Universität Konstanz
FB Informatik und Informationswissenschaft
Studiengang Information Engineering

Masterarbeit

Methods and tool-support for interdisciplinary requirements modeling and user interface specification

zur Erlangung des akademischen Grades eines
Master of Science (M.Sc.)

von

Johannes Rinn

Erstgutachter: Prof. Dr. Harald Reiterer
Zweitgutachter: Prof. Dr. Marcel Waldvogel

Oktober 2008

Abstract

The development of graphical user interfaces is a key aspect in the creation of interactive systems, since their distribution has risen dramatically and natural interaction is increasingly becoming a crucial requirement. Therefore, not just the demands on the system to be developed augmented, but also the relevance of the underlying development process. A particularly important phase within this process is the requirements analysis. This phase is characterized as a cooperative process accompanied by communication and learning between people with different knowledge, experience and methods. Consequently, it is a complex task to specify a user interface unambiguously in an interdisciplinary context. Since formal, text-based and discipline-specific specification documents harm mutual understanding among stakeholders, they tend to increase overall costs and time consumption. This thesis introduces a novel, lightweight approach to the specification of user interfaces. It respects the interdisciplinary nature of the requirement analysis by employing a simple structured process model and a reduced set of models in order to enhance mutual understanding. The theoretical foundations of this approach are based on an analytical consideration of formal, semi-formal and informal expression of participating disciplines. The core concepts of the developed specification method are based on a study of modeling approaches for user interface development, existing design and specification options and well-established interdisciplinary approaches. The result is a structured model, which relates an interdisciplinary consensus of models within multiple abstraction layers. The relationships between the identified design artifacts are highlighted to make the overall process traceable and understandable. Eventually, the theoretical approach is put into practice in conjunction with an innovative tool, which finally resembles an interactive visual specification as an alternative method for text-based documents. The thesis concludes with a summary on elaborated concepts, critical reviews and recommendations for further investigations.

Überblick

Die Entwicklung von grafischen Benutzerschnittstellen ist ein zentraler Aspekt bei der Erstellung von interaktiven Systemen geworden, da deren Verbreitung stark zugenommen hat und der natürliche und unkomplizierte Umgang mit ihnen immer relevanter wird. Dadurch sind nicht nur die Ansprüche an das zu entwickelnde System gestiegen, sondern auch die Anforderungen an den zugrunde liegenden Entwicklungsprozess. Eine besonders wichtige Phase bei der Entwicklung von Benutzerschnittstellen ist die Anforderungsanalyse, die als kooperativer Kommunikations- und Lernprozess zwischen Personen mit unterschiedlichem Wissen, Erfahrungen und Methoden charakterisiert werden kann. Eine Benutzerschnittstelle im interdisziplinären Kontext nachvollziehbar und widerspruchsfrei zu spezifizieren ist daher eine komplexe Aufgabe. Da formale, textlastige und disziplinspezifische Spezifikationsdokumente das gegenseitige Verständnis unter beteiligten Akteuren erschweren, kann es zu kosten- und zeitintensiven Nachbesserungen kommen. Diese Arbeit stellt einen neuen, leichtgewichtigen Ansatz zur Spezifikation von Benutzerschnittstellen vor, der den interdisziplinären Charakter der Anforderungsanalyse mithilfe eines strukturierten Vorgehensmodells und einer reduzierten Modellpalette berücksichtigt. Zunächst werden die Grundlagen für diesen Ansatz durch eine analytische Betrachtung von formalen, semiformalen sowie informalen Ausdrucksweisen aller beteiligten Disziplinen dargelegt. Daraufhin werden die Kernkonzepