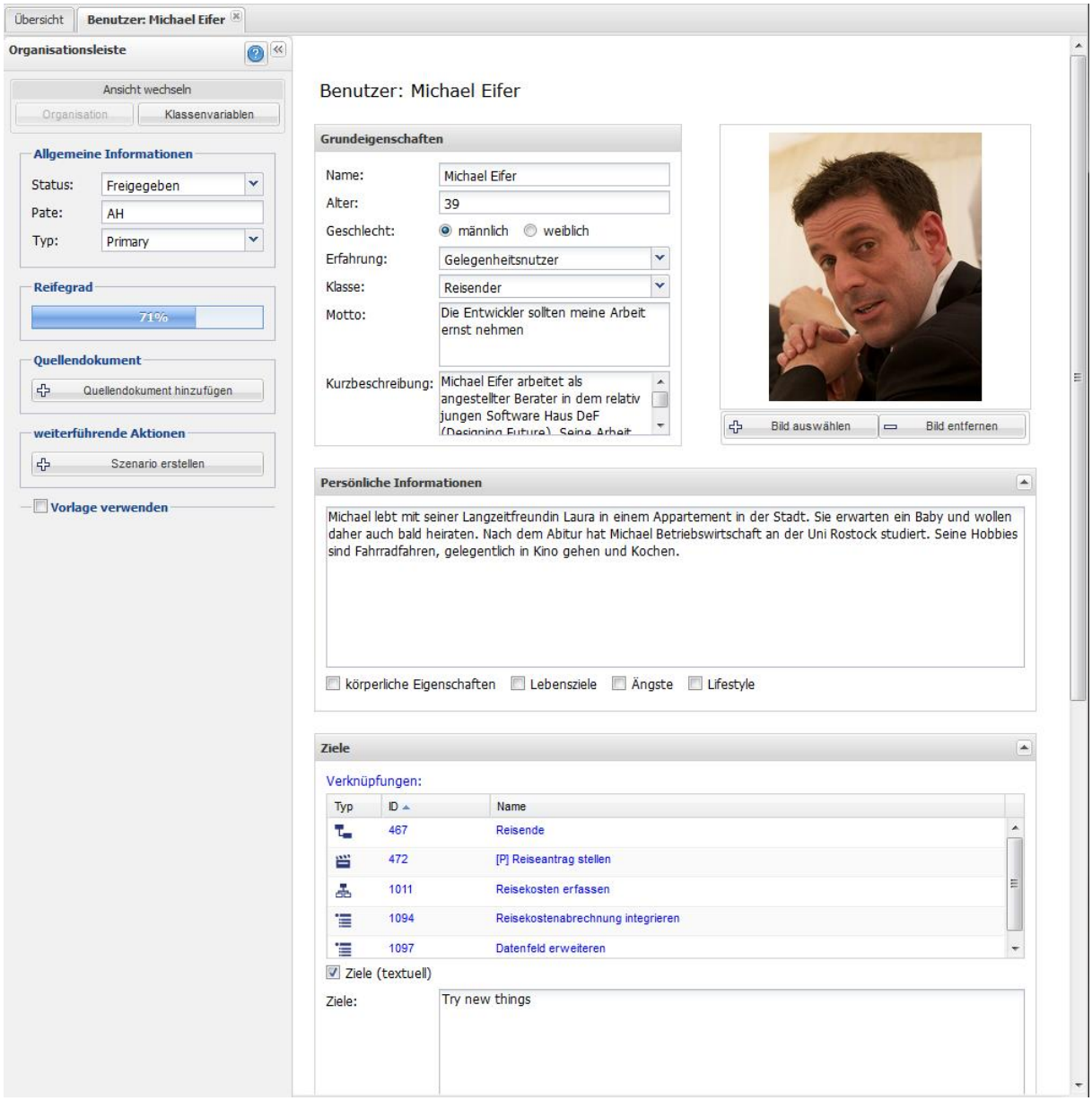


Abbildung 86: Ansicht für fiktive Benutzerbeschreibung in Form einer Persona

Wie bereits erwähnt, sollen Benutzerprofile möglichst ausführlich ausgedacht werden. Der bereits erwähnte Fortschrittbalken *Reifegrad* wird dadurch beeinflusst, wie viele der einzelnen Beschreibungsfelder auf der Profilseite (siehe Abbildung 86) ausgefüllt sind. Um die Benutzerbeschreibungen während des Entwicklungsprozesses stärker zu berücksichtigen, wurde eine weitere Funktionalität in diesem Modul implementiert. Sie soll eine Art Indikator für die Zufriedenheit des beschriebenen Benutzers symbolisieren. In Form von unterschiedlich stark lächelnden „Smilies“ wird angezeigt, wie viele der mit einer Benutzerbeschreibung verknüpften Anforderungen bei der Entwicklung bereits berücksichtigt wurden. In dem Beispiel aus Abbildung 86 wurden beispielsweise zwei Anforderungen „Reisekostenabrechnung integrieren“ und „Datenfeld erweitern“ mit dem Benutzerprofil von Michael Eifer verknüpft. Wird im späteren Verlauf der Entwicklung der Status dieser Anforderungen beispielsweise in den Status „in Bearbeitung“ geändert, so wirkt sich dies positiv auf die Zufriedenheit dieses Benutzers aus. Sollte sich der Status auf „Verschoben“ oder sogar „Abgelehnt“ ändern, würde sich dies negativ auf die Zufriedenheit auswirken. Der Status einer Anforderung lässt sich ausschließlich in dem Modul für die Anforderungsverwaltung ändern.

Eine formative Evaluation des Moduls (siehe Abbildung 87) mit sechs Probanden, die nach Fertigstellung des Moduls verschiedene Anwendungsfälle wie beispielsweise das Erstellen einer Benutzerklasse und Persona oder die Verknüpfung dieser in einem Szenario ergaben eine überwiegend positive Resonanz (Hüttig, 2012).

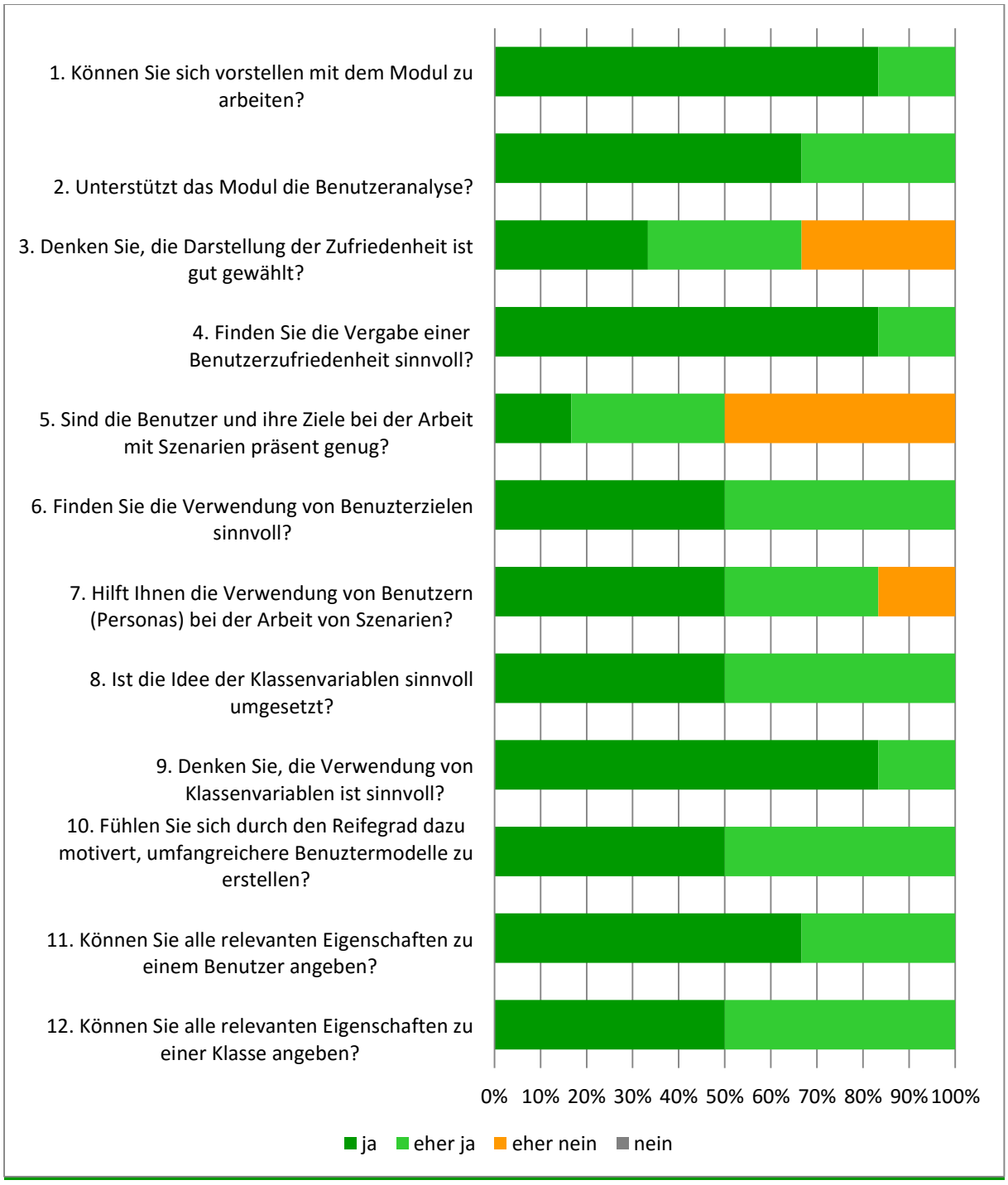


Abbildung 87: Evaluationsergebnisse Benutzeranalysemodul

So konnten sich alle Probanden vorstellen, mit dem Modul zu arbeiten oder fanden dieses hilfreich bei der Erstellung einer Benutzeranalyse (siehe Frage 1-2). Das Ergebnis der dritten Frage bzgl. der Darstellung der Zufriedenheit hatte zur Folge, dass diese nach der Befragung noch einmal überarbeitet wurde. Dass eine Zufriedenheit angeführt wird, hielten die meisten für sinnvoll oder sogar innovativ. Es wurde jedoch angemerkt, dass die Darstellung zu wenig präsent sei. Daher wurde die Übersichtstabelle aus Abbildung 84 dahingehend überarbeitet, dass die zuvor als Prozentzahl dargestellte Zufriedenheit auch hier als Smiley angezeigt wird. Für weitere Überarbeitungen könnte über einen Lösungsansatz nachgedacht werden, der die Zufriedenheit als Leitinformation im

gesamten Projekt visualisiert. Dies würde sich möglicherweise auch positiv auf die Antworten der fünften Frage auswirken. Hier fanden die Probanden die mit den Benutzerbeschreibungen verknüpften Anforderungen bzw. Ziele bei der Arbeit mit Szenarien zu wenig präsent. Dieser Umstand könnte auch durch die Verwendung von Persona bei der Arbeit mit Szenarien verbessern (siehe Frage 7). Die Umsetzung und der Sinn der Verhaltensvariablen (in der Evaluation als Klassenvariable bezeichnet) wurden als hilfreich empfunden (siehe Frage 8-9), obwohl die Idee nicht auf Anhieb für alle verständlich war. Auch der Reifegrad einer Benutzeranalyse in Form des Fortschrittbalkens wurde von den Testpersonen als motivierend bewertet (siehe Frage 10).

5.2.2 Aufgaben-Analyse

Je mehr Wissen über die Aufgaben eines Benutzers - insbesondere einer betrieblichen Rolle - bekannt sind und auch während der Designphase zur Definition der Systemeigenschaften und Merkmale verwendet werden kann, desto höher der Grad der Akzeptanz und Zufriedenheit der Benutzer (Atwood, Bomsdorf & Szwillus, 1999). In UsER wurden für die Analyse von Aufgaben bisher zwei verschiedene Module entwickelt. Je nach Entwicklungsfortschritt sollten Aufgaben unterschiedlich genau beschrieben werden können (Pohl, 2008). Zu Anfang ist es vor allem wichtig, alle entwicklungsrelevanten Aufgaben der späteren Benutzer aufzunehmen und grob strukturieren zu können. Zu diesem Zweck eignen sich vor allem Baumdarstellungen mit den jeweiligen hierarchischen Abhängigkeiten. Für die Spezifizierung eines detaillierten Konzeptes mit einem genauen Programmablauf ist die genaue Abfolge der jeweiligen Aktivitäten und möglicher Alternativen wichtig.

Das HTA-Modul

Das erste Modul ermöglicht die hierarchische Darstellung von Aufgaben. In der ersten Iterationen wurde vor allem die Idee der „Hierarchischen Aufgabenanalyse“ nach Annett und Duncan (1967) verfolgt. Durch eine Reihe von funktionalen Erweiterungen werden inzwischen jedoch auch die übrigen in Kapitel 2.3.1.2.1 beschrieben Aufgabenmodelle unterstützt.

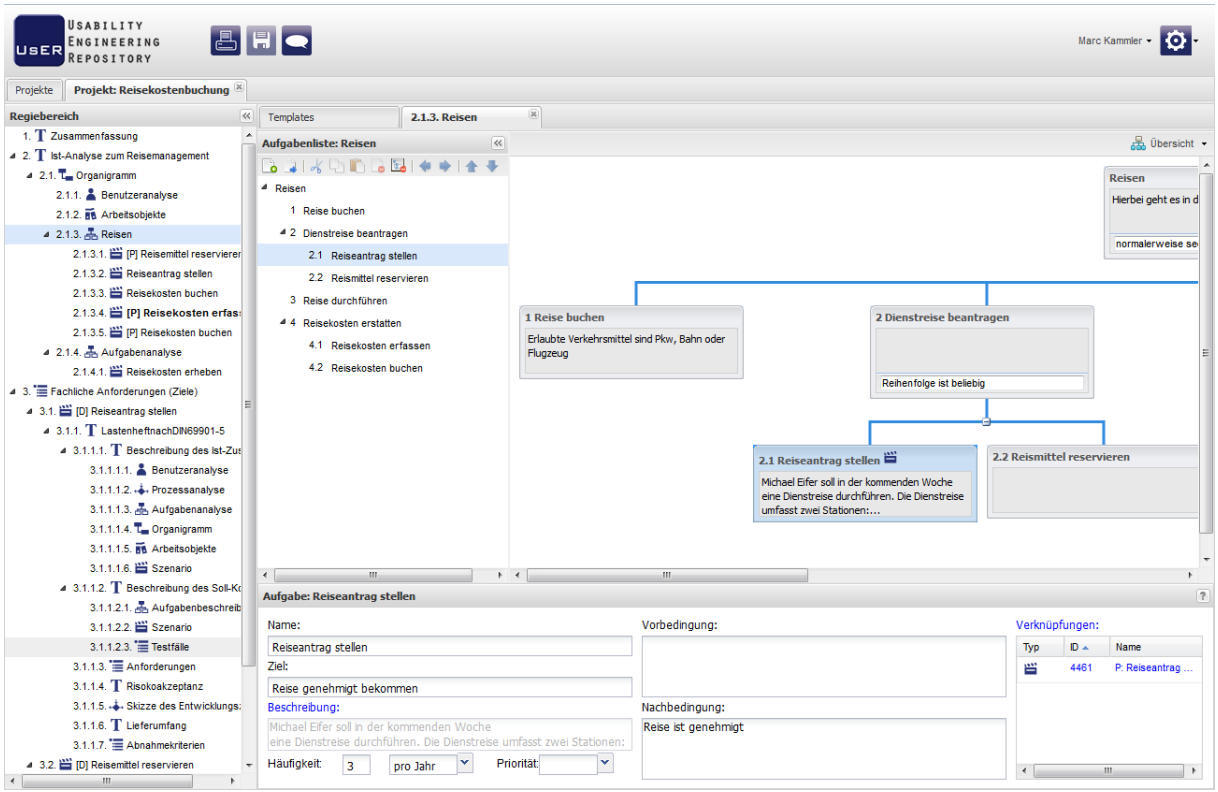


Abbildung 88: Modul für Aufgabenanalysen

Die hierarchische Aufgabenanalyse kann entweder über die Verknüpfungsfunktion des zuvor beschriebenen Benutzeranalysemoduls direkt aufgerufen werden. Alternativ kann jedoch auch die Drag & Drop-Funktion der Projektverwaltung dazu verwendet werden, ein neues Aufgabenanalysekapitel zu erzeugen. Ersteres hätte zur Folge,

dass unter dem entsprechenden Kapitel der Benutzeranalyse in der Kapitelübersicht automatisch ein neues Unterkapitel erzeugt werden würde, welches anschließend geöffnet wird. Das Datenmodell des Moduls orientiert sich an den in Kapitel 2.3.1.2.1 beschriebenen Aufgabenmodellen. So besitzt jede Aufgabe eine eindeutige ID. Diese ist für den Anwender des Moduls zwar nicht wichtig, sorgt jedoch innerhalb des Systems dafür, dass beliebige Entitäten mit einer einzelnen Aufgabe verknüpft werden können. Über den Schriftzug *Verknüpfung* – im Detailbereich unten rechts – kann das Verknüpfungsmenü (siehe Abbildung 89) aufgerufen werden. Diese Funktion ist in einer späten Iteration des UsER-Systems für alle Module aus Konsistenzgründen auf gleiche Weise umgesetzt worden.

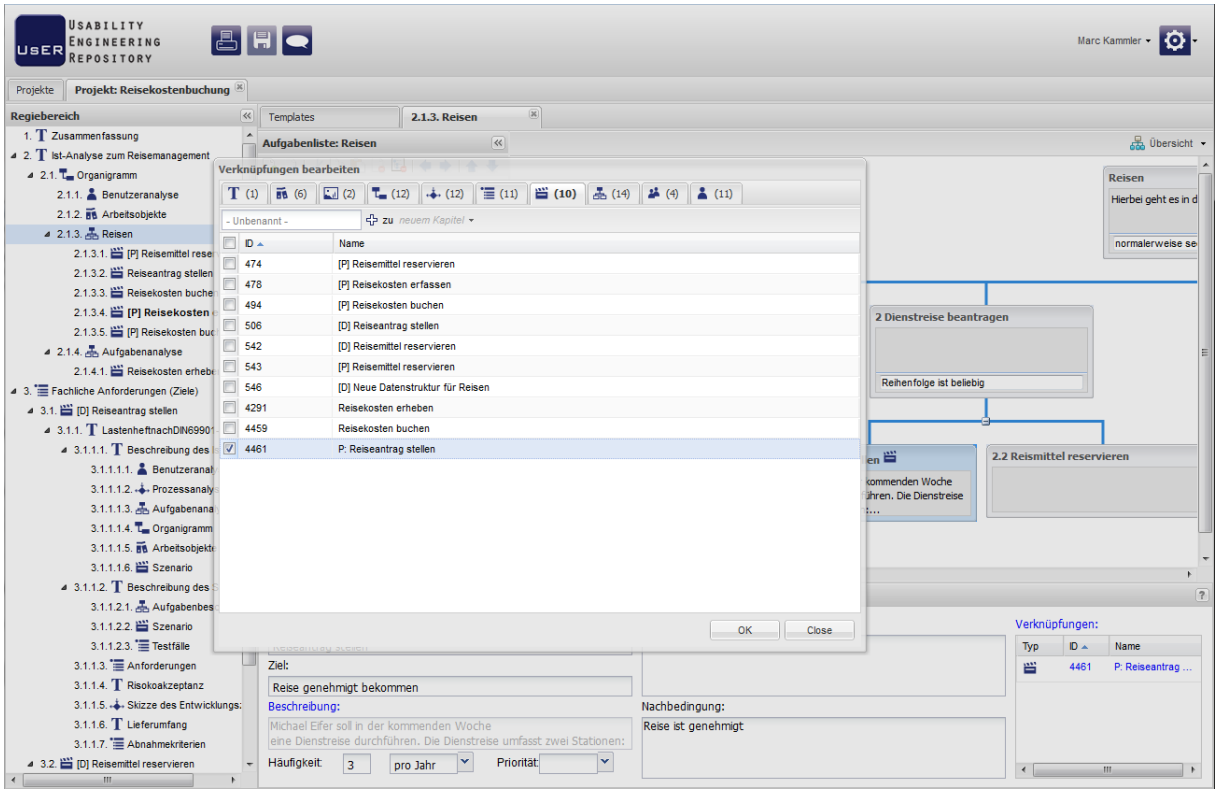


Abbildung 89: Geöffnetes Verknüpfungsmenü im Aufgabenanalyse-Modul

Das sich öffnende Verknüpfungsmenü bewirkt ein Ausgrauen der gesamten Anwendung, und es öffnet sich das Popupmenü *Verknüpfungen bearbeiten*. In diesem sind alle Informationsentitäten eines Projektes in verschiedenen Karteireitern aufgelistet. Hier kann über eine „Choice Box“ eine Entität wie beispielsweise ein Szenario, ein Arbeitsobjekt oder ein Benutzer der jeweiligen Aufgabe hinzugefügt oder entfernt werden. Durch diese Funktion wird nicht nur das Prinzip der HTA unterstützt, sondern auch die von MAD*, GTA und CTT (siehe Kapitel 2.3.1.2).

Um die Eingabe der einzelnen Aufgaben und ihrer Teilaufgaben zu vereinfachen, wurde eine *Aufgabenliste* entwickelt (siehe Abbildung 88 links neben der Baumansicht), der das Editieren der Aufgabendekomposition nur über die Tastatur ermöglicht. In diesem wird der Aufgabenbaum in Form einer Liste dargestellt. Hier können Aufgaben über Tastenkombinationen hinzugefügt und in ihrer Hierarchie verschoben werden. So führt beispielsweise die Return-Taste zur Erzeugung einer neuen Teilaufgabe. Durch die Tastenkombination „Strg + <Pfeiltaste>“ können Aufgaben in dem Baum verschoben werden. Für Gelegenheitsbenutzer können die Funktionen auch über ein entsprechendes Menü oberhalb der Aufgabenliste ausgeführt werden (siehe Abbildung 90).

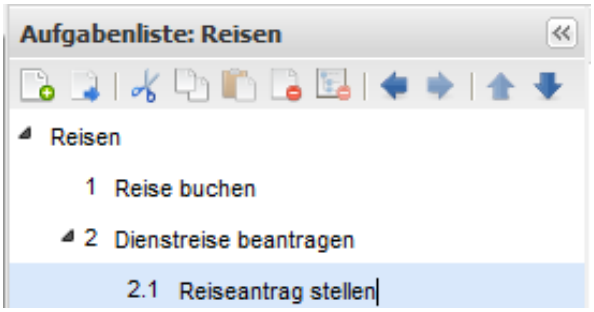


Abbildung 90: Menü der Aufgabenliste

Eine weitere Besonderheit zu den oben diskutierten Aufgabenmodellen wurde ebenfalls erst in einer späteren Iteration des UsER-Systems in das Modul integriert. Das Eingabefeld „Beschreibung“ kann dazu genutzt werden, die bisherige Beschreibung direkt zu einem ausführlichen Szenario auszuformulieren. Dies ermöglicht einen fließenden Übergang von einer abstrakt systematischen Aufgabenbeschreibung hin zu einer narrativen Beschreibungsform. Ähnlich wie bei der Verknüpfungsfunktion kann durch einen einfachen Klick auf das blau gekennzeichnete Label *Beschreibung* direkt ein neues Szenario erstellt werden. Der bisher geschriebene Text wird dabei in dem sich öffnenden Szenario-Unterkapitel übernommen. Die so erweiterte Teilaufgabe wird dadurch noch mit einem Szenario-Symbol markiert (siehe Abbildung 88 neben dem Aufgabennamen „Reiseantrag stellen“). Diese Funktion ermöglicht eine zusätzliche Form Szenarien etwas übersichtlicher zu strukturieren.

Nach Fertigstellung der vierten Konzeptiteration bot sich im Rahmen einer Mitarbeiterschulungswoche des Kooperationspartners die Möglichkeit einer größeren Zahl an Kundenberatern bzw. Anforderungsanalysten die Methode der HTA vorzustellen und auszuprobieren. Zudem konnte im Rahmen dieser Veranstaltung das aktuelle Konzept evaluiert werden (siehe Abbildung 91), welches diese Methode zukünftig unterstützen könnte. Insgesamt nahmen 14 Probanden an der Evaluation teil. Wie auch bei den vorhergehenden Evaluationen wurden die verschiedenen Konzepte in Form von bebilderten Szenarien mit verschiedenen Anwendungsfällen veranschaulicht (Dierck, 2012).

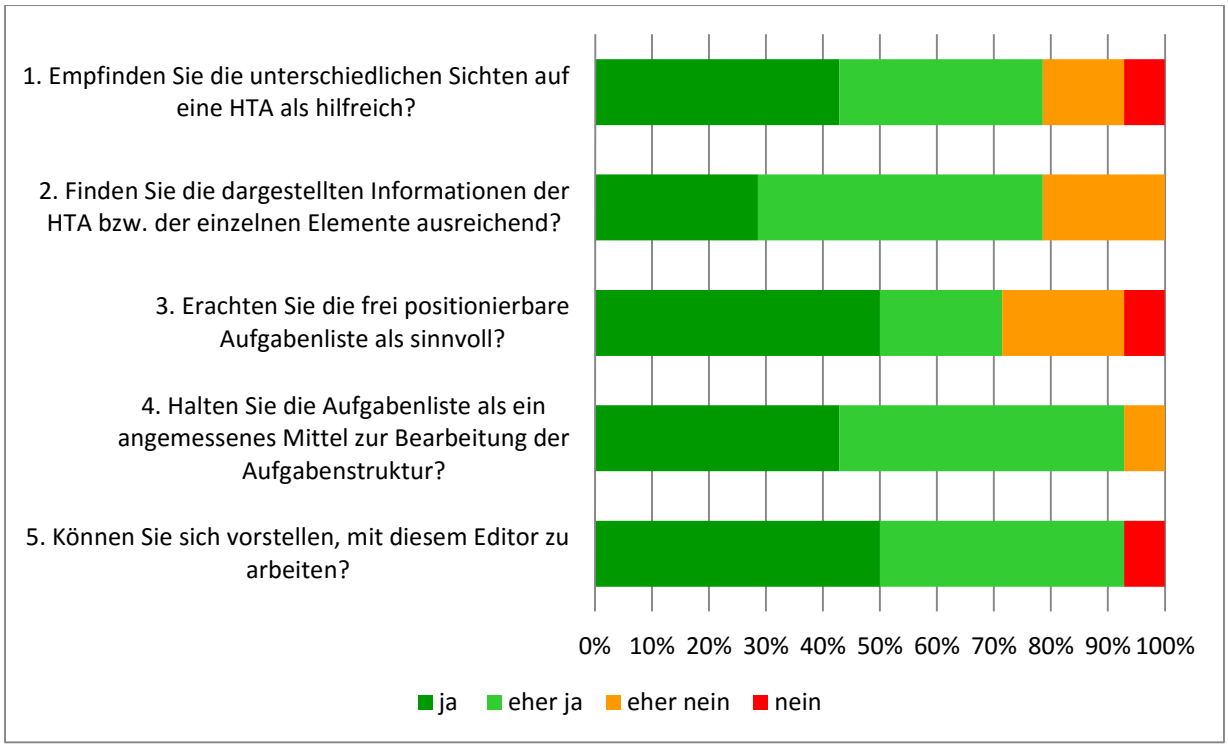


Abbildung 91: Evaluationsergebnisse Aufgabenanalysemodul

Das Feedback der Probanden war überwiegend positiv. Das größte Verbesserungspotential wurde durch Frage 3 in einer angedachten frei positionierbaren Aufgabenliste gesehen. Hier wünschten sich die meisten Probanden

eine feste Position dieser Liste. Dies führte zu der Entscheidung, die Aufgabenliste links neben der Baumansicht fest zu positionieren. Den meisten Probanden reichten die angezeigten Informationen der HTA-Elemente. Zu Testzwecken wurde eine weitere Funktionalität implementiert, die es erlaubte, Rollen und Benutzerinformationen aus dem Benutzeranalysemodul in einem HTA-Element direkt anzuzeigen. Durch eine Hover-Funktion konnten Informationen des jeweiligen Elementes dann in Form eines Tooltips angesehen werden, oder durch Anklicken konnte das entsprechende Modul geöffnet werden. In der aktuellen Version des Moduls wurde diese Funktion aus Gründen der Konsistenz für alle Module in der sogenannten *Verknüpfungskomponente* vereinheitlicht. Mittels dieses Lösungsansatzes können alle Informationselemente eines Projektes mit einer Aufgabe bzw. ihren Teilaufgaben verknüpft werden, wodurch eine Vielzahl von hierarchischen Aufgabemodellen, die in Kapitel 2.3.1.2.1 beschrieben wurden, über das Aufgabenanalysemodul unterstützt werden.

Das Prozess-Modul

Für die detaillierte Darstellung von Arbeitsabläufen kann in UsER das Prozessmodul verwendet werden. Dieses verwendet als Datenmodell eine Teilmenge des BPMN 2.0 Standards. Wie bereits erwähnt, sollten detaillierte Prozessdarstellungen erst im fortgeschrittenen Verlauf der Systementwicklung durchgeführt werden. Um jedoch komplexe Arbeitsabläufe umfassend zu verstehen, ist es oft unvermeidbar, bereits in frühen Phasen auf diese Diagrammdarstellung des Flussmodells zurückzugreifen. So könnte mit Hilfe dieses Moduls beispielsweise eine komplexe Teilaufgabe aus der hierarchischen Baumdarstellung des zuvor beschriebenen Moduls verfeinert werden. Auch komplexe Szenarien können durch die BPMN-Darstellung in strukturierter Weise formuliert werden. In der oberen Menüleiste des Prozessanalysemoduls (siehe Abbildung 92) können die zur Verfügung stehenden BPMN-Elemente ausgewählt werden. Nachdem eines der Elemente ausgewählt wurde, kann es anschließend an beliebiger Stelle auf der Zeichenfläche platziert werden. Aktivitäten lassen sich entweder als Sequenz oder Nachricht miteinander verbinden. Eine Sequenz beschreibt in der Regel eine einfache Abfolge eines Benutzers. Nachrichten treten auf, wenn eine Aktivität von einer zweiten Person übernommen wird. Aufgaben, die von mehr als einem Benutzer bearbeitet werden, können im BPMN-Standard als *Lanes* dargestellt. In dem Beispiel aus Abbildung 92 sind beispielsweise die beiden Rollen des „Reisenden“ und die des „Verwalters“ dargestellt. Über die Funktion *Verknüpfung* von UsER könnte hier jetzt auch noch die Persona-Beschreibung von Michael Eifer der Lane zugeordnet werden. Jedes Element innerhalb des Diagrammes kann über den Detailbereich editiert werden. Neben einem beschreibenden Namen, sieht der BPMN-Standard noch die Möglichkeit vor, Elemente zu kommentieren. Zu diesem Zweck kann im Detailbereich das Textfeld *Annotation* verwendet werden. Über Anklicken der Choice-Box *anzeigen*, kann der Kommentar in dem Diagramm angezeigt oder versteckt werden.

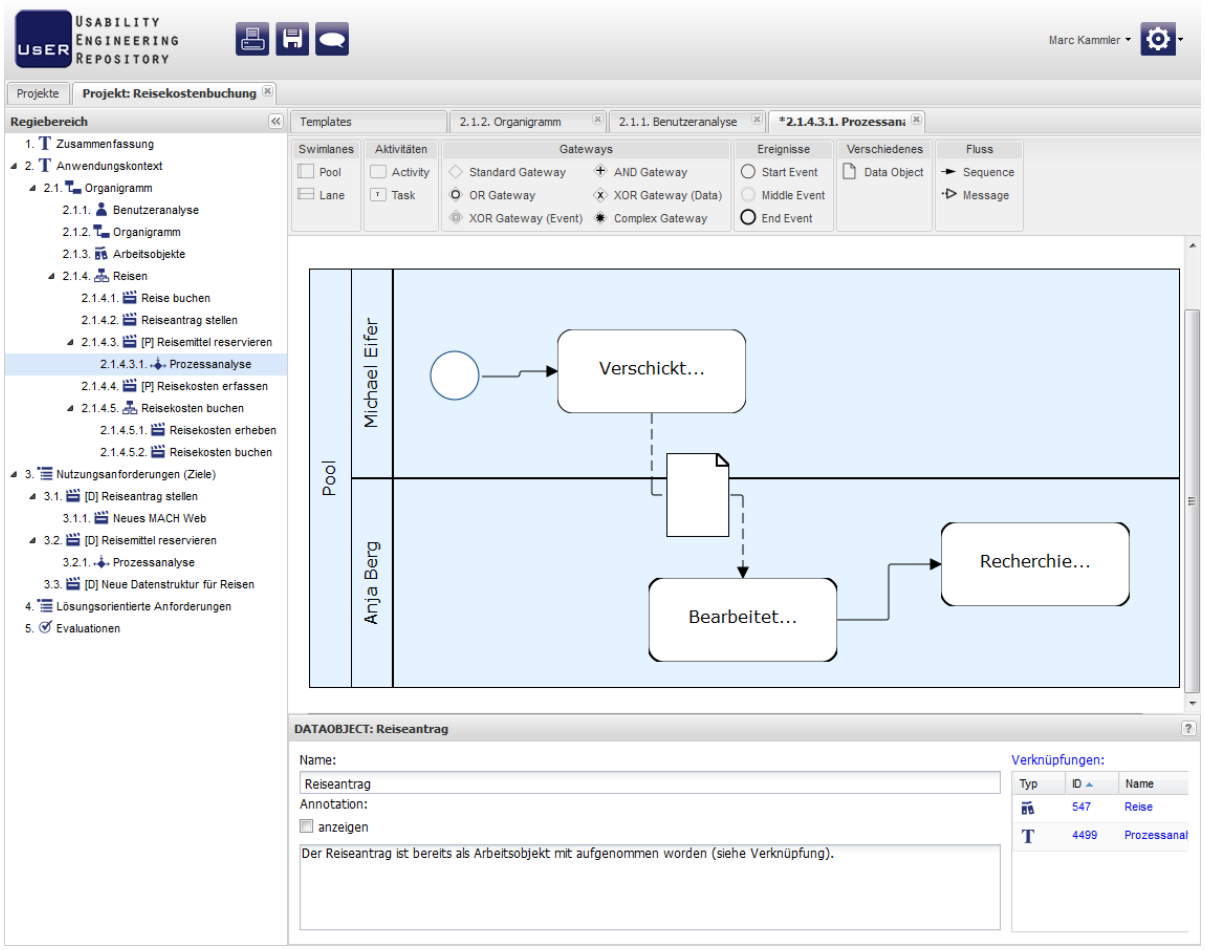


Abbildung 92: Prozessanalysemodul

Über die Funktion *Verknüpfung* können alle Elemente des Diagramms mit beliebigen Entitäten aus dem Projekt verknüpft werden. So kann beispielsweise das in Abbildung 92 markierte Arbeitsobjekt „Reiseantrag“ mit einem Element aus dem im Folgenden beschriebenen Artefakt-Modul ergänzt werden. Auch die Schachtelung in Form des 6-Ebenen Modells (siehe Kapitel 4.4.5), lässt sich über die Verknüpfungsfunktion abbilden.

5.2.3 Artefakt-Analyse

„Aus Sicht des Benutzers liegt das Hauptinteresse zunächst nicht am Werkzeug als solchem, sondern an den zu manipulierenden Objekten des Gegenstandsbereiches, […].“ (Specker, 1998). Arbeitsobjekte speziell für die Entwicklung betrieblicher Systeme sind häufig der eigentliche Gegenstand der Arbeit und dürfen bei einer umfassenden Analyse einer Ablauforganisation nicht fehlen. Zu diesem Zweck ermöglicht das folgende Modul die Erfassung von Arbeitsobjekten. Neben der Bezeichnung und einer optionalen Beschreibung des jeweiligen Arbeitsobjektes ermöglicht es auch die Dokumentation wichtiger Attribute des jeweiligen Arbeitsobjektes (Abbildung 93).

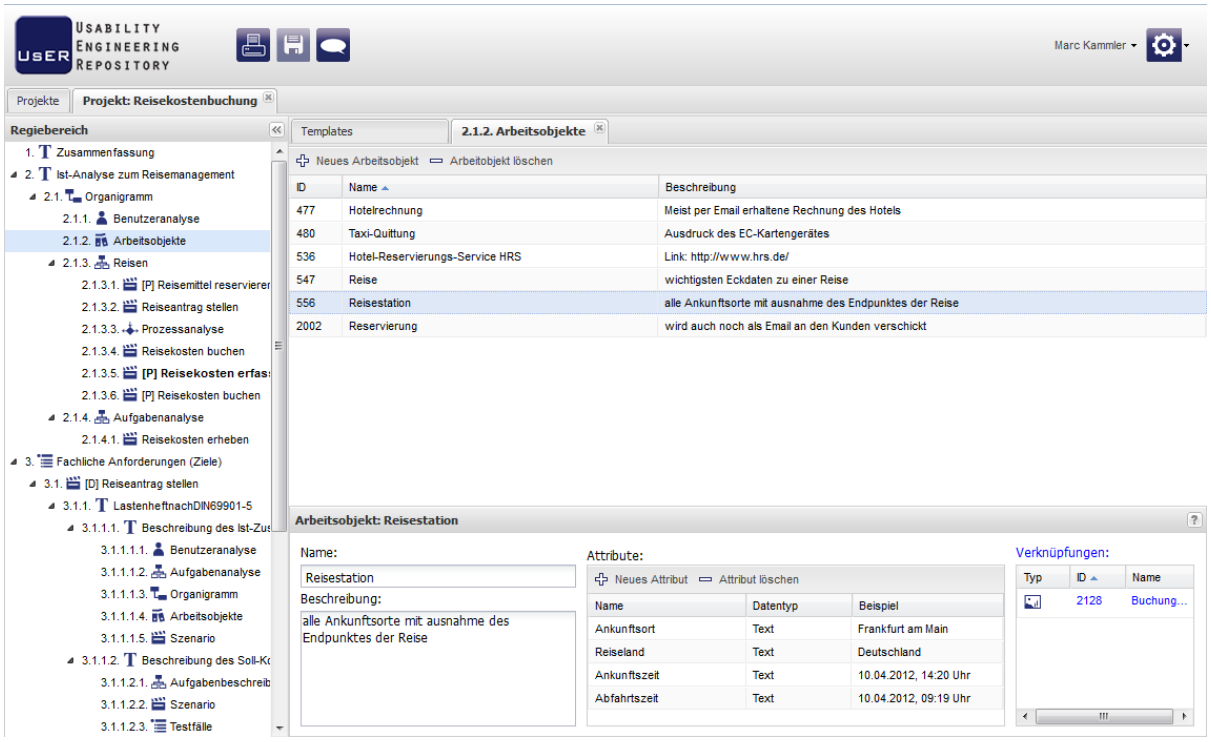


Abbildung 93: Modul für die Artefakt-Analyse

Diese können als Ausgangslage eines Datenmodells fungieren oder in den folgenden Modulen der Beschreibung der Aufbauorganisation dienen.

5.2.4 Organisations-Analyse

Informationen zu betrieblichen Aufbaustrukturen können in UsER mit Hilfe des Organisationsanalyse-Moduls erstellt werden. Wie in Kapitel 2.3.1.3 bereits diskutiert, sollten in einem Prozess zur Gestaltung gebrauchstauglicher interaktiver Systeme auch Informationen zu betrieblichen Aufbaustrukturen einfließen. Diese werden häufig in Form von *Organigrammen* abgebildet. Da hierfür jedoch keine festen Standards existieren, mussten bei der Entwicklung dieses Moduls auf Best Practices aus der Betriebswirtschafts- und Organisationslehre zurückgegriffen werden. Zusätzlich wurden existierende Systemlösungen wie beispielsweise Microsoft Visio dazu verwendet, um eine möglichst erwartungskonforme Lösung zu finden. Wir entschieden uns ebenfalls für eine hierarchische Struktur wobei *Organisationseinheiten*, *Stellen* und *Stabstellen* unterschieden werden. Alle bisher vorgestellten Informationseinheiten wie *Artefakte* und *Benutzer* lassen sich durch dieses Modul weiter verfeinern, wodurch das eigentliche Ziel des ersten Prozessschrittes aus der ISO 9241:210 unterstützt werden soll, den Nutzungskontext zu verstehen und zu beschreiben. Auf diese Weise kann auch die in Kapitel 2.3.1.3 diskutierte *Aufgabensynthese* durchgeführt werden. Diese sieht für die Beschreibung der Aufbauorganisation die Zuordnung von Arbeitsaufgaben zu einer bestimmten Stelle vor. In dem Beispiel aus Abbildung 94 kann so die zuvor mit dem HTA-Modul beschriebene Aufgabe des „Reisens“ der entsprechenden Stelle des „Kundenbetreuers“ zugeordnet werden. Zusätzlich ist in diesem Beispiel auch gleich die Persona von „Michael Eifer“ und die Benutzerklasse der „Reisenden“ der Stelle zugeordnet worden.

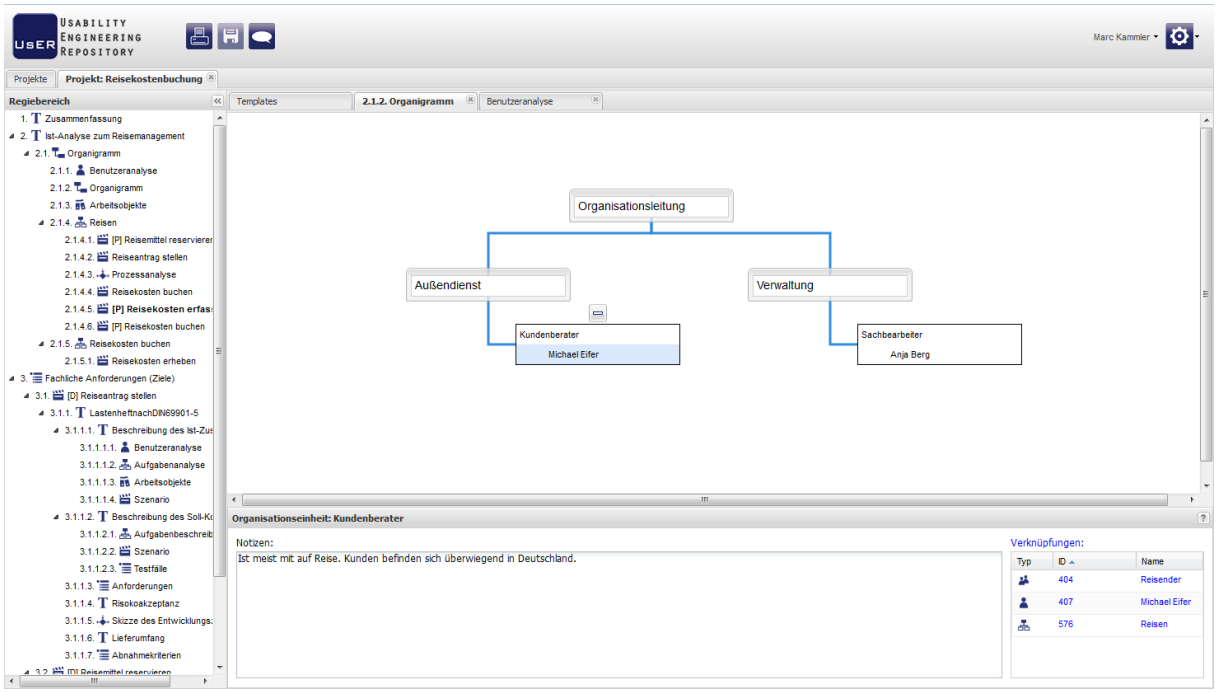


Abbildung 94: Modul für Organisationsanalysen

Um dem Organigramm neue Elemente hinzuzufügen, wurde das Modul mit einem Mouse-Over-Menü ausgestattet (siehe Abbildung 95). D.h. um einem Element einen neuen Kindknoten hinzuzufügen, erscheint ein Menü oberhalb des Elementes, sobald die Maus über den Knoten bewegt wird. In diesem Menü kann dann ausgewählt werden, welchen Typus das zu erstellende Element darstellen soll. Es kann zwischen einer neuen Stelle, Organisationeinheit oder Stabstelle gewählt werden. Der vierte Menüpunkt wird zum Löschen des ausgewählten Knotens verwendet. Das Löschen eines Vaterknotens bewirkt das Entfernen aller Kindknoten.

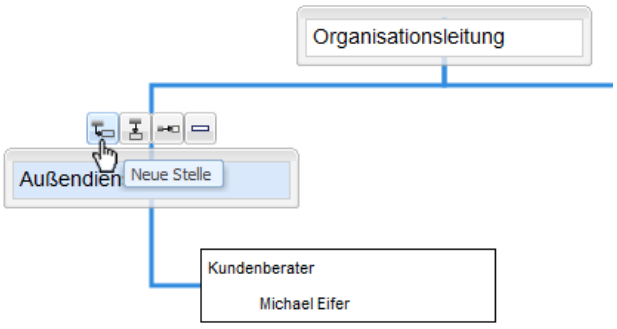


Abbildung 95: Mouse-Over- Menü des Organisationsanalyse-Moduls

Sowohl das Organisations-Analyse-Modul als auch das zuvor beschriebene Artefakt-Analyse-Modul wurden in einer ersten Version im Rahmen einer Diplomarbeit (Haid, 2011) erstellt. Die Ergebnisse der nachfolgenden Evaluation (siehe Abbildung 96) beruhen auf den Antworten von 14 Mitarbeitern des Kooperationspartners, die sich ausgehend von der ersten Frage mit insgesamt 86 % der Antworten für die Verwendung dieser Module ausgesprochen hatten. Weiterhin war zum damaligen Zeitpunkt in diesem Zusammenhang wichtig zu erfahren, ob die Mitarbeiter überhaupt die Möglichkeit besitzen, die Kunden bei der Arbeit beobachten zu können, um anschließend eine solch detaillierte Kontextanalyse durchführen zu können. Was aus der Antwort an dieser Stelle nicht hervorgeht, ist die Tatsache, dass die Mitarbeiter zwar diese Möglichkeit besitzen, sie jedoch aus Zeitgründen nur sehr selten wahrnehmen.

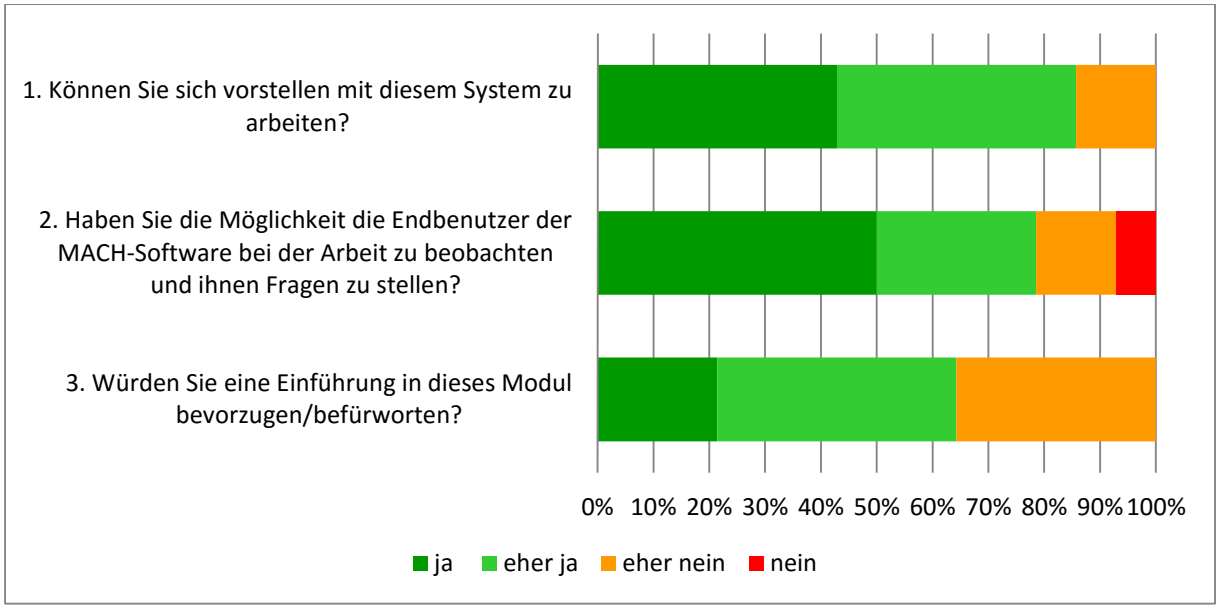


Abbildung 96: Evaluationsergebnisse Organisation-/Artefakt-Modul

5.3 Spezifizieren der Anforderungen

Wie aus dem Praxistest aus Kapitel 4.3 hervor geht, ist die Ermittlung der „richtigen“ Anforderungen eine komplexe Aufgabe. Häufig werden von den Kunden Anforderungen bereits in Form von Lösungsansätzen formuliert. Nicht selten sind diese Personen selbst keine Anwender, die von der Systemänderung oder Systemneuerung betroffen sind. Wie bereits erwähnt, half in den Kundengesprächen der Praxistests häufig die einfache Frage nach dem „Warum?“, um zu dem eigentlichen Problem bzw. zu der „richtigen“ Anforderung zu gelangen. Somit ist es wichtig, an dieser Stelle noch einmal zu erwähnen, dass die Spezifikation der Anforderungen dieses Prozessschrittes häufig erst nach mehreren Iterationen abgeschlossen ist. Zudem sollten Anforderungen dieses Prozessschrittes keine Lösungsansätze beschreiben, sondern die Ziele der im nächsten Prozessschritt zu entwickelnden Gestaltungslösungen. Die Zielformulierungen (Anforderungen) sollten dabei auf den bereits gesammelten Informationen zu den Benutzern, ihren Arbeitsaufgaben und der organisatorischen Strukturen basieren. Neben funktionalen Anforderungen sollten auch organisatorische Änderungen und neue Arbeitsweisen erfasst werden. Mit anderen Worten sollten die zuvor erhobenen Kontextinformationen dazu genutzt werden, bisherige Probleme zu identifizieren und zu benennen. Zu diesem Zweck wurde die bereits erwähnte *Verknüpfungs-Komponente* in allen Modulen von UsER integriert (siehe Abbildung 94 unten rechts). Sie ermöglicht die direkte Erstellung von Anforderungen aus allen Modulen heraus. Die so erstellten Anforderungen können dann im folgenden Modul weiter spezifiziert und bearbeitet werden.

5.3.1 Anforderungsmanagement

Mit Hilfe des Anforderungsmoduls werden jegliche Funktionen unterstützt, die für die Organisation und die Verwaltung von Anforderungen benötigt werden. Dabei werden Informationen zum Autor, dem Erstelldatum oder der Quelle aus der die Anforderung abgeleitet wurde, vom System automatisch erfasst. Innerhalb des Moduls können Anforderungen dann noch weiter kategorisiert (z.B. nach Systemkomponenten), sortiert, priorisiert und gruppiert werden. Die zuvor angesprochene Rückverfolgbarkeit aller für die Entscheidung relevanten Informationen aus Analyse, Design oder Evaluation sind jederzeit über die Verknüpfungskomponente aufrufbar. Auf diese Weise können Anforderungen als Schnittstellenelement zwischen beiden Bereichen des UE und des SE verwendet werden. Der Lebenszyklus einer Anforderung (siehe Kapitel 3.2.3) wird über die Eigenschaft *Status* abgebildet. So kann jede Anforderung individuell dem jeweiligen Reifegrad innerhalb des gesamten Entwicklungsprozesses von der Analyse bis zum Test zugeordnet werden. Auf diese Weise können wie bei jedem der vorgestellten SE-Prozesse Anforderungen dazu verwendet werden, um mit deren Hilfe den Entwicklungsprozess zu koordinieren. Durch die Anpassungen bei der Formulierung von Anforderungen in dem Textfeld *Beschreibung* kann eine Anforderung in Form von User Stories (siehe Kapitel 3.1.5.2) oder Zielen (siehe Kapitel 3.3.1) formuliert werden

und auf diese Weise dem Kunden den abstrakten Mehrwert kommunizieren, der sich nach der Implementierung ergeben sollte. Im weiteren Entwicklungsprozess sollten sie als Ausgangsinformation weiterer Verfeinerungen dienen.

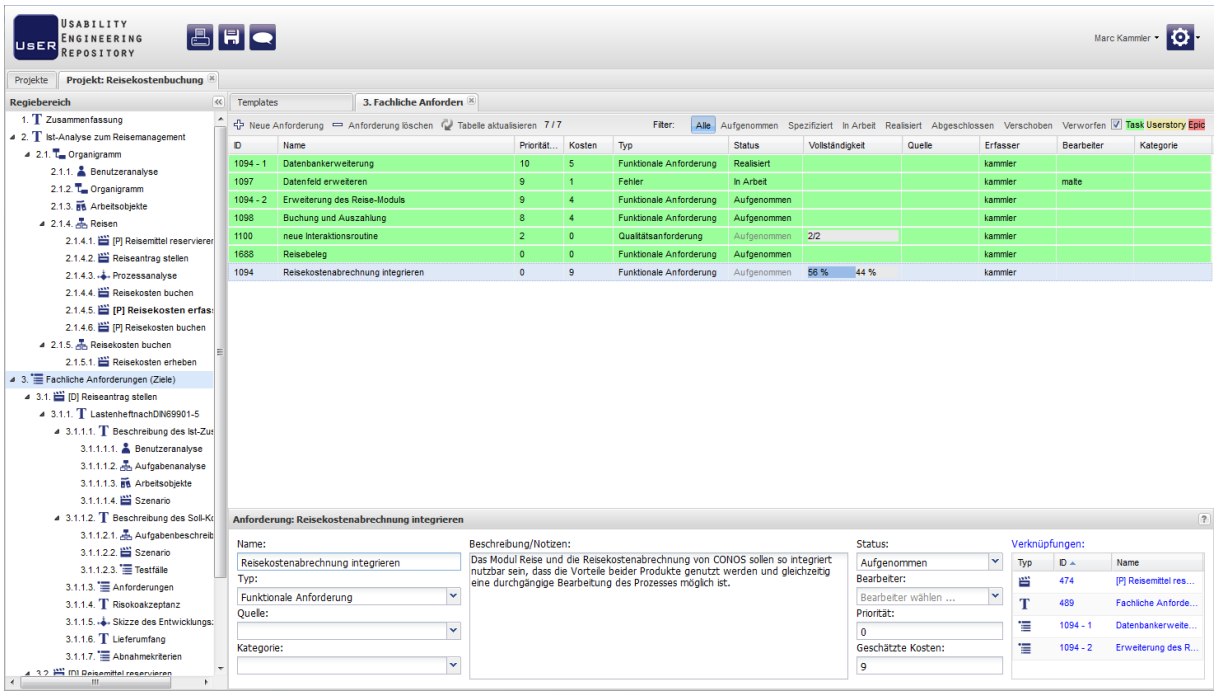


Abbildung 97: Modul zur Anforderungsverwaltung

Darüber hinaus kann jede Anforderung auch in beliebig viele Teilanforderungen verfeinert werden. Für diese Form der Dekomposition wird die entsprechende Anforderung markiert und über das Kontextmenü die Funktion *Verfeinern* (siehe Abbildung 98) aufgerufen. Dadurch wird eine neue Anforderung mit derselben ID und dem Zusatz in Form von „ - <Zahl>“ generiert. Die neu generierte Anforderung ist dadurch unmittelbar mit der Ursprungsanforderung verknüpft (Mertens, 2012).

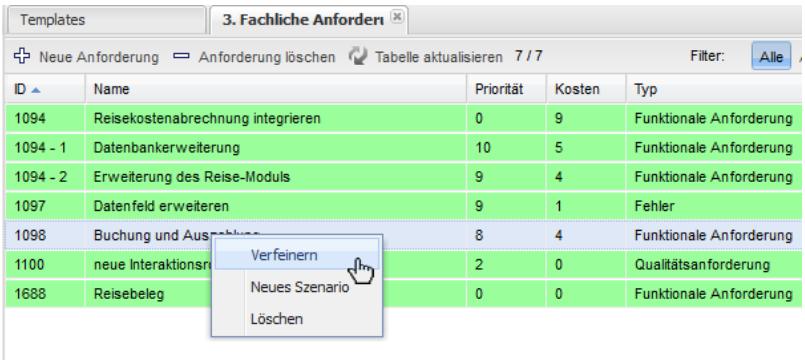


Abbildung 98: Dekomposition von Anforderungen in UsER

5.4 Verfeinern der Anforderungen und Entwurf von Lösungen

Im nächsten Prozessschritt gilt es, Gestaltungslösungen zu entwerfen, welche die zuvor identifizierten Nutzungsanforderungen[17] erfüllen. In UsER können zu diesem Zweck Gestaltungslösungen direkt aus der entsprechenden Anforderung erstellt werden. Über das in Abbildung 98 zu sehende Kontextmenü kann über die Funktion *Neues Szenario* ein neues Szenario-Kapitel erstellt werden. Die Inhalte dieses Kapitels sind auf diese Weise dann direkt

[17] Der Begriff der Nutzungsanforderungen wird in der ISO 9241-210 für Anforderungen verwendet, die sich klar von klassischen Anforderungen aus dem SE unterscheiden.

mit der entsprechenden Anforderung verknüpft. Gleichzeitig öffnet sich das nachfolgend beschriebene Szenario-Modul, mit dem dann eine mögliche Gestaltungslösung erstellt werden kann.

5.4.1 Szenario-Modul

Mit Hilfe des Szenario-Moduls kann dann die entsprechende Gestaltungslösung erstellt werden. Methodisch sollte bei der Erstellung der Gestaltungslösung durch Szenarien darauf geachtet werden, dass die Beschreibungen unterschiedliche Abstraktionsgrade durchlaufen. Rosson und Carroll (2002) schlagen hierzu die schrittweise Transformation von Szenarien in Form von Aktivitäts- über Informations- hin zu Interaktionsszenarien vor (siehe Kapitel 2.2.4). Im ersten Schritt werden nur die zukünftig durchzuführenden Aktivitäten der Benutzer beschrieben. Erst wenn diese geklärt sind, werden die dazu notwendigen Informationen ergänzt. Sind diese in weiteren Iterationen geklärt, wird die Interaktion mit den vom System anzuzeigenden Informationen ergänzt. Erst in der letzten Iteration wird die Interaktion mit dem System beschrieben.

Der Grundaufbau des in UsER integrierten Szenarien-Moduls orientiert sich mit seinem unteren Detailbereich, sowie dem Hauptbereich an der zugrunde liegenden UsER-Architektur (Beholz, 2012). Der sich im Hauptbereich befindliche Texteditor ist mit einer Werkzeugleiste ausgestattet. Diese ermöglicht bekannte Textfunktionen wie Schriftgröße, Schriftstil, Fett, Kursiv und Farbänderungen. Zudem sind Absatzfunktionen wie Listen, Aufzählungen und Einzüge möglich. Für die Integration von Bildern und Grafiken wurden verschiedene Möglichkeiten implementiert. Dem Benutzer ist es freigestellt, ob er externe Grafiken über ein integriertes Medienarchiv in den Editor integriert oder diese direkt von seinem Desktop an die entsprechende Textstelle zieht. Grafiken, die durch die zweite Möglichkeit integriert wurden, werden beim Speichern des Szenarios automatisch in dem Medienarchiv abgelegt. Zudem wurde eine Möglichkeit geschaffen, zusätzliche Kontextinformationen, wie beispielsweise Benutzer, Rollen oder Artefakte (siehe Kapitel 2.3.3.1) mit dem Text zu verbinden.

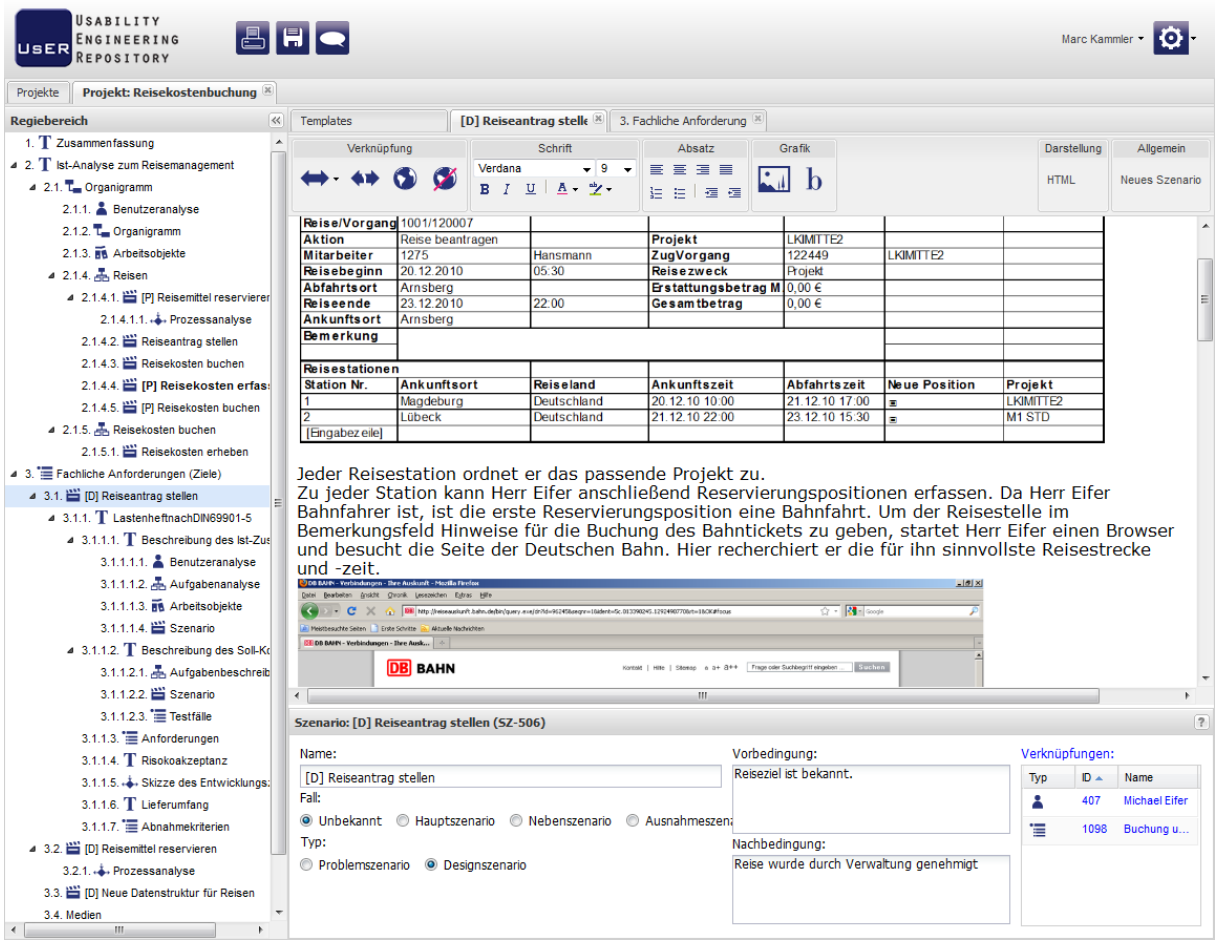


Abbildung 99: Beschreibung einer Gestaltungslösung mit Szenarien-Modul

Dazu muss eine Textstelle markiert werden und anschließend über das Verknüpfungsmenü, wie in Abbildung 100 zu sehen, ein entsprechendes Element gewählt werden. Zudem kann über die Menüfunktion *neue Entität erzeugen* auch ein leeres Element erstellt werden, welches dann zu einem späteren Zeitpunkt in dem entsprechenden Modul detailliert werden kann.

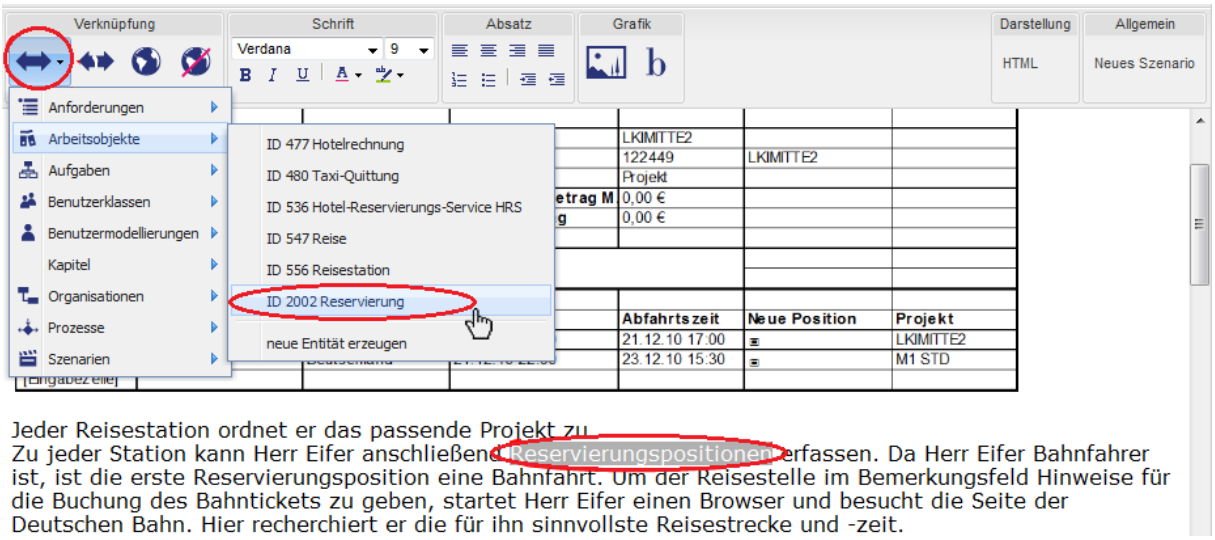


Abbildung 100: Verknüpfung von Kontextinformationen mit Texten

Für die Erstellung ganzer Anwendungsfälle nach Pohl (2008) (siehe auch Kapitel 2.3.3.1) kann über die Funktion *Neues Szenario* (siehe Abbildung 101 rechts) ein neues Szenarien-Unterkapitel erstellt werden. In diesem können dann weitere Alternativ- oder Ausnahmefälle beschrieben werden, die direkt mit dem Hauptszenario verknüpft sind.

Abbildung 101: Aufruf des integrierten Mockup-Editors

Auf diese Weise lassen sich beliebig detaillierte Aktivitätsszenarien beschreiben. Um diese anschließend auf einfache Weise zu Informationsszenarien auszubauen, wurde in das Szenarien-Modul der nachfolgend beschriebene Mockup-Editor integriert.

5.4.2 Mockup-Editor

Um mit dem Szenarien-Modul auch die in Kapitel 2.3.3.1 besprochenen Storyboards erstellen zu können, wurde in UsER ein externer Mockup-Editor der Firma Balsamiq integriert. Über die in Abbildung 101 markierte Schaltfläche des Szenarien-Moduls öffnet sich der Mockup-Editor (siehe Abbildung 102).

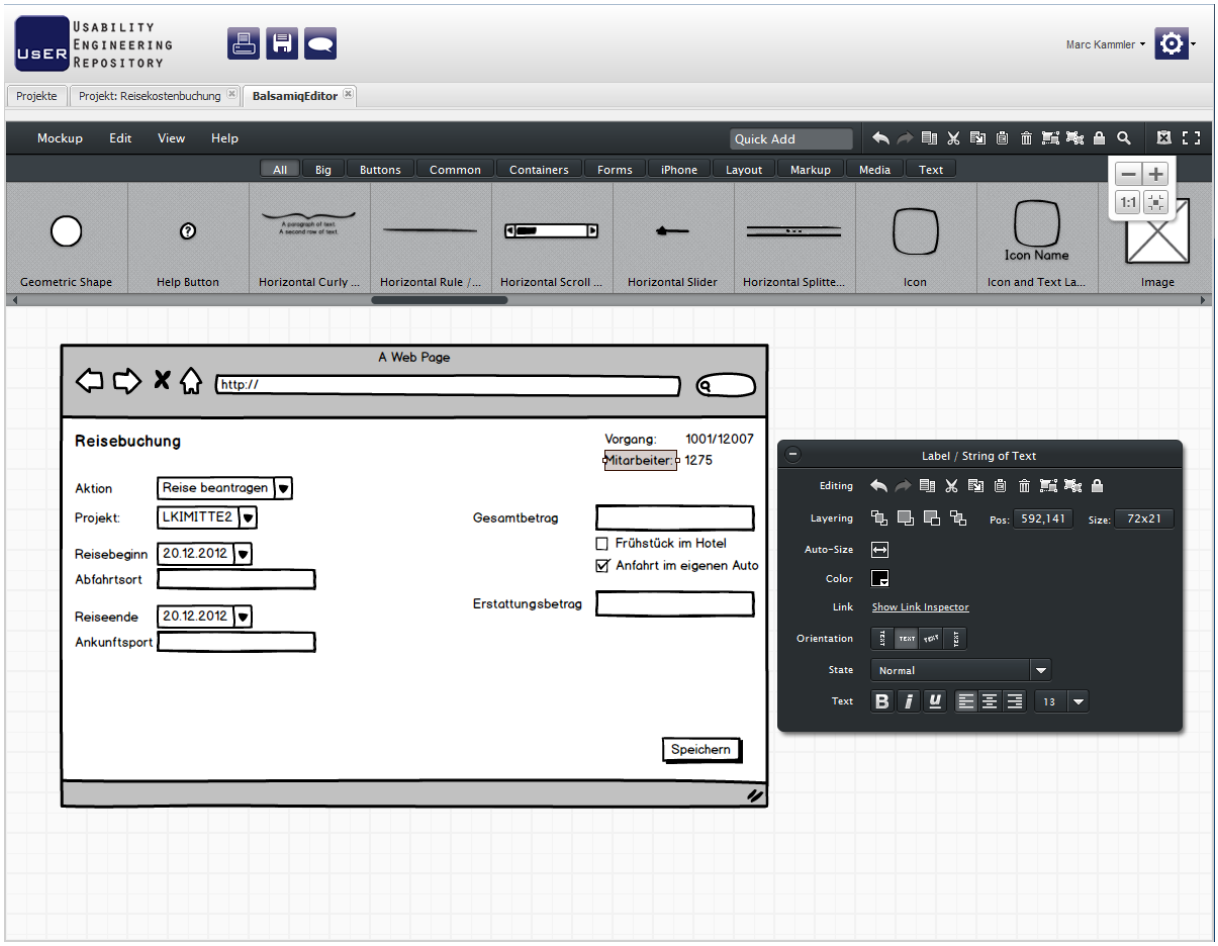


Abbildung 102: Aufruf von Balsamiq in UsER

Dieser ermöglicht eine sehr effiziente Erstellung von Mockups, die eine Gestaltungslösung verdeutlicht. Die mit dem Mockup-Editor erstellten Grafiken lassen sich anschließend direkt in das jeweilige Szenario überführen und dadurch ein Aktivitätsszenario in ein Informationsszenario verwandeln (Dubbels, 2011). Auf diese Weise können aussagekräftige Storyboards erstellt werden, die auch von Kunden verstanden werden können.

5.4.3 Verfeinern der Anforderungen

Bei der Verfeinerung der Anforderungen wurde in UsER eine Lösung gesucht, die sich an den Ideen und Modellen von CREWS (siehe Kapitel 2.4.1) und der Ziel-Szenario-Kopplung von Pohl (siehe Kapitel 3.3.1) orientiert. Die Anforderungsanalyse sollte ausgehend von einem Ziel (abstrakte Anforderung) mittels eines Szenarios konkretisiert werden. In dem CREWS-Projekt wurde die Kombination der beiden Entitäten als *Chunk* bezeichnet. Diese Kombination aus abstrakter Anforderung und konkretisierendem Szenario lassen sich in UsER problemlos miteinander verknüpfen (siehe Abbildung 103).

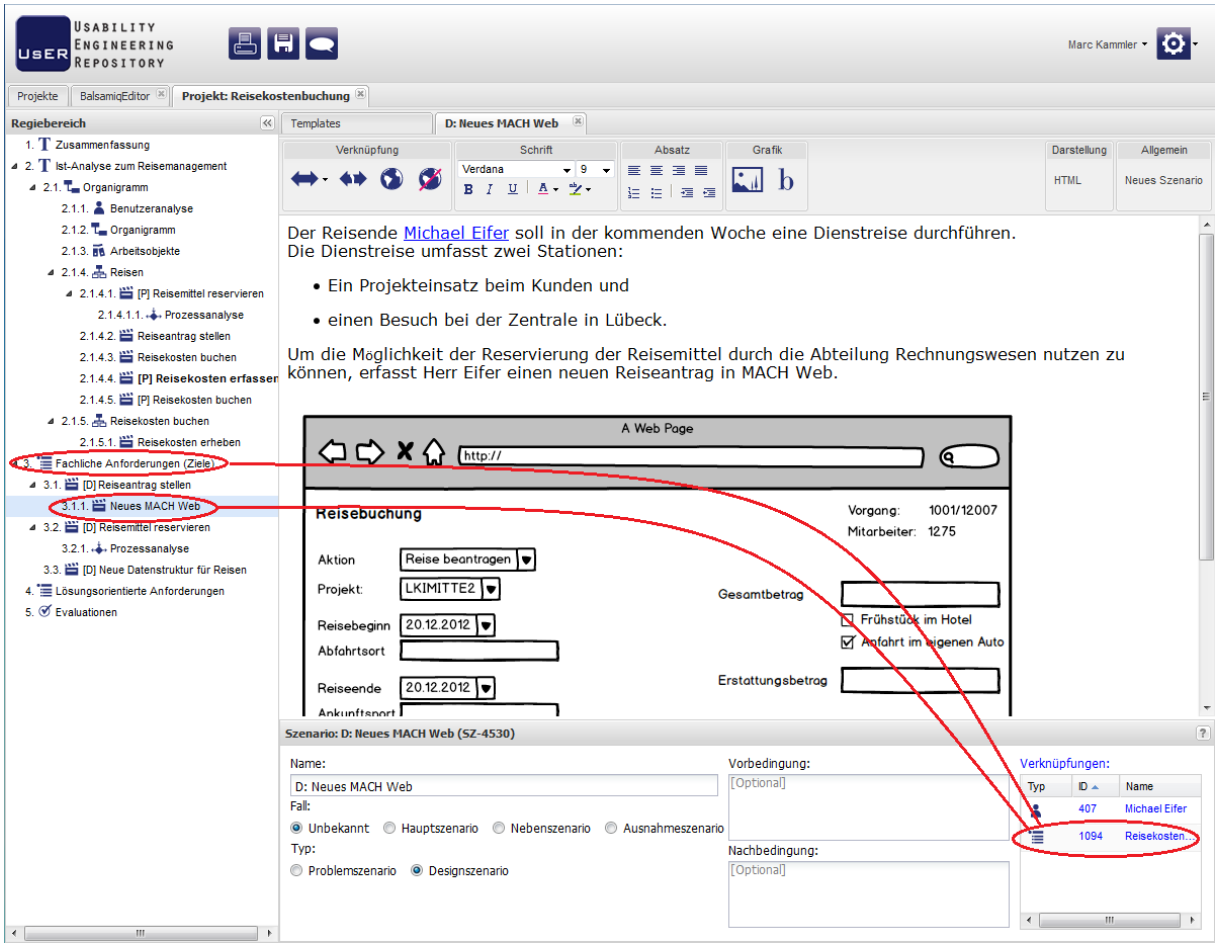


Abbildung 103: Verknüpfung eines Szenarios mit einer Nutzungsanforderung

Durch die Formulierung einer Gestaltungslösung in Form eines Szenarios können zum einen weitere Anforderungen des Kunden identifiziert werden, die bisher noch nicht berücksichtigt wurden. Zum anderen kann die Gestaltungslösung jedoch auch dazu verwendet werden, Anforderungen zu formulieren, die anschließend der Entwicklung übergeben werden können. Klaus Pohl bezeichnet den zweiten Typ von Anforderung als „Lösungsorientierte Anforderung“ (siehe Kapitel 3.3.1). Dieses Vorgehen wird ebenfalls von UsER unterstützt. Anforderungen können in beliebig vielen Kapiteln strukturiert werden. Dazu wurde die bereits bekannte *Verknüpfungskomponente* aus dem Detailbereich um eine Funktionalität erweitert. Für das Beispiel aus der oben angeführten Grafik kann in dem geöffneten Verknüpfungsfenster (siehe Abbildung 104) das Kapitel mit dem Namen „Lösungsorientierte Anforderung“ ausgewählt werden. Die Benennung der Kapitel kann abhängig von dem Vokabular des jeweiligen Entwicklungsprozesses gewählt werden. So könnte beispielsweise auch die Benennung in Lasten- und Pflichtenanforderungen, wie beim V-Modell XT propagiert, verwendet werden. Über das Textfeld links dieses Dropdownmenüs können anschließend die neu identifizierten Anforderungen in das entsprechende Kapitel eingefügt werden.

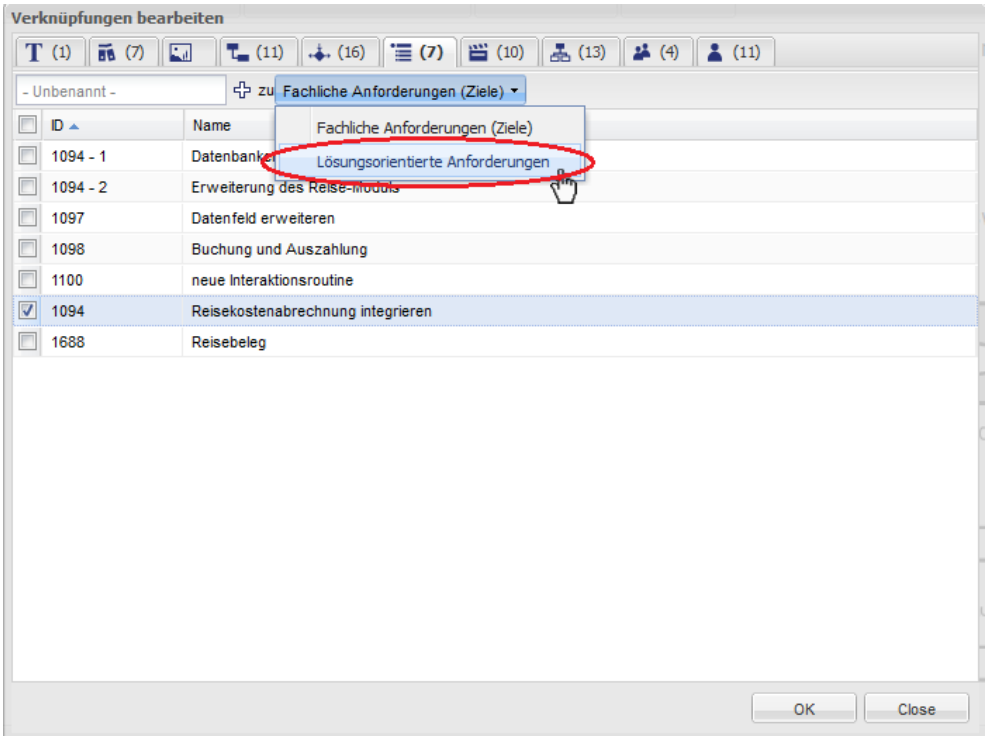


Abbildung 104: Verknüpfungen in beliebigen Kapiteln strukturieren

Auf diese Weise kann jede zu implementierende Anforderung (bspw. Pflichtanforderung oder lösungsorientierte Anforderung) ihrer Nutzungsanforderung (bspw. Ziele oder Lastenanforderungen) zugeordnet werden. Zudem kann ein Entwickler die zu einer Anforderung gehörige Gestaltungslösung in Form eines Szenarios oder einem Storyboard einsehen.

In einem dokumentierten Praxistest mit drei Probanden, wurde der Einsatz des Szenarien-Moduls getestet (Beholz, 2012). Die sich bereits im Vorfeld abzeichnende positive Einstellung bezüglich des Einsatzes von Szenarien in der Softwareentwicklung wurde auch hier bestätigt (siehe Abbildung 105). So waren alle Probanden einstimmig der Meinung, dass Szenarien ein angemessenes und hilfreiches Mittel zur Spezifikation und Identifikation von Anforderungen seien. Die Fragen eins und drei zeigten, dass mit der Umsetzung des Szenarien-Moduls bereits die Methodik der Szenarien Erstellung unterstützt werden kann. Es wurde jedoch auch deutlich, dass einige der unterstützten Methoden wie beispielsweise die Unterteilung in Problem- und Design-Szenarien nach Rosson und Carroll (2002) auch das Verständnis des entsprechenden Vorgehens verlangt. Probleme mit der Performance - siehe Frage 5 - wurden nach der Evaluation des Moduls nachträglich verbessert.

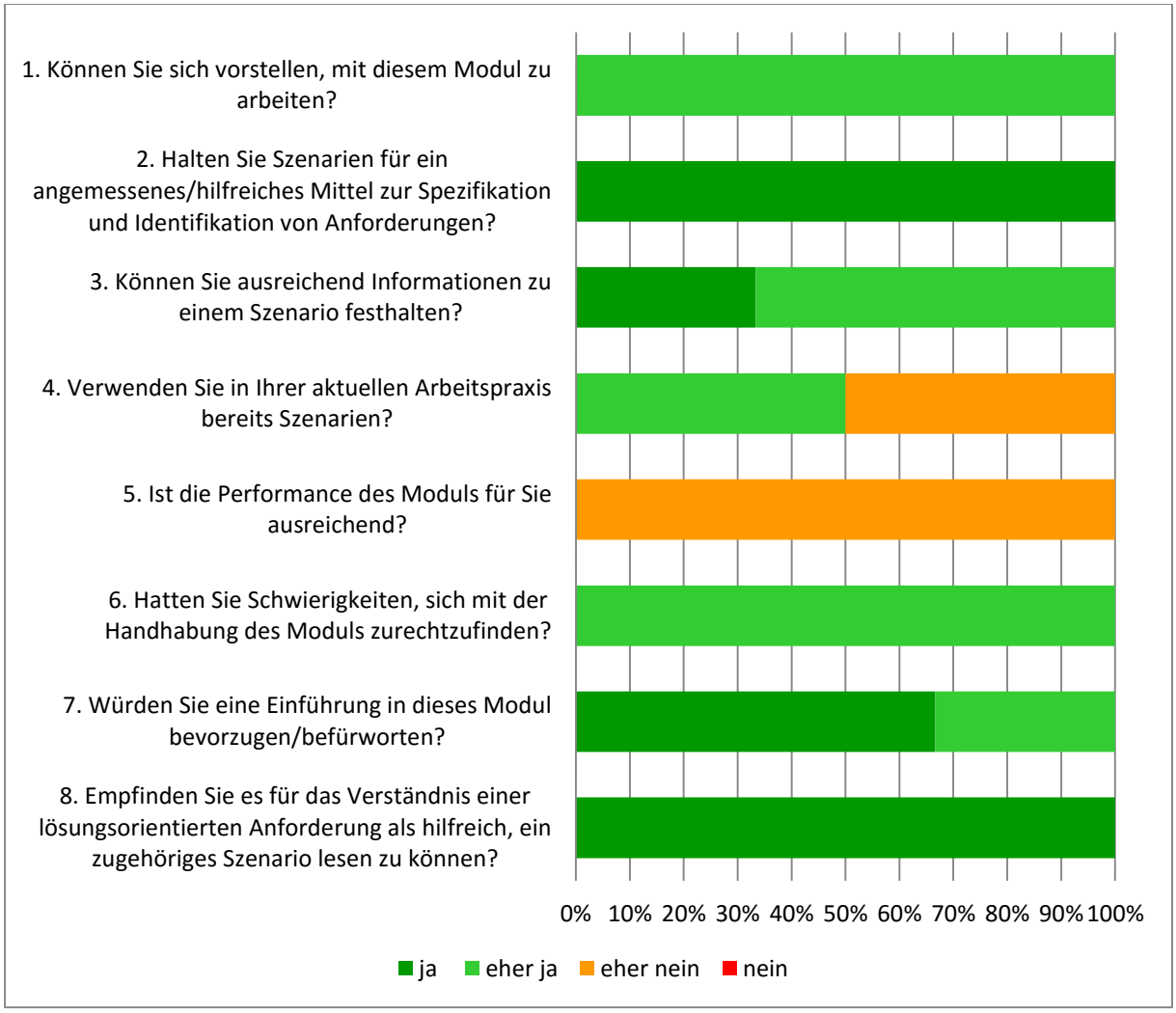


Abbildung 105: Evaluationsergebnisse Szenarien-Modul

5.5 Evaluieren und Bewerten von Lösungen

In der nächsten Phase des Entwicklungsprozesses müssen die entstandenen Gestaltungslösungen in Bezug auf die formulierten Anforderungen evaluiert und bewertet werden. Hierzu werden in der ISO 9241-210 zwei verschiedene Verfahren vorgeschlagen. Zum einen durch die *inspektionsbasierten Evaluationen* durch Fachleute aus dem Bereich der Gebrauchstauglichkeit. Bei dieser formativen Form der Evaluation versetzen sich die Fachleute in die Lage der späteren Benutzer und überprüfen die entsprechenden Gestaltungslösungen hinsichtlich geltender Richtlinien und Normen. Zum anderen die Bewertung durch den Benutzer. Beide Verfahren werden durch UsER unterstützt.

5.5.1 Kommentare und Diskussion

Über die bereits angesprochene Kommentarfunktion kann jedes Element eines Projektes annotiert und hinterfragt werden. Über eine Schaltfläche im Hauptmenü (siehe Abbildung 106) kann der *Kommentarbereich* geöffnet werden. In diesem können beliebig viele Empfänger aus einem Dropdown-Menü ausgewählt werden, an den die Nachricht geschickt werden soll. Autor und Zeitpunkt der Erstellung werden automatisch mitgespeichert. Dies ermöglicht die Ansicht chronologischer Diskussionsverläufe, wodurch Entscheidungsprozesse nachvollzogen werden können. Durch das Selektieren einer beliebigen Nachricht in der Liste des Kommentarbereiches ändert sich die Beschriftung des Buttons unter dem Kommentartextfeld von *Senden* in *Antworten*, um dem Benutzer anzudeuten, dass es sich bei seinem Kommentar entweder um eine neue Nachricht handelt oder eine Antwort auf eine bereits bestehende Nachricht. Eine Antwort wird direkt unter der Ursprungsnachricht als eingerücktes Element in der Liste angezeigt.

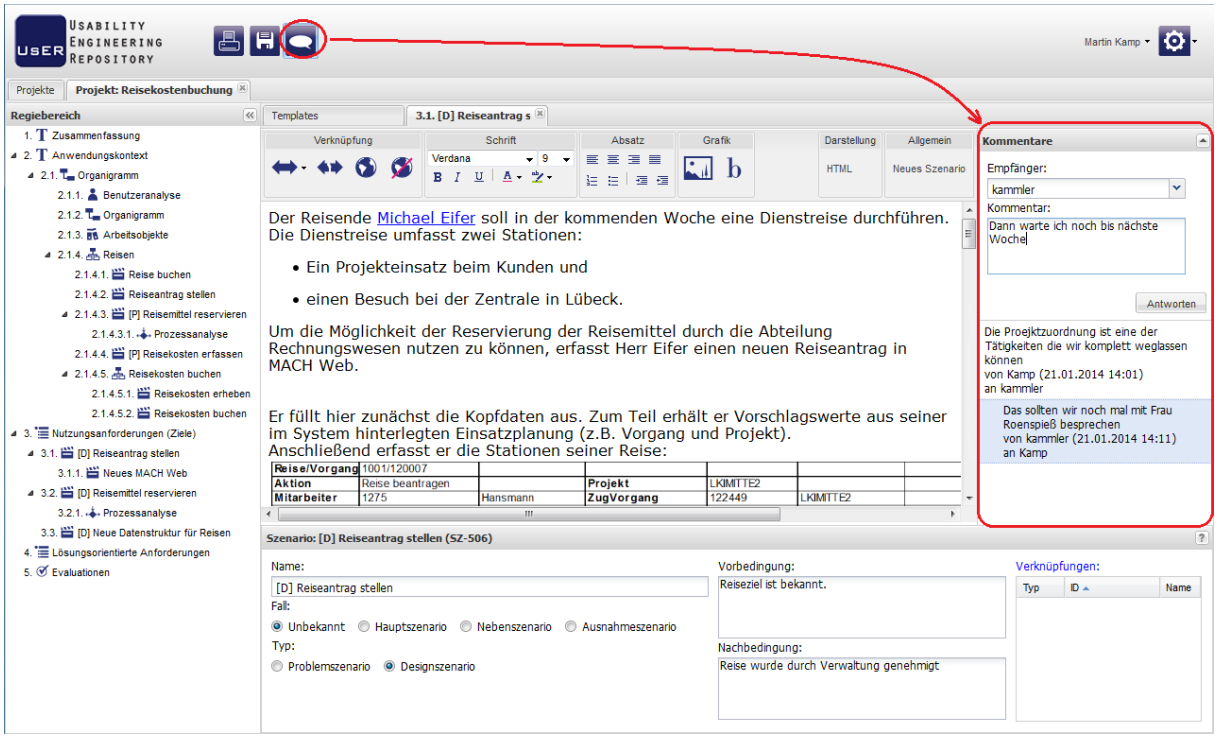


Abbildung 106: Geöffneter Kommentarbereich

Die Ursprungsidee dieser Art der formativen Evaluation lässt sich zurückführen auf das Bewertungsverfahren des Szenario-based Design (Rosson & Carroll, 2002) durch Claims (siehe Kapitel 2.2.4). Aus diesem Grund stammen die beiden nachfolgenden Fragen (siehe Abbildung 107) auch aus der Benutzerbefragung, die im Rahmen der Entwicklung des Szenarien-Moduls durchgeführt wurden. Während der Konzeption des Moduls stellte sich heraus, dass diese Art der Bewertung durch Kommentare auch für alle anderen Entitäten geeignet ist, die im Rahmen eines Entwicklungsprozesses erhoben werden. Aus diesem Grund wurde diese Funktion als globale Funktion für alle Module realisiert.

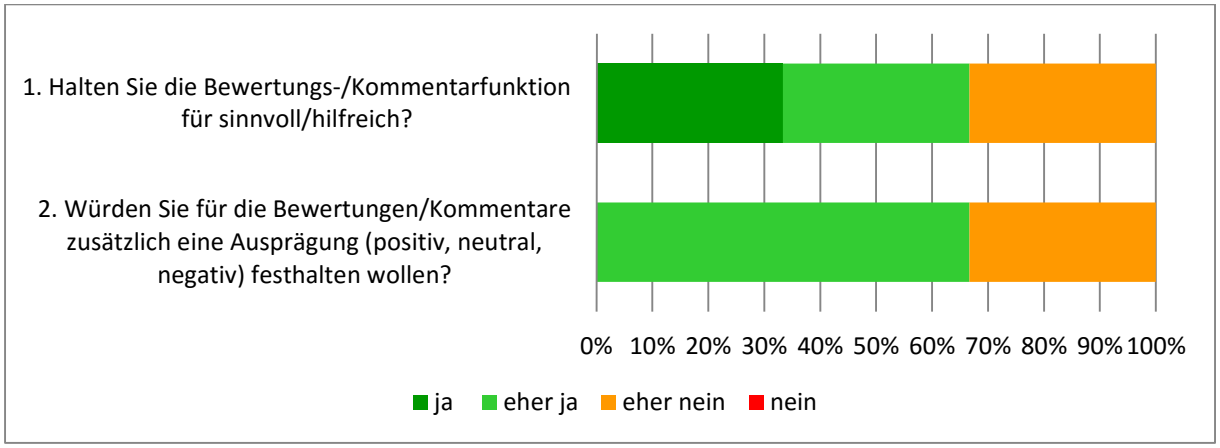


Abbildung 107: Evaluationsergebnisse Kommentarfunktion

Da bei dem Bewertungsverfahren von Rosson und Carroll (2002) durch Claims diese zusätzlich noch als positiv, neutral oder negativ ausgezeichnet werden können, wollten wir bei der Befragung wissen, ob sich diese Ausprägung auch auf die von uns umgesetzte Kommentarfunktion übertragen lassen könnte. Da das Ergebnis jedoch nicht sehr aussagekräftig war, entschieden wir uns dafür, die Komplexität der gesamten Anwendung dadurch nicht noch weiter zu erhöhen. Entsprechende Entwürfe sind jedoch in der Diplomarbeit von Kathrin Beholz (2012) umgesetzt worden.

5.5.2 Evaluationsmodul

Um summative Evaluationen zu einer Gestaltungslösung durchzuführen, besitzt UsER das sogenannte *Evaluationsmodul*. Dieses ermöglicht die Erstellung projektübergreifender Fragebögen. Über das Menü der Fragebogenverwaltung (siehe Abbildung 108) kann ein projektübergreifender neuer Fragebogen erstellt werden.

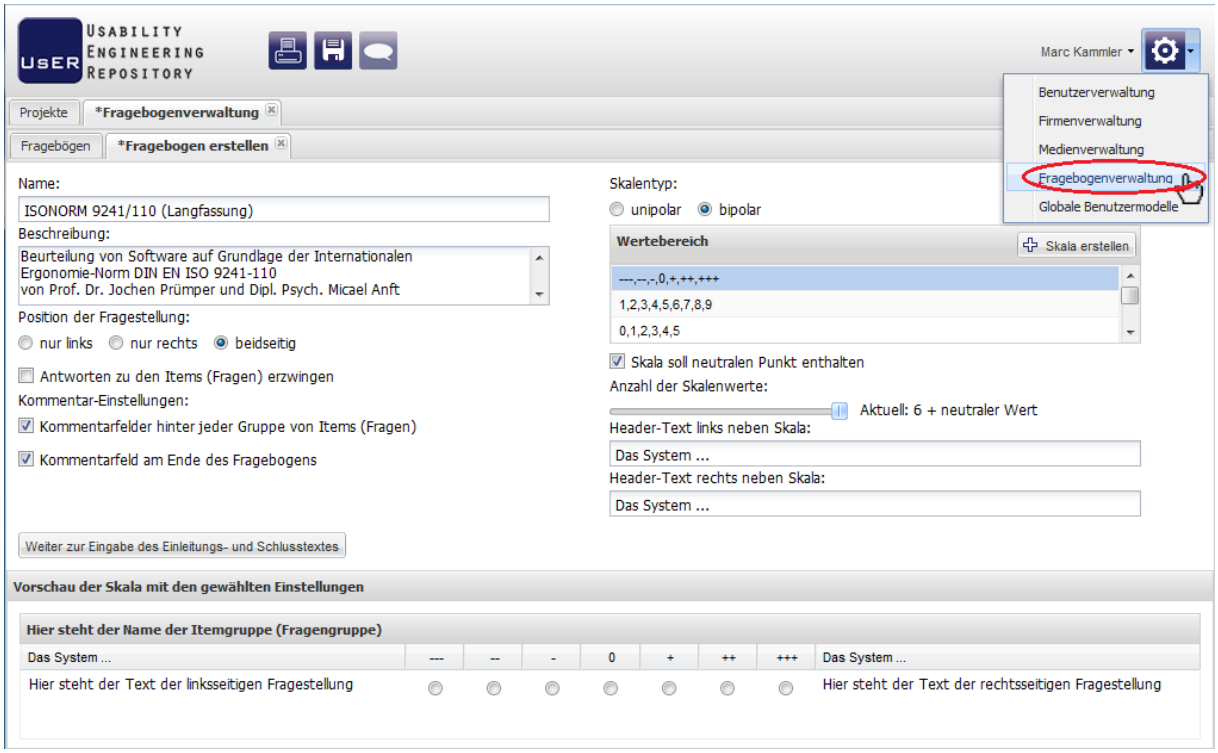


Abbildung 108: Erstellung von Fragebögen in UsER

In dem Erstellungsmenü aus Abbildung 108 muss der Name des Fragebogens angegeben werden. Zur weiteren Konkretisierung kann er optional weiter beschrieben werden. Zudem lässt sich hier der Skalentyp als unipolar oder bipolar wählen und wie viele Werte dieser enthalten soll. Im unteren Bereich wird eine Vorschau der entsprechenden Werte angezeigt. Über die Schaltfläche *Weiter zur Eingabe des Einleitungs- und Schlusstextes* gelangt der Benutzer zur zweiten Ansicht. Diese enthält zwei große Textfelder zur Eingabe des Textes, der vor dem eigentlichen Fragebogen als Startseite angezeigt wird, und den Schlusstext am Ende des Fragebogens. Über eine Menüleiste oberhalb der Textfelder können noch weitere Textbearbeitungsfunktionen wie Größe, Farbe und Schriftauszeichnungsarten aufgerufen werden.

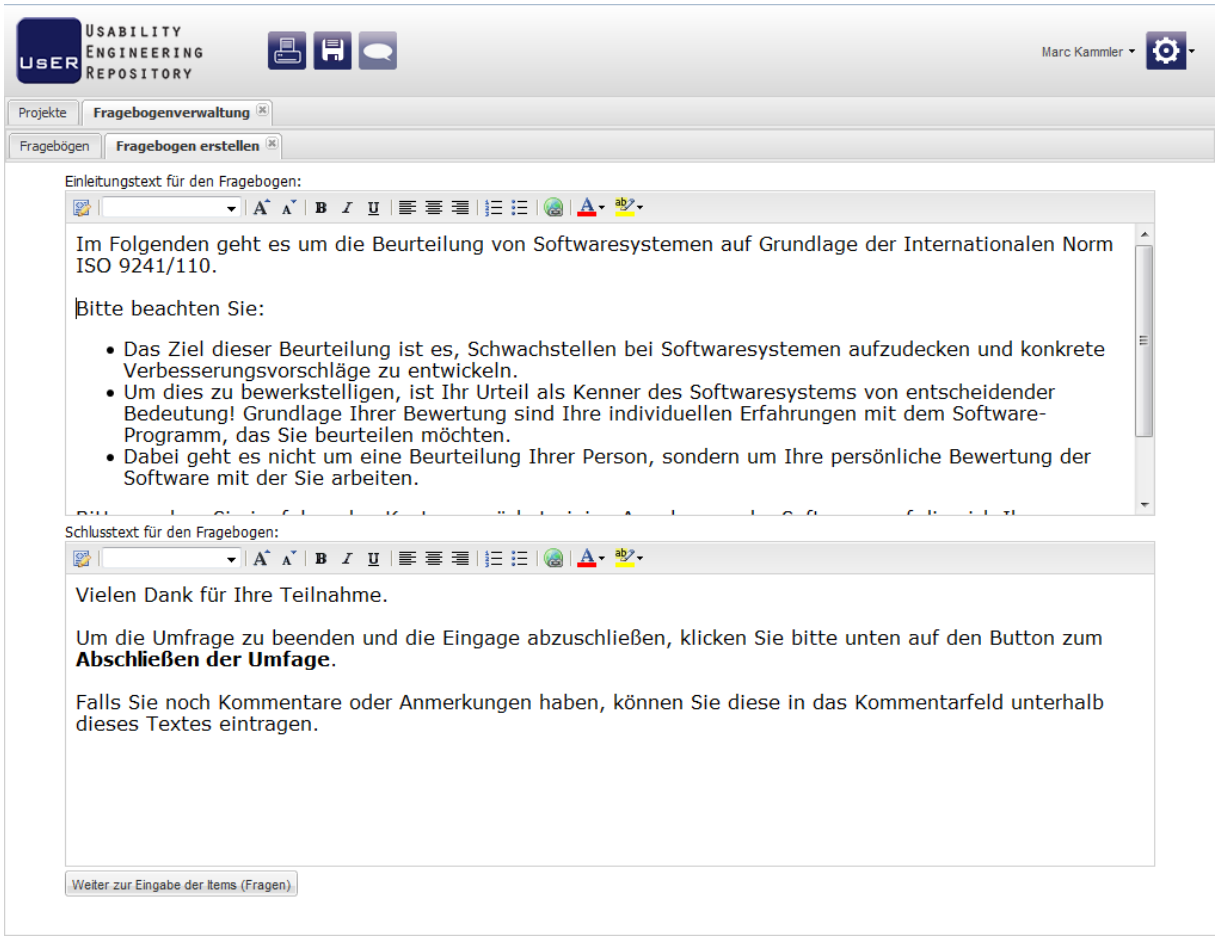


Abbildung 109: Eingabe des Anfangs- und Schlusstextes bei der Fragebogenerstellung

Über die Schalfläche „*Weiter zur Eingabe der Items (Fragen)*“ kann der eigentliche Fragenkatalog erstellt werden (siehe Abbildung 110). Dieser ermöglicht, je nach gewähltem Skalentyp, die Eingabe der Frageitems. Die Fragen können in Gruppen sortiert oder wie bei einem Papierfragebogen auf verschiedenen Seiten verteilt werden. Die Reihenfolge der Items kann durch Drag & Drop des jeweiligen Items geändert werden. Zudem besteht die Möglichkeit, weitere Textfelder hinzuzufügen, um den Probanden die Eingabe weiterer Kommentare zu erlauben.

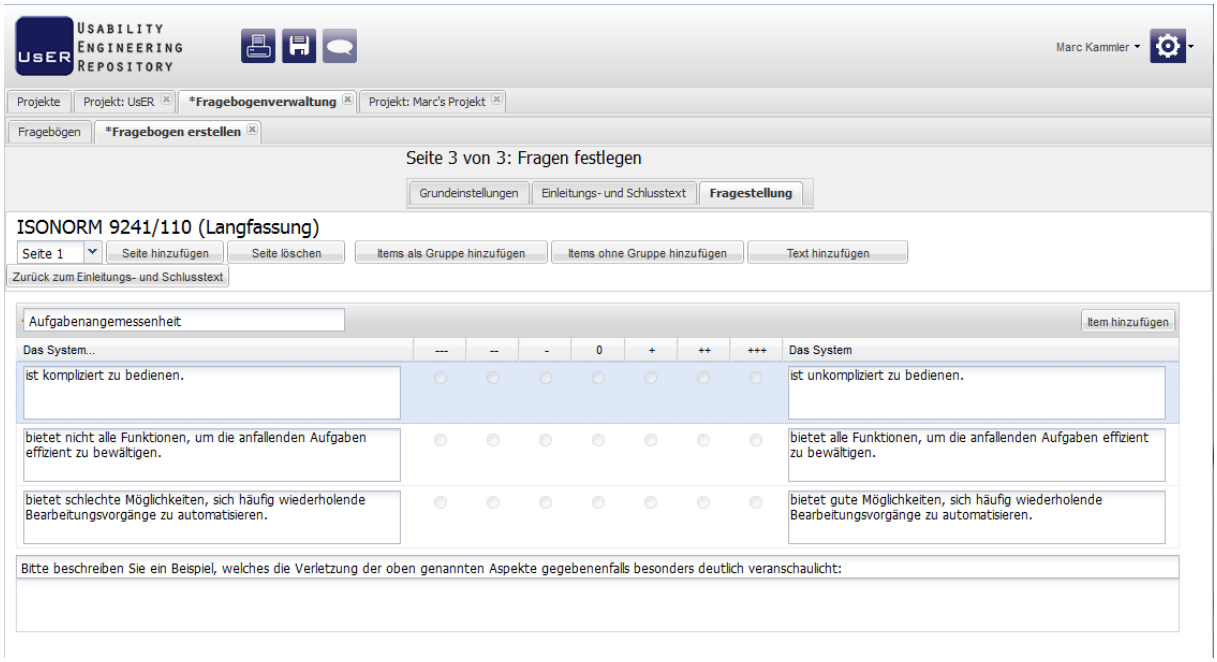


Abbildung 110: Eingabe der Fragen

Um innerhalb eines Projektes eine summative Evaluation zu starten, muss dem Projekt das *Evaluationsmodul* als Kapitelelement hinzugefügt werden. Dieses wird auf die gleiche Weise, wie bei den anderen Modulen, in den Regiebereich des Projektes gezogen. Anschließend kann einer der zuvor angelegten Fragebögen ausgewählt und die Evaluation gestartet werden (siehe Abbildung 111). Dazu wird von UsER eine E-Mail an die Probanden verschickt, in der eine URL enthalten ist, über die an der Evaluation teilgenommen werden kann.

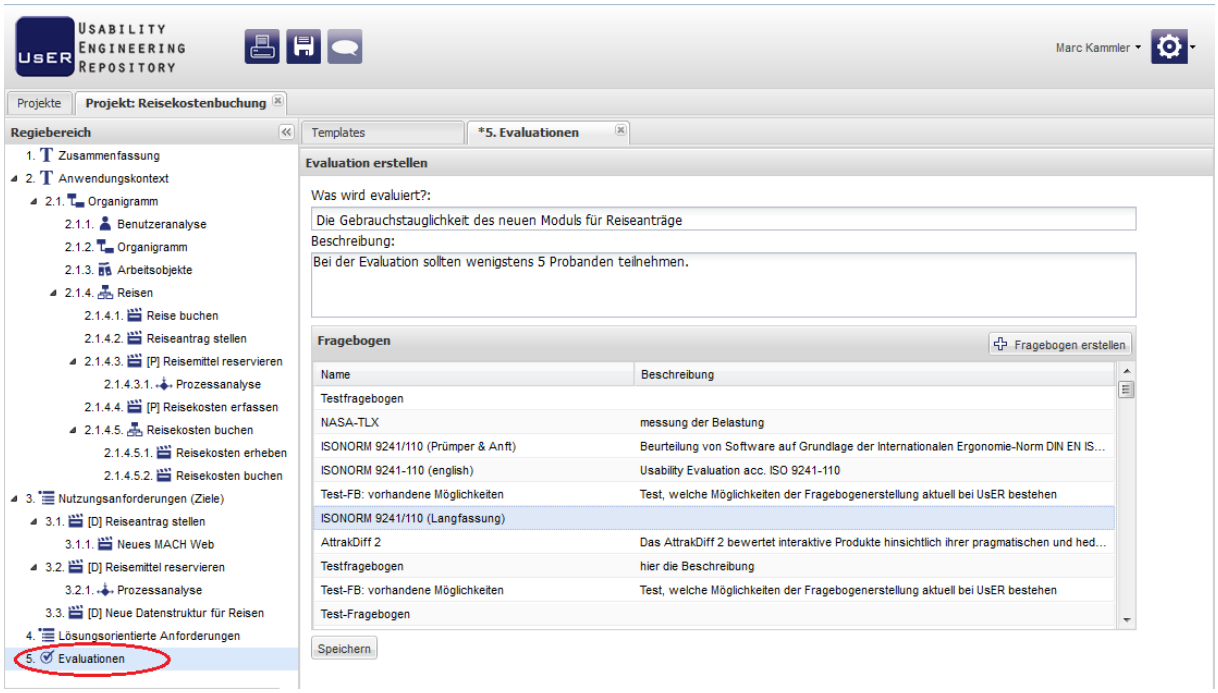


Abbildung 111: Initiieren einer summativen Evaluation

Innerhalb des Evaluationsmoduls können auf diese Weise beliebig viele verschiedene Evaluationen initiiert werden, wodurch sämtliche Gestaltungslösungen eines Projektes von unterschiedlichen Probanden bewertet werden können. Die Ergebnisse der Evaluationen sind jederzeit in dem jeweiligen Projekt einzusehen. In dem Beispiel

aus Abbildung 112 wurde der zuvor erstellte Fragebogen bereits von drei Teilnehmern beantwortet. In einer weiteren Evaluation soll die Beanspruchung von Benutzern bei der Arbeit durch einen NASA-TLX-Test ermittelt werden.

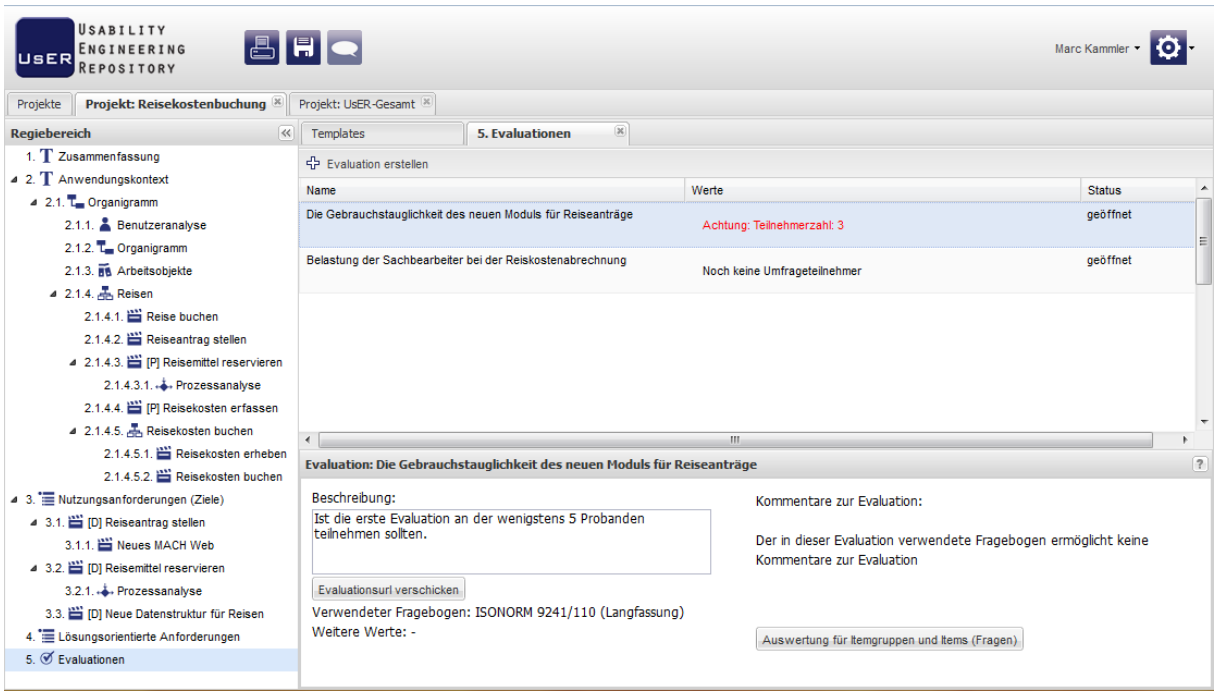


Abbildung 112: Übersicht der in einem Projekt durchgeführten Evaluationen

Um sich die bisherigen Ergebnisse der ersten Evaluation durch den ISONORM-Fragebogen im Detail anzuschauen, muss dieser markiert werden. Im Detailbereich werden dann zusätzliche Informationen zu der jeweiligen Evaluation angezeigt. Über die Schaltfläche *„Auswertung für Itemgruppen und Items (Fragen)“* können die Ergebnisse im Detail analysiert werden. Es wird eine Liste mit Einzelergebnissen aller Items eines Fragebogens angezeigt (siehe Abbildung 113). Über das Menü *Ergebnisdarstellung* kann ausgewählt werden, ob die Ergebnisse als horizontales, vertikales Balkendiagramm oder als Tortendiagramm angezeigt werden soll. In Abbildung 113 ist die Darstellung der Ergebnisse als Balkendiagramm angeführt.

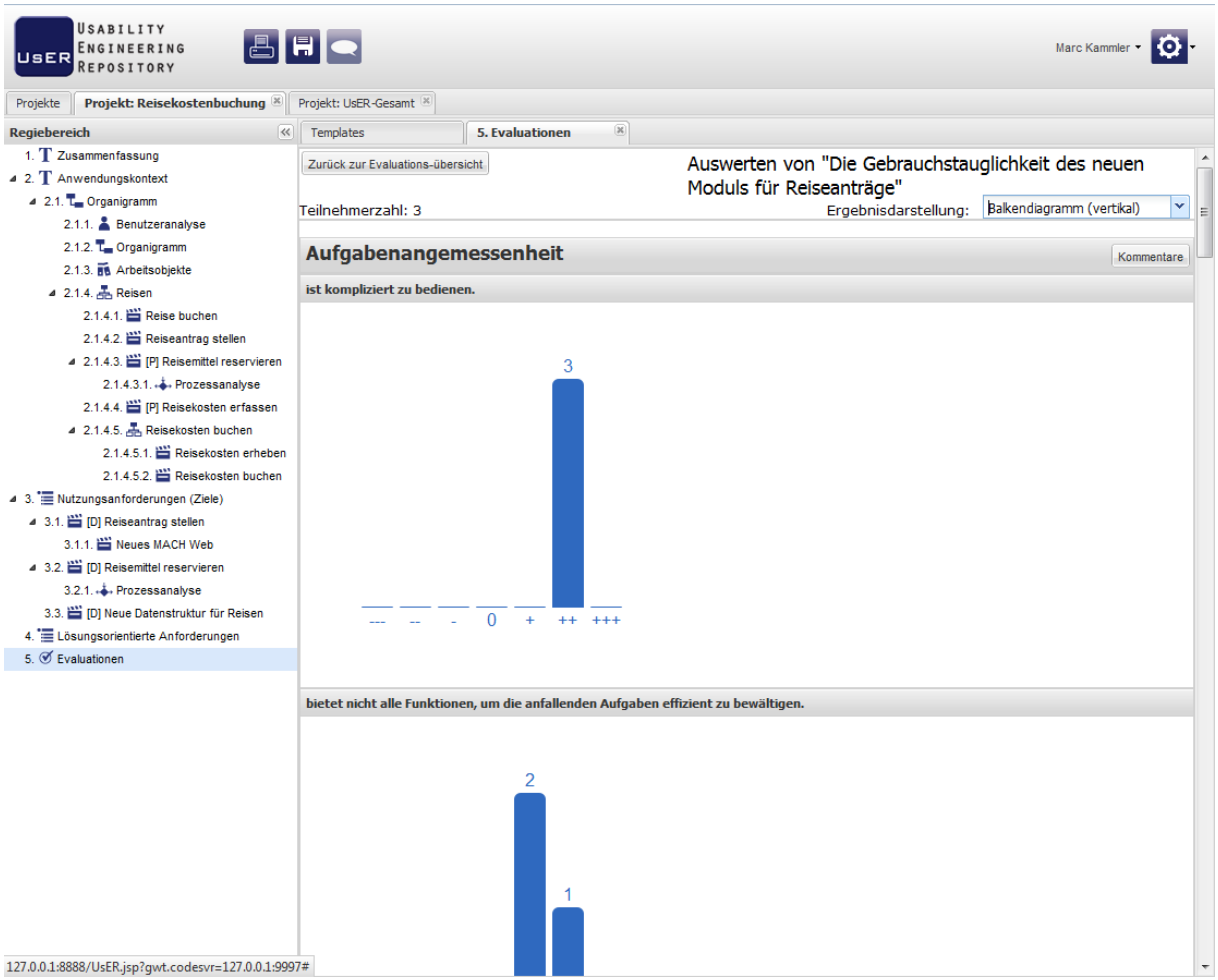


Abbildung 113: Anzeige der Einzelergebnisse aller Items eines Fragebogens

Im Rahmen einer summativen Evaluation mit sechs Teilnehmern wurde den Probanden erst die Funktionsweise des Moduls an einem durchgehenden Beispiel veranschaulicht (Berndt, 1012). Anschließend wurden die Probanden gebeten, den Fragebogen ISONORM 9241/110-S (Prümper, 2014) zu beantworten. Dieser verhilft zu einer Aussage über die Gebrauchstauglichkeit durch folgende Punkte:

- Aufgabenangemessenheit (Fragen 01-03)
- Selbstbeschreibungsfähigkeit (Fragen 04-06)
- Erwartungskonformität (Fragen 07-09)
- Lernförderlichkeit (Fragen 10-12)
- Steuerbarkeit (Fragen 13-15)
- Fehlertoleranz (Fragen 16-18)
- Individualisierbarkeit (Fragen 19-21)

Zugunsten der Einfachheit ist in den beiden nachfolgenden Diagrammen (siehe Abbildung 114 und Abbildung 115) zu jedem Frageitem nur die rechte Seite des im Original als bipolar aufgebauten Fragebogens dargestellt. Die Probanden hatten dabei die Möglichkeit die Fragen mit „ist schlecht“ (---) über „neutral“ (-/+) bis hin zu „ist gut“ (+++) zu beantworten. Aus den ersten drei Fragen ergibt sich, dass das Modul bei seiner angedachten Aufgabe den Probanden angemessen unterstützt. Bei der Selbstbeschreibungsfähigkeit (Fragen 4 bis 6) kann aus den Antworten abgeleitet werden, dass die Probanden sich weitere situationsspezifische Erklärungen des Systems zu einzelnen Funktionalitäten gewünscht hätten. Dem Großteil der Probanden wurden jedoch die Funktionalitäten geboten, die sie für ein Evaluationsmodul erwartet hätten (Fragen 7 bis 9).

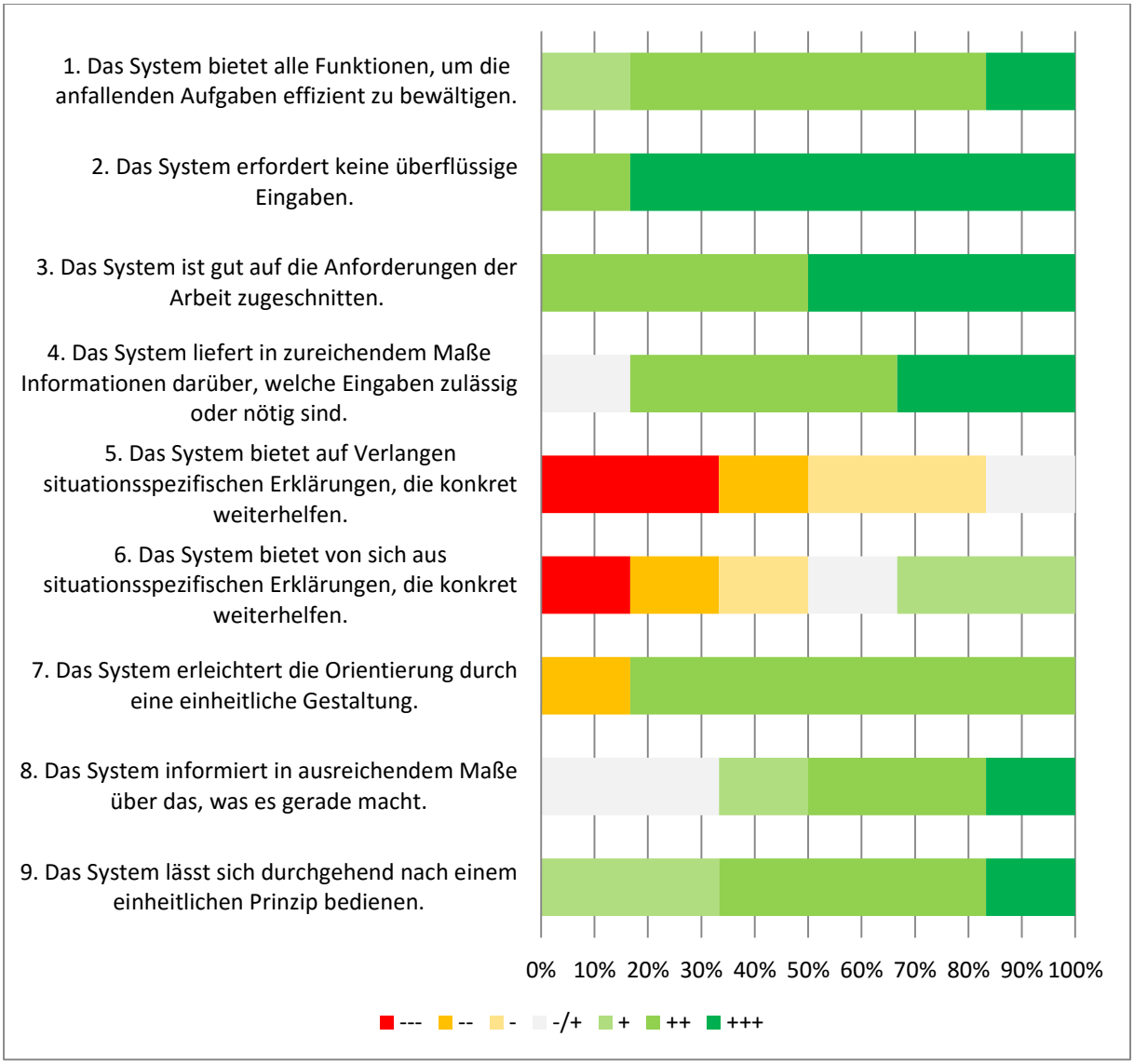


Abbildung 114: Evaluationsergebnisse Evaluationsmodul 1/2

In Bezug auf die Lernförderlichkeit (Fragen 10 bis 12) zeigte sich, dass das Modul gut ohne weitere Hilfen verwendbar ist. Die Steuerbarkeit (Fragen 13 bis 15) scheint einigen Probanden zu unflexibel zu sein, was sich auch in Bezug auf die Fehlertoleranz (Fragen 16 bis 18) widerspiegelt. Fragen, die sich auf die Individualisierbarkeit (Fragen 19 bis 21) bezogen, wurden überwiegend neutral bewertet. Die formative Evaluation zeigt noch ein weites Feld für Verbesserungen dieses Moduls.

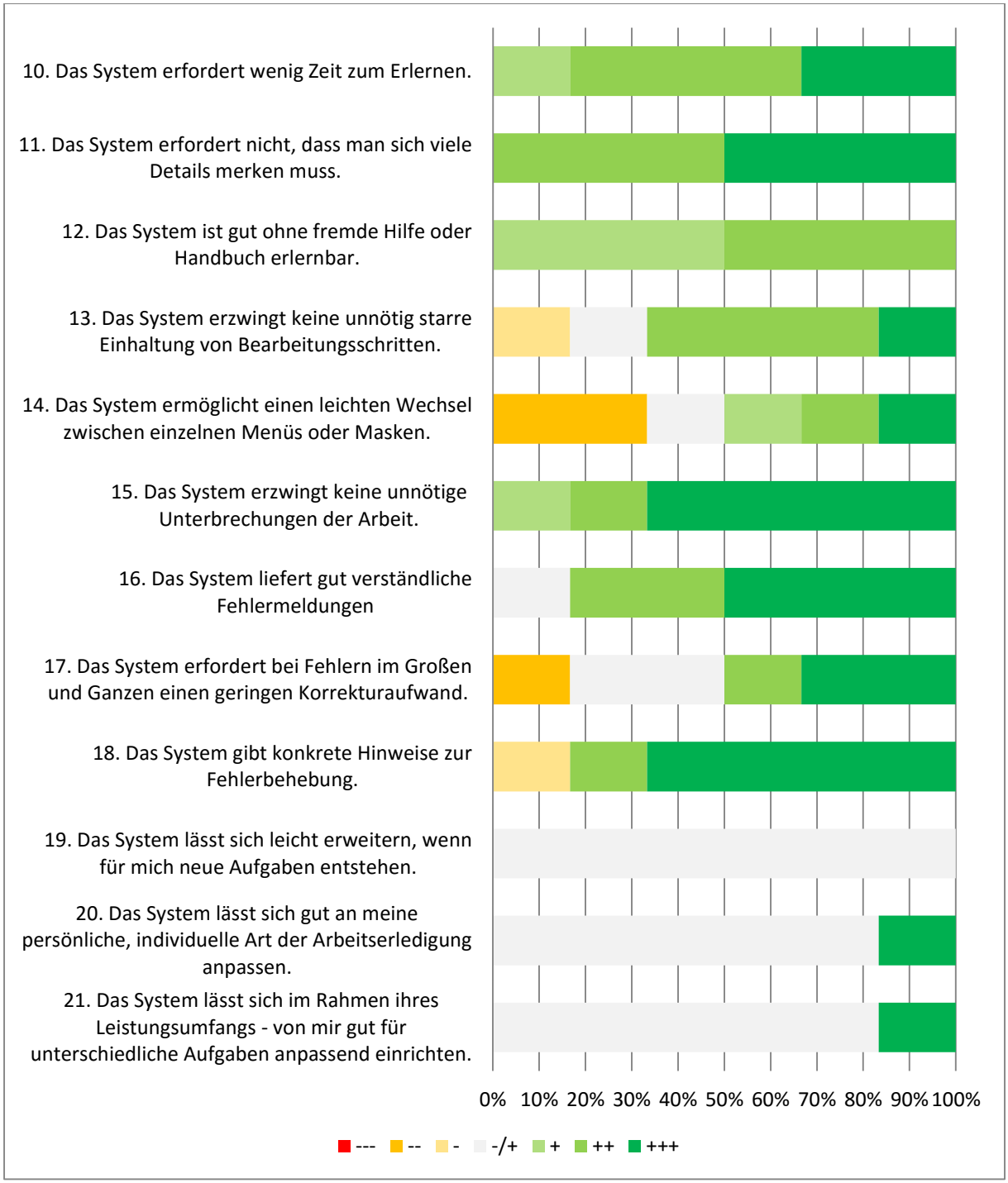


Abbildung 115: Evaluationsergebnisse Evaluationsmodul 2/2

6 Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit wurden unterschiedliche Prozesse, Methoden und Ansätze aus dem Bereich des UE in Kooperation mit einem Software- und Beratungshaus analysiert, bewertet und in unterschiedlichen Praxistests realer Kunden bei Software-Entwicklungsprojekten des Kooperationspartners erprobt. Erkenntnisse dieser Untersuchungen wurden optimiert, iterativ verfeinert und in Form von Software-Werkzeugen in dem hierfür entwickelten System UsER (Usability-Engineering-Repository) verbunden. Erkenntnisse dieser Praxistests wurden dazu verwendet, eine Vielzahl von Modulen zu entwickeln und umzusetzen, die sich in das Software-Engineering (SE) integrieren lassen. Die entstandenen Module sind Teilkomponenten der im Rahmen dieser Arbeit entstandenen Kollaborationsplattform, die sich auf beliebige Weise miteinander kombinieren lassen. Das System mit einem Code von inzwischen über 10 000 Zeilen ermöglicht über diese Arbeit hinaus eine gute Grundlage für die Entwicklung weiterer Werkzeugmodule. Es wurde gezeigt, wie der Prozess zur Entwicklung gebrauchstauglicher interaktiver Systeme auf Basis der DIN EN ISO 9241-210 praktisch und methodisch durch die modulare webbasierte Kollaborationsplattform UsER umgesetzt werden kann. Es wurde ein Lösungsansatz für eine Systemunterstützung aufgezeigt, die es ermöglicht Informationen wiederzuverwenden und miteinander zu kombinieren. Der in Abbildung 116 dargestellte Prozess verdeutlicht diesen Ansatz und vereinigt darüber hinaus in unüblicher Weise die beiden Bereiche UE und SE, um eine systematische planbare Umsetzung gewährleisten zu können. UsER ist ein prozessneutrales Werkzeug zur benutzerzentrierten Entwicklung von Softwaresystemen. Klassische wie auch agile Entwicklungsprozesse lassen sich damit flexibel unterstützen. Das UE stört damit in keiner Weise bewährte Entwicklungsprozesse und erlaubt so gewissermaßen die Nachrüstung von SE um UE.

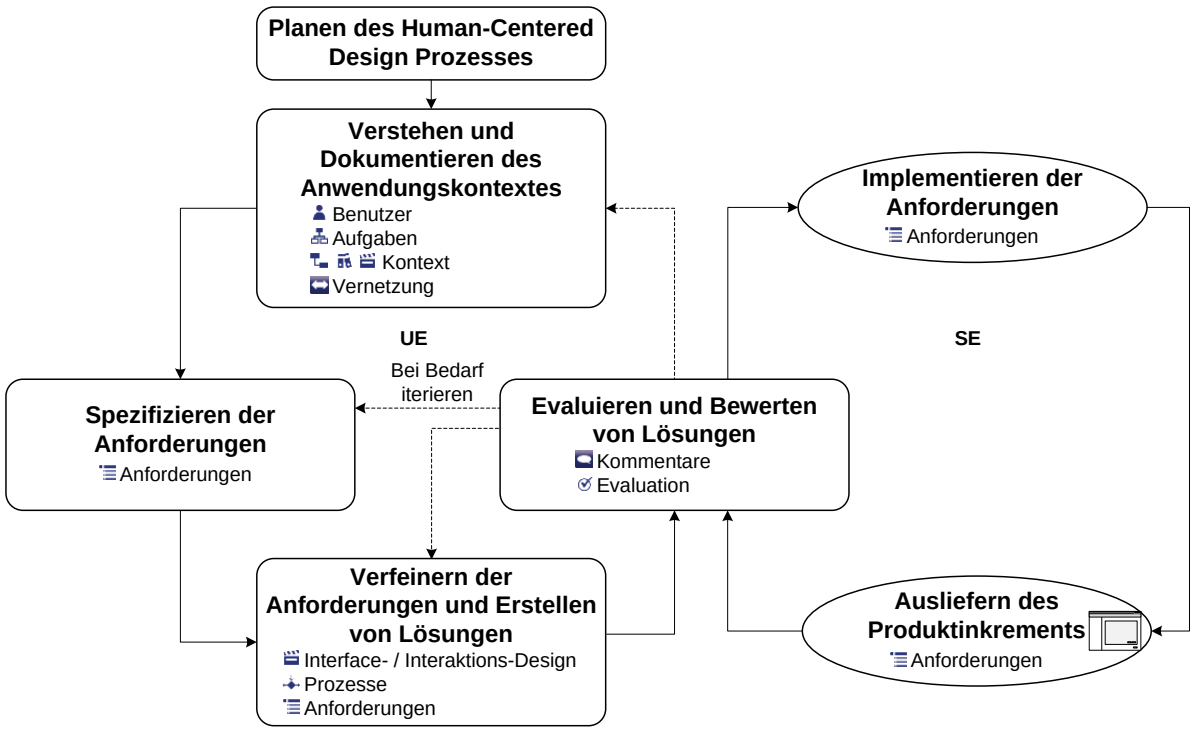


Abbildung 116: Erweiterter Human-centered Design Prozess (Paul, Roenspieß & Herczeg, 2013)

Der hier aufgezeigte Prozess lässt sich gut mit den Worten von John Johnson zusammenfassen:

> *„First, solve the problem. Then, write the code.“* (John Johnson, zit. nach Esposito & Saltarello, 2009)

In einem Punkt geht der Prozess sogar noch einen Schritt weiter. Bevor ein Problem gelöst werden kann, muss dieses erst erkannt werden. Die Erkenntnisse im Rahmen dieser Arbeit haben gezeigt, dass eben dieser wichtige Schritt häufig übergangen wird. Funktionalitäten, die ohne diesen Schritt entwickelt werden, lösen kein Problem, sondern sind Ursache für neue. Das UE bietet eine Vielzahl verschiedener Analyseverfahren, die diesem Problem

begegnen. Viele wirken jedoch sehr schwerfällig und zeitaufwändig, wodurch zahlreiche Unternehmen den vermeintlich größeren Aufwand scheuen. Typischerweise enthalten heute Entwicklungsprozesse keine oder nur einzelne UE-Aktivitäten. Als Schnittstellenelement beider Bereiche dienen die Anforderungen. Sie beschreiben eine Art Kernelement, welches sich durch beide Prozesse hindurch bewegt und dabei verschiedene Reifegrade durchlebt. Angefangen von der Formulierung des eigentlichen Problems über sich daraus ableitende Lösungsansätzen bis hin zur Validierung der Implementierung.

Um die Verwendbarkeit des Prozesses und des Systems zu verbessern, sollten weitere Ansätze aus dem Bereich des SE integriert werden. Neben den Aufgaben für die Neuentwicklung einer Softwarelösung ist ein wesentlicher Bereich des SEs mit der Wartung bereits existierender Systeme beschäftigt. Zu diesem Zweck werden weitere Werkzeugmodule benötigt, die eine dynamische Systemdokumentation ermöglichen. Die Wartung von Systemen wird manchmal durch die Tatsache erschwert, dass bestimmtes Systemwissen ausschließlich in den Köpfen weniger Entwickler vorhanden ist. Stehen diese eines Tages nicht mehr zur Verfügung, ist es häufig schwierig dieses Wissen nachzuvollziehen. Da Neuentwicklungen, Fehlerkorrekturen oder andere geringfügige Änderungen bei der Softwareentwicklung meist innerhalb eines Prozesses behandelt werden, sollte eine Werkzeugunterstützung auch diese wichtige Aufgabe unterstützen, um die Akzeptanz bei der Verwendung dieses Werkzeuges durch die Entwickler weiter zu erhöhen.

Abbildungsverzeichnis

Abbildung 1: Elemente zur Bestimmung des Gebrauchstauglichkeitsgrades (nach ISO 9241-11) 4
Abbildung 2: Überlappung von Disziplinen im UE (nach Saffer, 2007) 5
Abbildung 3: Prozess zur Gestaltung gebrauchstauglicher interaktiver Systeme aus der ISO 9241-210:2011 7
Abbildung 4: Beispiel eines Flow Models (Benyon, 2010) 15
Abbildung 5: Beispiel für ein Cultural Model (Benyon, 2010) 17
Abbildung 6: Vorgehensmodell Szenario-basierter Entwicklung (Rosson & Carroll, 2002, S. 25) 19
Abbildung 7: Beispiel zweier Claims (Rosson & Carroll, 2003) 20
Abbildung 8: Beispiel einer "Context Map" (Constantine, 2006b) 22
Abbildung 9: Überarbeiteter Ausschnitt einer Activity-Task Map (Constantine, 2009) 23
Abbildung 10: Usability Engineering Lifecycle von Deborah Mayhew (www.thinkbrownstone.com) 25
Abbildung 11: Zusammenführung aller Wünsche und Ziele (Quelle: Cooper et al., 2007) 28
Abbildung 12: erweiterte Benutzerklassifikationen (Pruitt & Adlin, 2006) 28
Abbildung 13: Verteilung der benutzerspezifischen Verhaltensvariablen (Cooper et al., 2007, S. 99) 30
Abbildung 14: Hierarchische Struktur von Aktivitäten (Specker, 1998) 31
Abbildung 15: Seven Stages of Action (Norman, 1998, S. 46) 32
Abbildung 16: Modell der Mensch-Computer-Kommunikation (Herczeg, 1994) 33
Abbildung 17: Ausdruckskraft versus Komplexität bei Aufgabenmodellen (Limbourg & Vanderdonckt, 2004) 34
Abbildung 18: Datenmodell der Hierarchischen Aufgaben Analyse (Limbourg & Vanderdonckt, 2004) 34
Abbildung 19: HTA-Beispiel für das Ausleihen eines Buches (Johansson, 2004) 35
Abbildung 20: Datenmodell von MAD* (Limbourg & Vanderdonckt, 2004) 36
Abbildung 21: Datenmodell der Groupware Task Analysis (Limbourg & Vanderdonckt, 2004) 37
Abbildung 22: Datenmodell von MUSE (Limbourg & Vanderdonckt, 2004) 37
Abbildung 23: Datenmodell von ConqurTaskTree (Limbourg & Vanderdonckt, 2004) 38
Abbildung 24: Einfaches Prozessdiagramm 39
Abbildung 25: Organisationsaufteilung nach Funktion oder Sparte 41
Abbildung 26: Modell Organisatorischer Gestaltung (Bleicher, 1991, S. 49) 42
Abbildung 27: Kontextinformationen in Szenarien (angelehnt an Pohl, 2008, S. 125; Rosson & Carroll, 2002) 43
Abbildung 28: Wechselwirkung zwischen Zielen und Szenarien (Pohl, 2008) 44
Abbildung 29: Szenarien als Kommunikationsmittel zwischen der Realität und der abstrakten Welt 45
Abbildung 30: Die zwei Dimensionen von Prototypen (Nielsen, 1993, S. 94) 45
Abbildung 31: Notation eines Use Cases (Object Management Group, 2011b, S. 18) 49
Abbildung 32: Aufbau eines Anwendungsfalls mit Szenarien 50
Abbildung 33: Gestaltungsempfehlungen der Normreihe ISO 9241 (nach ISO 9241-110) 51
Abbildung 34: Goal-Scenario-Coupling (Pohl, 2008) 54

Abbildung 35: Datenmodell eines "Requirement Chunks" (Tawbi & Souveyet, 1999) 55
Abbildung 36: Überblick über das CREWS-Verfahren (Rolland, Souveyet & Achour, 1998) 56
Abbildung 37: Auswahl der Projektphase im Usability Planner (http://www.usabilityplanner.org) 58
Abbildung 38: Concur Task Tree Enviroment (HIIS Laboratory, 2013) 59
Abbildung 39: Screenshot einer Benutzerbefragung aus der User Experience Suite 60
Abbildung 40: Um Methoden erweiterter HCD-Prozess 62
Abbildung 41: Wasserfallmodell nach Royce (Royce, 1970) 63
Abbildung 42: Darstellung des klassischen V-Modells (Boehm, 1979) 64
Abbildung 43: Spiralmodell (Boehm, 1988) 65
Abbildung 44: Aufteilung des RUP-Prozesses (Hruschka, Rupp & Starke, 2009, S. 78) 66
Abbildung 45: Scrum-Prozess angelehnt an Schwaber (2004) und Hanser (2010) 68
Abbildung 46: Mindmapstruktur eines Anforderungsworkshops 69
Abbildung 47: Verschiedene Arten von User Stories im Product-Backlog (Wirdemann, 2011, S. 64) 70
Abbildung 48: Grundlegende Schritte des SE 71
Abbildung 49: Beispiel für ein Use-Case-Diagramm 74
Abbildung 50: Beispiel für ein Aktivitätsdiagramm 75
Abbildung 51: Beispiel für ein Datenflussdiagramm 76
Abbildung 52: Lebenszyklus einer Anforderung (Ebert, 2008, S. 260) 76
Abbildung 53: Ziel-Szenario-Kopplung (Pohl, 2008, S. 276) 78
Abbildung 54: Der agile User Centered Design Prozess (Paelke & Nebe, 2008) 79
Abbildung 55: Phasenaufteilung des KoSSE-Verbundprojektes 83
Abbildung 56: Standard-Software mit Anpassungen für den Kunden 84
Abbildung 57: möglicher Versionszyklus der Software 84
Abbildung 58: Ausgangssituation des bestehenden SE-Prozesses 85
Abbildung 59: Struktur für Dokument bei Anforderungsaufnahmen 86
Abbildung 60: Erste Version des idealisierten Entwicklungsprozesses 89
Abbildung 61: Ausschnitt des Aufgabenbaums eines kommunalen Kunden mit Szenarienzuordnung 91
Abbildung 62: Storyboard im Praxistest 92
Abbildung 63: Mockup-Skizze für die Verknüpfung von Aktivitäten mit Storyboard-Abschnitten 93
Abbildung 64: Screenshot der Flash-Anwendung für die Verknüpfung von Szenarien und Interaktionsfolge 94
Abbildung 65: Die Umsetzung des 6-Ebenen-Modells im Praxistest 95
Abbildung 66: Organisationsanalyse mit einer Aufgabenanalyse gekoppelt 96
Abbildung 67: Mockup aus einem Konzept des Kooperationspartners 97
Abbildung 68: Ergebnis des ISONORM 9241/110 Fragebogens zur Gebrauchstauglichkeit 99
Abbildung 69: Idealisierter agiler UE-Prozess 100
Abbildung 70: Grundidee der Kollaborationsplattform UsER 102

Abbildung 71: generelle Aufteilung von UsER 103
Abbildung 72: Projektverwaltung von UsER 104
Abbildung 73: Projektansicht von UsER 105
Abbildung 74: Generelle Modulaufteilung 105
Abbildung 75: Evaluation des UE-Repository Konzeptes 106
Abbildung 76: Client Server Aufteilung von UsER 108
Abbildung 77: Datenmodell der Kernkomponente von UsER (Stand 01.06.2011) 110
Abbildung 78: Idealisierter UE-Prozess mit UE-Werkzeugen 112
Abbildung 79: Projektübersicht von UsER 113
Abbildung 80: Auswahl der zu einem Projekt gehörigen Teilnehmer 114
Abbildung 81: Nachrichten beteiligter Stakeholder 114
Abbildung 82: Beliebig strukturierbares Spezifikationsdokument 115
Abbildung 83: Übersicht der Benutzer und ihrer Rollen innerhalb eines Projektes 116
Abbildung 84: Einer Rolle einen neuen Benutzer hinzufügen 116
Abbildung 85: Persona Eigenschaften durch Verhaltensvariablen dargestellt 117
Abbildung 86: Ansicht für fiktive Benutzerbeschreibung in Form einer Persona 118
Abbildung 87: Evaluationsergebnisse Benutzeranalysemodul 119
Abbildung 88: Modul für Aufgabenanalysen 120
Abbildung 89: Geöffnetes Verknüpfungsmenü im Aufgabenanalyse-Modul 121
Abbildung 90: Menü der Aufgabenliste 122
Abbildung 91: Evaluationsergebnisse Aufgabenanalysemodul 122
Abbildung 92: Prozessanalysemodul 124
Abbildung 93: Modul für die Artefakt-Analyse 125
Abbildung 94: Modul für Organisationsanalysen 126
Abbildung 95: Mouse-Over- Menü des Organisationsanalyse-Moduls 126
Abbildung 96: Evaluationsergebnisse Organisation-/Artefakt-Modul 127
Abbildung 97: Modul zur Anforderungsverwaltung 128
Abbildung 98: Dekomposition von Anforderungen in UsER 128
Abbildung 99: Beschreibung einer Gestaltungslösung mit Szenarien-Modul 129
Abbildung 100: Verknüpfung von Kontextinformationen mit Texten 130
Abbildung 101: Aufruf des integrierten Mockup-Editors 130
Abbildung 102: Aufruf von Balsamiq in UsER 131
Abbildung 103: Verknüpfung eines Szenarios mit einer Nutzungsanforderung 132
Abbildung 104: Verknüpfungen in beliebigen Kapiteln strukturieren 133
Abbildung 105: Evaluationsergebnisse Szenarien-Modul 134
Abbildung 106: Geöffneter Kommentarbereich 135

Abbildung 107: Evaluationsergebnisse Kommentarfunktion 135
Abbildung 108: Erstellung von Fragebögen in UsER 136
Abbildung 109: Eingabe des Anfangs- und Schlusstextes bei der Fragebogenerstellung 137
Abbildung 110: Eingabe der Fragen 138
Abbildung 111: Initiieren einer summativen Evaluation 138
Abbildung 112: Übersicht der in einem Projekt durchgeführten Evaluationen 139
Abbildung 113: Anzeige der Einzelergebnisse aller Items eines Fragebogens 140
Abbildung 114: Evaluationsergebnisse Evaluationsmodul 1/2 141
Abbildung 115: Evaluationsergebnisse Evaluationsmodul 2/2 142
Abbildung 116: Erweiterter Human-centered Design Prozess (Paul, Roenspieß & Herczeg, 2013) 143

Tabellenverzeichnis

Tabelle 1: Essential Use Case bzw. Task Case (Constantine, 2006b) 24
Tabelle 2: Grafische Basiselemente von BPMN (Object Management Group, 2011a) 39
Tabelle 3: Scrum Rollen 68

Literaturverzeichnis

Abrazhevich, V. (2009). Integration des Usability-Engineering in den Softwareentwicklungsprozess im Hochschulbereich am Beispiel der Erstellung eines Usability-Kriterienkatalogs für Online-Bewerbungssysteme. In S. Fischer (Hrsg.), *Informatik 2009. Im Focus das Leben ; Beiträge der 39. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 28.9. - 2.10.2009 in Lübeck* (GI-Edition : Proceedings 154). Bonn: Ges. für Informatik.

Angermeier, G. (2013). *Definition im Projektmanagement-Glossar des Projekt Magazins. Lastenheft.* Zugriff am 02.08.2013. Verfügbar unter https://www.projektmagazin.de//glossarterm/lastenheft

Annett, J. & Duncan, K. (1967). *Task Analysis and Training Design*. Universitiy of Hull: Department of Psychology.

Atwood, M. E., Bomsdorf, B. & Szwillus, G. (1999). Tool support for task-based user interface design. In *CHI '99 extended abstracts* (S. 169).

Badke-Schaub, P. (2008). *Human Factors. Psychologie sicheren Handelns in Risikobranchen : mit 17 Tabellen*. Heidelberg: Springer.

Balzert, H. (2000). *Lehrbuch der Software-Technik. Software-Entwicklung* (Lehrbücher der Informatik, 2. Auflage). Heidelberg: Spektrum Akademischer Verlag.

Balzert, H. (2009). *Lehrbuch der Softwaretechnik. Basiskonzepte und Requirements Engineering* (Lehrbücher der Informatik, 3. Aufl). Heidelberg [u.a.]: Spektrum, Akad. Verl.

Beck, K. (2000). *Extreme programming. Das Manifest* (Professionelle Softwareentwicklung). München: Addison-Wesley (Die revolutionäre Methode für Softwareentwicklung in kleinen Teams).

Beck, K. & Grenning, J. (2001). *Manifesto for Agile Software Development.* Zugriff am 02.08.2013. Verfügbar unter http://www.agilemanifesto.org/

Beholz, K. (2011). *Angewandte Methoden des Usability Engineerings im Rahmen einer ERP-Standardsoftware*. Studienarbeit, Universität zu Lübeck. Lübeck.

Beholz, K. (2012). *Anforderungsspezifikation durch Szenarien für das Usability Engineering Repository „UsER“*. Diplomarbeit, Universität zu Lübeck. Lübeck.

Behring, A. & Petter, Andreas, Mühläuser, Max. (2009). Towards Integrating Usability and Software Engineering Using the Mapache Approach grating Usability and Software Using the Mapache Approach. In S. Fischer (Hrsg.), *Informatik 2009. Im Focus das Leben ; Beiträge der 39. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 28.9. - 2.10.2009 in Lübeck* (GI-Edition : Proceedings 154). Bonn: Ges. für Informatik.

Benyon, D. (2010). *Designing interactive systems. A comprehensive guide to HCI and interaction design* (2nd ed). Harlow, England: Addison Wesley.

Bergmann, R. & Garrecht, M. (2008). *Organisation und Projektmanagement* (BA KOMPAKT). Heidelberg: Physica-Verlag Heidelberg.

Berndt, H. (1012). *Konzeption und Realisierung eines Evaluations-Moduls für das Usability Engineering Repository „UsER“*. Bachelorarbeit, Universität zu Lübeck. Lübeck.

Bevan, N. (2001). International standards for HCI and usability. *International Journal of Human-Computer Studies, 55* (4), 533–552.

Bevan, N. (2009). Criteria for selecting methods in user centred design. *I-USED*.

Beyer, H. & Holtzblatt, K. (1998). *Contextual design. Defining customer-centered systems*. San Francisco, Calif: Morgan Kaufmann.

Bischhoff, W. (2007). *Zielorientierte Ansätze des Requirements Engineerings*. Seminar, Universität Stuttgart. Stuttgart. Zugriff am 14.08.2013. Verfügbar unter http://www.iste.uni-stuttgart.de/fileadmin/user_upload/iste/se/teaching/courses/hsre/res-WS2007-2008/HSRE-WS0708-Wolfgang_Bischoff-Ziele.pdf

Boehm, B. W. (1979). Guidelines for Verifying and Validating Software Requirements and Design Specifications. *Euro IFIP 79*, 711–719.

Boehm, B. W. (1988). A spiral model of software development and enhancement. *Computer, 21* (5), 61–72.

Breuer, S. & Maier, A. (2009). *Ansätze zur Schließung der Lücke zwischen Software Engineering und Usability Engineering. FSQ-Bericht Nr. 09-08-18*, Software-Qualität, FH Köln.

Bühner, R. (2004). *Betriebswirtschaftliche Organisationslehre* (Oldenbourgs Lehr- und Handbücher der Wirtschafts- und Sozialwissenschaften, 10., bearb. Aufl). München [u.a.]: Oldenbourg.

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

Cockburn, A. & Dieterle, R. (2008). *Use Cases effektiv erstellen. [das Fundament für gute Software-Entwicklung ; Geschäftsprozesse modellieren mit Use Cases ; die Regeln für Use Cases sicher beherrschen]* (1. Nachdr). Heidelberg: Mitp/REDLINE.

Cohn, M. (2004). *User stories applied. For agile software development* (Addison-Wesley signature series). Boston, Mass. [u.a.]: Addison-Wesley.

Cohn, M. (2012). *2011 CHAOS Report Says Agile is More Successful Than Waterfall | Mountain Goat Software.* Zugriff am 13.08.2013. Verfügbar unter http://www.mountaingoatsoftware.com/blog/agile-succeeds-three-times-more-often-than-waterfall

Constantine, L. (1995). Essential modeling: use cases for user interfaces. *interactions, 2* (2), 34–46.

Constantine, L., Biddle, R. & Noble, J. (2003). Usage-Centered Design and Software Engineering: Models for Integration. In Rick Kazman, Len Bass & Jan Bosch (Hrsg.), *Proceedings of ICSE 2003 Workshop on Bridging the Gaps Between Software Engineering and Human-Computer Interaction, May 3-4, 2003, Portland, Oregon, USA* (S. 106–113). IFIP.

Constantine, L. & Lockwood, L. (1999). *Software for use. A practical guide to the models and methods of usage-centered design*. Boston, MA: Addison Wesley.

Constantine, L. (2006a). *Activity Modeling: Toward a Pragmatic Integration of Activity Theory with Usage-Centered Design*, Laboratory for Usage-centered Software Engineering.

Constantine, L. (2006b). Users, Roles, and Personas. In J. Pruitt & T. Adlin (Hrsg.), *The persona lifecycle. Keeping people in mind throughout product design* (The Morgan Kaufmann series in interactive technologies, S. 499–519). Amsterdam: Elsevier; Morgan Kaufmann Publishers, an imprint of Elsevier.

Constantine, L. (2009). Human Activity Modeling:. Toward A Pragmatic Integration of Activity Theory and Usage-Centered Design. In J. Karat, J. Vanderdonckt, G. Abowd, G. Calvary, J. M. Carroll, G. Cockton et al. (Hrsg.), *Human-Centered Software Engineering* (Human-Computer Interaction Series, S. 27–51). London: Springer London.

Constantine, L. (2011). Activity-Centered Interaction Design: A Model-Driven Approach. In D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell et al. (Hrsg.), *Human-Computer Interaction – INTERACT 2011* (Lecture Notes in Computer Science, Bd. 6949, S. 696–697). Berlin, Heidelberg: Springer Berlin Heidelberg.

Cooper, A. (1999). *The inmates are running the asylum*. Indianapolis, IN: Sams.

Cooper, A., Reimann, R. & Cronin, D. (2007). *About face 3. The essentials of interaction design* [3rd ed.], Completely rev. & updated). Indianapolis, IN: Wiley Pub.

Dahm, M. (2006). *Grundlagen der Mensch-Computer-Interaktion* (Informatik : Software-Ergonomie). München [u.a.]: Pearson Studium.

Demarco, T. (1979). *Structured analysis and system specification* (Yourdon press computing series). Englewood Cliffs: Yourdon Press.

Diaper, D. & Stanton, N. (Hrsg.). (2004). *The Handbook of Task Analysis for Human-Computer Interaction*. Mahwah: Lawrence Erlbaum Associates.

Dierck, M. (2012). *Tätigkeitsanalyse-Modul für das Usability Engineering Repository„UsER"*. Diplomarbeit, Universität zu Lübeck. Universität zu Lübeck.

Dubbels, T. (2011). *Mockup-Modul für das Usability Engineering Repository „UsER"*. Bachelorarbeit, Universität zu Lübeck. Lübeck.

Duden. (2013). *Duden online.* Zugriff am 09.10.2013. Verfügbar unter http://www.duden.de/suchen/dudenonline/heuristik

Ebert, C. (2008). *Systematisches Requirements-Engineering und Management. Anforderungen ermitteln, spezifizieren, analysieren und verwalten* (2., aktualisierte und erw. Aufl). Heidelberg: Dpunkt-Verl.

Ebert, C. (2010). *Systematisches Requirements-Engineering. Anforderungen ermitteln, spezifizieren, analysieren und verwalten* (3., aktualisierte und erw. Aufl). Heidelberg: dpunkt-Verl.

Eller, B. (2009). *Usability Engineering in der Anwendungsentwicklung. Systematische Integration zur Unterstützung einer nutzerorientierten Entwicklungsarbeit* (Gabler Research : Information Engineering und IV-Controlling, 1. Aufl). Wiesbaden: Gabler.

Engeström, Y. (1999). *Perspectives on activity theory. [chiefly selected contributions from the Second International Congress for Research on Activity Theory, held in 1990 in Lahti, Finland]* (Learning in doing). Cambridge [u.a.]: Cambridge Univ. Press.

Esposito, D. & Saltarello, A. (2009). *Architecting Microsoft.NET solutions for the enterprise*. Redmond, Wash: Microsoft Press.

Essigkrug, A. & Mey, T. (2007). *Rational unified process kompakt* (Kompakt-Reihe, 2. Aufl). München: Elsevier, Spektrum, Akad. Verl.

Ferre, X., Bevan, N. & Escobar, T. A. (2010). UCD method selection with usability planner. In E. Þ. Hvannberg, M. K. Lárusdóttir, A. Blandford, J. Gulliksen, X. Ferre, N. Bevan et al. (Hrsg.), *the 6th Nordic Conference* (S. 829).

Fischer, H., Nebe, K. & Klompmaker, F. (2011). A holistic model for integrating usability engineering and software engineering enriched with marketing activities. In M. Kurosu (Hrsg.), *Human centered design. Second international conference, HCD 2011, held as part of HCI International 2011, Orlando, Fl, USA, July 9-14, 2011 : proceedings* (LNCS sublibrary. SL 3, Information systems and applications, incl. internet/web, and HCI, Bd. 6776, S. 28–37). Heidelberg: Springer.

Fischer, P. & Hofer, P. (2011). *Lexikon der Informatik* (15., überarb. Aufl). Berlin: Springer.

Floyd, C. (1984). A Systematic Look at Prototyping. In R. Budde, K. Kuhlenkamp & L. Mathiassen (Hrsg.), *Approaches to prototyping. Proceedings of the Working conference on prototyping, Namur 1983. Working conference on prototyping 1983*. Berlin: Springer,1984.

Gantt, H. L. (1910). *Work, wages, and profits. Their influence on the cost of living*. New York: The Engineering Magazine.

Gay, G. & Hembrooke, H. (2004). *Activity-centered design. An ecological approach to designing smart tools and usable systems* (Acting with technology). Cambridge, Mass: MIT Press.

Gebhardt, K. (2012, 12. Januar). *Das QM-Lexikon - Lexikon.* Zugriff am 06.08.2013. Verfügbar unter http://www.quality.de/lexikon.htm

Gifford, B. R. & Enyedy, N. D. (1999). Activity centered design: towards a theoretical framework for CSCL. In *Proceedings of the 1999 conference on Computer support for collaborative learning* (CSCL '99). International Society of the Learning Sciences.

Göpfert, J. & Lindenbach, H. (2013). *Geschäftsprozessmodellierung mit BPMN 2.0. Business Process Model and Notation*. München: Oldenbourg.

Grande, M. (2011). *100 Minuten für Anforderungsmanagement. Kompaktes Wissen nicht nur für Projektleiter und Entwickler* (1. Aufl). Wiesbaden: Vieweg+Teubner Verlag / Springer Fachmedien Wiesbaden GmbH, Wiesbaden.

Grass, A. (2009). *Dreamteam BPMN 2 und UML*, Elmshorn.

Grechenig, T., Bernhart, M., Breiteneder, R. & Kappel, K. (2010). *Softwaretechnik. Mit Fallbeispielen aus realen Entwicklungsprojekten* (Informatik). München: Pearson Studium.

Hackos, J. T. & Redish, J. (1998). *User and task analysis for interface design* (Wiley computer publishing). New York: Wiley.

Haid, B. (2011). *Benutzer- und Kontextanalyse-Modul für das Usability Engineering Repository „ UsER "*. Diplomarbeit, Universität zu Lübeck. Lübeck.

Hanser, E. (2010). *Agile Prozesse: von XP über Scrum bis MAP* (eXamen.press). Berlin [u.a.]: Springer.

Herczeg, M. (1994). *Software-Ergonomie. Grundlagen der Mensch-Computer-Kommunikation* (1. Aufl). Bonn [u.a.]: Addison-Wesley.

Herczeg, M. (2006). *Interaktionsdesign. Gestaltung interaktiver und multimedialer Systeme* (Lehrbücher interaktive Medien). München [u.a.]: Oldenbourg.

Herczeg, M. (2009). *Software-Ergonomie. Theorien, Modelle und Kriterien für gebrauchstaugliche interaktive Computersysteme* (Lehrbuchreihe interaktive Medien, 3., vollst. überarb. und erw. Aufl). München: Oldenbourg.

HIIS Laboratory. (2013). *Human Interface in Information Systems (HIIS) Laboratory*, Isttituto di Scienza e Tecnologie. Zugriff am 22.10.2013. Verfügbar unter http://giove.cnuce.cnr.it/ctte.html

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

Holtzblatt, K. (2008). Contextual Design. In A. Sears & J. A. Jacko (Hrsg.), *The human-computer interaction handbook. Fundamentals, evolving technologies, and emerging applications* (Human factors and ergonomics, 2nd ed). New York: Lawrence Erlbaum Associates.

Hruschka, P., Rupp, C. & Starke, G. (2009). *Agility kompakt. Tipps für erfolgreiche Systementwicklung* (2nd ed). Dordrecht: Springer.

Hull, E., Jackson, K. & Dick, J. (2005). *Requirements engineering* (2nd. ed). London [etc.]: Springer.

Hüttig, A. (2012). *Entwicklung eines Moduls zur Benutzeranalyse für das Usability-Engineering Repository U-sER*, Universität zu Lübeck. Lübeck.

IBM Corporation. (2013). *IBM Design: What is user experience design?* Zugriff am 01.08.2013. Verfügbar unter http://www-01.ibm.com/software/ucd/designconcepts/whatisUXD.html

Jacobson, I. (1992). *Object-oriented software engineering. A use case driven approach*. [New York]: ACM Press; Addison-Wesley Pub.

Janeiro, J., Barbosa, S. D. J., Springer, T. & Schill, A. (2009). *SIGDOC'09. Proceedings of the 27th ACM International Conference on Design of Communication, Bloomington, Indiana, USA, October 5-7, 2009*. New York: ACM.

Johansson, N., Blomkvist, S. (Mitarbeiter). (2004). *Users, tasks and requirements. Part 3: Task Analysis*, Department of Information Technology. Zugriff am 02.08.2013.

Kammler, M., Roenspieß, A. & Herczeg, M. (2012). *Ein modulares Usability-Engineering-Repository*, Konstanz. Zugriff am 03.02.2014.

Kane, D. (2003). Finding a place for discount usability engineering in agile development: throwing down the gauntlet. In *Proceedings of the Agile Development Conference, 2003. ADC 2003* (S. 40–46). Los Alamitos, Calif: IEEE Computer Society.

Kaptelinin, V. & Nardi, B. A. (1997). Activity theory. Basic Concepts and Applications. In *CHI '97 extended abstracts* (S. 158–159).

Kirwan, B. & Ainsworth, L. K. (1992). *A Guide to task analysis*. London: Taylor & Francis.

Kistner, K.-P. & Steven, M. (2002). *Betriebswirtschaftslehre im Grundstudium* (Physica-Lehrbuch, 4., aktualisierte Aufl). Heidelberg: Physica-Verl.

Laux, H. & Liermann, F. (2005). *Grundlagen der Organisation. Die Steuerung von Entscheidungen als Grundproblem der Betriebswirtschaftslehre ; mit 13 Tabellen* (Springer-Lehrbuch, 6. Aufl). Berlin: Springer.

Leontyev, A. N. (1978). *Activity, Consciousness and Personality:* Prentice-Hall.

Limbourg, Q. & Vanderdonckt, J. (2004). Comparing Task Models for User Interface Design. In D. Diaper & N. Stanton (Hrsg.), *The Handbook of Task Analysis for Human-Computer Interaction.* Mahwah: Lawrence Erlbaum Associates.

Mann, C. & Maurer, F. (2005). A case study on the impact of scrum on overtime and customer satisfaction. In *Agile Conference, 2005. Proceedings* (S. 70–79).

Martin, L. (2008). *User-Centered Design (UCD) and Activity-Centered Design (ACD).* Zugriff am 23.08.2013. Verfügbar unter http://www.sitemotif.com/2008/07/user-centered-design-ucd-and-activity-centered-design-acd/

Mayhew, D. J. (1999). *The usability engineering lifecycle. A practitioner's handbook for user interface design* (The Morgan Kaufmann series in interactive technologies). San Francisco, Calif: Morgan Kaufmann Publishers.

Mayhew, D. J. (2010). *The usability engineering lifecycle. A practitioner's handbook for user interface design* (The Morgan Kaufmann series in interactive technologies, digital print). San Francisco [u.a.]: Morgan Kaufmann.

Mertens, K. R. (2012). *Entwicklungsphasenübergreifendes Management von Anforderungen im Rahmen des Projektes UsER*, Universität zu Lübeck. Lübeck.

Metzker, E. & Offergeld, M. (2001). Computer Aided Improvement of Human-Centered Design Processes. In H. Oberquelle, R. Oppermann & J. Krause (Hrsg.), *1. Fachubergreifende Konferenz. Mensch & Computer 2001* (Berichte des German Chapter of the ACM, Bd. 55, S. 375–384). Stuttgart: B.G. Teubner.

Metzker, E. & Reiterer, H. (2004). Integrating usability engineering in the software development lifecycle based on international standards. In S. D. Barbosa, J. C. Campos, R. Kazman, P. Palanque, M. Harrison & Reeves (Hrsg.), *the 4th ACM SIGCHI symposium* (S. 61–64).

Mikkelson, N. & Lee, W. O. (2000). Incorporating User Archetypes into Scenario-based Design. In *9th Annual Usability Professionals' Association (UPA) Conference*.

Nebe, K., Leuchter, S. & Beck, A. (2009). Integration von Software Engineering und Usability Engineering. In S. Fischer (Hrsg.), *Informatik 2009. Im Focus das Leben ; Beiträge der 39. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 28.9. - 2.10.2009 in Lübeck* (GI-Edition : Proceedings 154, S. 342–347). Bonn: Ges. für Informatik.

Nielsen, J. (1992). Evaluating the thinking-aloud technique for use by computer scientists. In H. R. Hartson (Hrsg.), *Advances in human-computer interaction* (Bd. 3, S. 69–82). Norwood: Ablex.

Nielsen, J. (1993). *Usability engineering*. Cambridge, Mass: AP Professional.

Norman, D. A. & Draper, S. W. (1986). *User centered system design. New perspectives on human-computer interaction*. Hillsdale, N.J: Lawrence Erlbaum.

Norman, D. A. (1998). *The design of everyday things*. London: MIT.

Norman, D. A. (2005). Human-centered design considered harmful. *interactions, 12* (4), 14.

Norman, D. A. (2006). Logic versus usage. The Case for Activity-Centered Design. *interactions, 13* (6), 45.

Oates, R. (2007). *Spring & Hibernate. Eine praxisbezogene Einführung*. München [u.a.]: Hanser.

Obendorf, H. & Finck, M. (2008). Scenario-Based Usability Engineering Techniques in Agile Development. In *CHI '08 extended abstracts on Human factors in computing systems* (S. 2159–2166). New York, NY: ACM.

Object Management Group. (2011a). Business Process Model and Notation. BPMN.

Object Management Group. (2011b). Requirements Interchange Format (ReqIF). OMG.

Paech, B. (2000). *Aufgabenorientierte Softwareentwicklung. Integrierte Gestaltung von Unternehmen, Arbeit und Software ; mit 28 Tabellen*. Berlin [u.a.]: Springer.

Paelke, V. & Nebe, K. (2008). Integrating agile methods for mixed reality design space exploration. In *Proceedings of the 7th ACM conference on Designing interactive systems* (S. 240–249). New York, NY: ACM.

Partsch, H. (2010). *Requirements-Engineering systematisch. Modellbildung für softwaregestützte Systeme* (eXamen.press, 2., überarb. und erw. Aufl). Berlin [u.a.]: Springer.

Paternó, F., Mori, G. & Galiberti, R. (2001). CTTE: An Environment for Analysis and Development of Task Models of Cooperative Applications. In *CHI '01 extended abstracts* (S. 21–22).

Paternó, F. (2004). ConcurTaskTrees: An Engineered Notation for Task Models. In D. Diaper & N. Stanton (Hrsg.), *The Handbook of Task Analysis for Human-Computer Interaction* (S. 483–500). Mahwah: Lawrence Erlbaum Associates.

Pohl, K. (2008). *Requirements Engineering. Grundlagen, Prinzipien, Techniken* (2., korrigierte Aufl). Heidelberg: Dpunkt-Verl.

Preece, J., Rogers, Y. & Sharp, H. (2002). *Interaction design. Beyond human-computer interaction*. New York, NY: J. Wiley & Sons.

Pruitt, J. & Adlin, T. (Hrsg.). (2006). *The persona lifecycle. Keeping people in mind throughout product design* (The Morgan Kaufmann series in interactive technologies). Amsterdam: Elsevier; Morgan Kaufmann Publishers, an imprint of Elsevier.

Pruitt, J. & Grudin, J. (2003). Personas: practice and theory. In *Proceedings of the 2003 conference on Designing for user experiences* (DUX '03, S. 1–15). New York, NY, USA: ACM.

Prümper, J. & Anft, M. (2009, 24. August). *ISONORM 9241/110 (Langfassung). Beurteilung von Software auf Grundlage der Internationalen Ergonomie-Norm DIN EN ISO 9241-110*, HTW-Berlin. Zugriff am 10.10.2013.

Prümper, J. (www.seikumu.de, Hrsg.). (2014). *Fragebogen ISONORM 9241/110-S. Beurteilung von Software auf Grundlage der Internationalen Ergonomie-Norm DIN EN ISO 9241-110*.

Richter, M. & Flückiger, M. D. (2010). Usability Engineering kompakt. Benutzbare Software gezielt entwickeln. *Usability Engineering kompakt*.

Roenspieß, A. (2011). *UsER: Konzeption und Design einer Kollaborationsplattform für betriebliches Usability Engineering*. Masterarbeit, Universität zu Lübeck. Lübeck.

Rolland, C., Ben Achour, C., Cauvet, C., Ralyté, J., Sutcliffe, A., Maiden, N. et al. (1998). A proposal for a scenario classification framework. *Requirements Engineering, 3* (1), 23–47.

Rolland, C., Souveyet, C. & Achour, C. B. (1998). Guiding goal modeling using scenarios. *IEEE Transactions on Software Engineering, 24* (12), 1055–1071.

Root, R. W. & Draper, S. (1983). Questionnaires as a software evaluation tool. In *the SIGCHI conference* (S. 83–87).

Rosemann, M. & Mühlen, M. (1996). *Der Lösungsbeitrag von Metadatenmodellen beim Vergleich von Workflowmanagementsystemen* (Becker, J. & Grob, H. L., Hrsg.). Westfälischen Wilhelms-Universität Münster.

Rosson, M. B. & Carroll, J. M. (2002). *Usability engineering. Scenario-based development of human-computer interaction* (Morgan Kaufmann series in interactive technologies, 1st ed). San Fancisco: Academic Press.

Rosson, M. B. & Carroll, J. M. (2003). Scenario-Based Design. In J. A. Jacko & A. Sears (Hrsg.), *The human-computer interaction handbook. Fundamentals, evolving technologies, and emerging applications* (Human factors and ergonomics, S. 1032–1050). Mahwah, N.J: Lawrence Erlbaum Associates.

Royce, W. (1970). Managing the Development of Large Software Systems. 9th International Conference on Software Engineering, March 30-April 2, 1987, Monterey, California, USA. Reprinted from Proceedings, IEEE WECSON 1970.

Rudd, J., Stern, K. & Isensee, S. (1996). Low vs. high-fidelity prototyping debate. *interactions, 3* (1), 76–85.

Saffer, D. (2007). *Designing for Interaction. Creating Smart Applications and Clever Devices*. Berkeley: New Riders; [Published in association with AIGA Design Press].

Schmid, K. (2009). *Prozess-Optimierung im Output-Management. Prozessmodellierung, Prozessqualität nach ISO/SPICE, ITIL, Organisation, Technologien, Lösungen, Strategien* (1. Aufl). Norderstedt: Books on Demand.

Schwaber, K. (2004). *Agile project management with Scrum*. Redmond, Wash: Microsoft Press.

Shneiderman, B. (1998). *Designing the user interface. Strategies for effective human-computer interaction* (3. ed., reprint. with corr). Reading, Mass. [u.a.]: Addison-Wesley.

Sohaib, O. & Khan, K. (2010). Integrating Usability Engineering and Agile Software Development. A Literature Review. In *International Conference on Computer Design and Applications (ICCDA 2010)* (S. V2-32 - V2-38). Institute of Electrical and Electronics Engineers (IEEE).

Specker, A. (1998). *Kognitives Software-Engineering. Ein schema- und scriptbasierter Ansatz* (Forschungsberichte für die Unternehmenspraxis, Bd. 6). Zürich: Vdf, Hochsch.-Verl. an der ETH.

Sutcliffe, A. (2000). On the effective use and reuse of HCI knowledge. *ACM Trans. Comput.-Hum. Interact., 7* (2), 197–221.

Sutherland, J. (2010). *Scrum Handbook. Everything you need to know to start a Scrum project in your organization,* Scrum Training Institute. Zugriff am 07.08.2013. Verfügbar unter http://jeffsutherland.com/scrumhandbook.pdf

Sutherland, J. & Schwaber, K. (2011). *Scrum Guide. Der gültige Leitfaden für Scrum: Die Spielregeln*. Verfügbar unter scrum.org

Tawbi, M. & Souveyet, C. (1999). Guiding Requirement Engineering with a Process Map.

Tawbi, M., Velez, F., Souveyet, C. & Achour, C. B. (2000). Evaluating the CREWS-L'Ecritoire Requirements Elicitation Process. *ICRE2000 Fourth IEEE International Conference on Requirements Engineering*.

Veer, G. C. V. D. & van Welie, M. (Groupware task analysis, Hrsg.). (1999). *Groupware task analysis*. Verfügbar unter http://www.few.vu.nl/~gerrit/gta/docs/CHI99tutorial.pdf

Vygotskiĭ, L. S. (1962). *Thought and Language. Studies in communication:* M.I.T. Press.

Walker, M., Takayama, L. & Landay, J. A. (2002). High-Fidelity or Low-Fidelity, Paper or Computer? Choosing Attributes when Testing Web Prototypes. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 46* (5), 661–665.

Walters, H. (2008). *Apple's design process*, Businessweek. Zugriff am 04.09.2013. Verfügbar unter http://www.businessweek.com/the_thread/techbeat/archives/2008/03/apples_design_p.html

Werneck, V. M. B., Oliveira, A. d. P. A. & Leite, J. C. S. d. P. (2009). Comparing GORE Frameworks: i-star and KAOS. *Workshop em Engenharia de Requisitos,* 15–26.

Williams, A. (2009). User-centered design, activity-centered design, and goal-directed design. a review of three methods for designing web applications. In *SIGDOC'09. Proceedings of the 27th ACM international conference on Design of communication* (Advancing computing as a science and profession). New York: ACM.

Wirdemann, R. (2011). *Scrum mit User Stories* (2., erw. Aufl). München [u.a.]: Hanser.

Wollesen, L. U. (2012). *Realisierung eines Rahmensystems für das Usability Engineering Repository „UsER“*. Diplomarbeit, Universität zu Lübeck. Zugriff am 29.10.2013.

Wolter, A. (2004). *CREWS – Scenario-Based Requirements Engineering*. Seminar, Universität Duisburg-Essen.

Zimmermann, D. & Grötzbach, L. (2007). A Requirement Engineering Approach to User Centered Design. In J. A. Jacko (Hrsg.), *Human-Computer Interaction. Interaction Design and Usability* (Lecture Notes in Computer Science, Bd. 4550, S. 360–369). Berlin, Heidelberg: Springer Berlin Heidelberg.

Zuser, W., Grechenig, T. & Köhle, M. (2004). *Software Engineering. Mit UML und dem Unified Process* (Informatik : Softwaretechnik, 2. überarbeitete Auflage). München: Pearson Studium.

Normen und Standards

IEEE Std 610.12:1990. IEEE standard glossary of software engineering terminology (610.12 Glossary of Software Engineering Terminology).

IEEE Std 830:1998. IEEE recommended practice for software requirements specifications (830-1998 Rec-ommended Practice for Software Requirements Specifications).

ISO 9241-11:1998. Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten. Deutsche Fas-sung EN ISO 9241-11 :1999, 13.180; 35.080; 35.18 (Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirm-geräten - Teil 11:Anforderungen an die Gebrauchstauglichkeit - Leitsätze - Teil 11).

ISO 9241-110:2008. Grundsätze der Dialoggestaltung. Deutsche Fassung EN ISO 9241-110:2006, 13.180; 35.080; 35.240.20. (Ergonomie der Mensch-System-Interaktion - Teil 110).

ISO 9241-210:2011. Prozess zur Gestaltung gebrauchstauglicher interaktiver Systeme. Deutsche Fassung EN ISO 9241-210:2010, 13.180; 35.180. (Ergonomie der Mensch-System-Interaktion - Teil 210).

ISO 9241-230:2009. Human-centred design and evaluation methods (Ergonomics of human system interaction - Part 230).

ISO/IEC 25010:2011. Systems and software engineering -- Systems and software Quality Requirements and Eval-uation (SQuaRE) -- System and software quality models, 1–34.

ISO/TR 16982:2002. Ergonomics of human-system interaction. Usability methods supporting human-centred de-sign, 16982.

ISO/TS 18152:2010. Ergonomics of human-system interaction - Specification for the process assessment of hu-man-system issues.

Publikationen

2014

Roenspieß A., **Paul M**., Mentler T., Herczeg M. (2014).
Levels of Abstraction for User Modeling in the Usabiliy Engineering Repository UsER. In Ahram, T, Karwowski, W & Marek, T (Eds.) *Proceedings of the 5th International Conference on Applied Human Factors and Ergonomics AHFE*. Krakau, Poland: AHFE. 390-400.

Paul M., Roenspieß A., Mentler T., Herczeg M. (2014).
The Usability Engineering Repository (UsER). In Hasselbring, W & Ehmke, N C (Eds.) *Software Engineering 2014 - Fachtagung des GI-Fachbereichs Softwaretechnik, 25.-28. Februar 2014*, Kiel. Gesellschaft für Informatik e.V. (GI). 2013

2013

Herczeg M., **Kammler M.**, Mentler T., Roenspieß A. (2013).
The Usability Engineering Repository UsER for the Development of Task- and Event-based Human-Machine-Interfaces. In (Ed.) *12th IFAC,IFIP,IFORS,IEA Symposium on Analysis, Design, and Evaluation of Human-Machine Systems*. Las Vegas: International Federation of Automatic Control. 483-490.

Paul M., Roenspieß A., Herczeg M. (2013).
UsER - Ein prozessorientiertes Entwicklungssystem für Usability-Engineering. In Boll, S, Maaß, S & Malaka, R (Eds.) *Mensch & Computer 2013*. München: Oldenbourg. 181-190.

2012

Herczeg M., **Kammler M.**, Roenspieß A. (2012).
Prozessorientierte Entwicklung von aufgaben- und ereignisorientierten Benutzungsschnittstellen für die Prozessführung mit Hilfe eines Usability-Engineering-Repositories (UsER). In Grandt, M & Schmerwitz, S (Eds.) *Fortschrittliche Anzeigesysteme für die Fahrzeug- und Prozessführung : 54. Fachausschusssitzung Anthropotechnik der Deutschen Gesellschaft für Luft- und Raumfahrt*. Koblenz: DGLR. 1-15.

Kammler M., Roenspieß A., Herczeg M. (2012).
UsER: Ein modulares Usability-Engineering-Repository. In Reiterer, H & Deussen, O (Eds.) *Mensch und Computer 2012: interaktiv informiert und allgegenwärtig allumfassend!?* Oldenbourg Verlag. 333-336.

2009

Winkler T., Günther S., **Kammler M.**, Feldner B., Schmitt F., Herczeg M. (2009).
Be-greifbare digitale Lernobjekte. In Kain, S, Struve, D & Wandke, H (Eds.) *Workshop-Proceedings der Tagung Mensch & Computer 2009*. Logos Verl. 353-35

Danksagung

Danken möchte ich an dieser Stelle Allen, die mich auf dem langen Weg der Entstehung begleitet und unterstützt haben, angefangen bei meinem Chef, Mentor und auch Vorbild Prof. Dr. Michael Herczeg, der dieses Projekt initiiert und ermöglicht hat. Vielen Dank für das Vertrauen und die fachliche Unterstützung in dieser gesamten Zeit und auch schon während des Studiums.

Mein weiterer Dank gilt den Kollegen des IMIS, vor allem den beiden guten Seelen des Institutes Hannelore Lamp und Anja Minzlaff, die mich moralisch und mental immer wieder von neuem motivierten. Ich danke Prof. Dr. Nicole Jochems, die zum Ende der Arbeit durch ihre Beherztheit dafür sorgte, dass ich auch ein Ende finde. Meinen lieben Kollegen und Freunden Amelie Roenspieß, Felix Schmitt, Tilo Mentler, Florian Scharf, Dr. Thomas Winkler und Christian Wolters danke ich für die schöne gemeinsame Zeit. Ganz besonders erwähnen möchte ich noch einmal Amelie und Tilo, durch deren Wortgewandtheit und fachliche Kompetenz so manche Publikation erst als solche anerkannt wurde.

Für den praktischen Bezug und die konstruktiven Besprechungen zum Thema Software Entwicklung danke ich Jan-Hendrik Krause, Lena Müller-Ontjes, Dr. Jan Müller-Ontjes und Dr. Eike Schmidt. Danke für das Vertrauen und die Zeit, die sie mir zur Verfügung gestellt haben und ohne die dieser Arbeit viele wichtige Erkenntnisse fehlen würden.

Weiterhin danke ich allen Studenten, die durch ihr Engagement maßgeblich die Entwicklung des UsER-Systems vorangetrieben haben und stellenweise auch noch dabei sind. Beginnen möchte ich hier bei Lars Ulrik Wollesen, erneut Amelie Roenspieß, zudem Kathrin Beholz, Martin Dierck und Bengi Haid, die dieses Projekt bereits in seinen ersten Iterationen maßgeblich mit beeinflussten. Aufbauend auf diesen Vorarbeiten folgten weitere Systemverbesserungen und Erweiterungen von Kim Ragna Mertens, Anna Hüttig, Lars Heyer, Tim Dubbels, Stefan Dreyer und Henrik Berndt. Danke Euch.

Ein ganz großer Dank gilt meiner Familie. Meinen Eltern Sigrid und Uwe danke ich besonders, auf deren Liebe und Unterstützung ich immer zählen konnte. Ich danke meinem Bruder Sven und meiner Schwägerin Amrita, sowie meinen lieben Schwiegereltern Amiya und Dr. Lipi Paul, die immer an mich geglaubt haben und Jürgen Richter, der mich auf der Zielgeraden mit einer Optimierung vor allem der Interpunktion dieser Arbeit unterstützte.

Zu guter Letzt danke ich meiner Frau, Dr. Pia Paul. Danke Dir für Alles.

Ehrenwörtliche Erklärung

Hiermit erkläre ich ehrenwörtlich, dass ich die vorliegende Dissertation selbständig verfasst und keine anderen als die angegebenen Hilfsmittel genutzt habe. Alle wörtlich oder inhaltlich übernommenen Stellen habe ich als solche gekennzeichnet.

Ich versichere, dass ich die beigefügte Dissertation nur in diesem und keinem anderen Promotionsverfahren eingereicht habe und, dass diesem Promotionsverfahren keine endgültig gescheiterten Promotionsverfahren vorausgegangen sind.

_____ _____

Ort, Datum Unterschrift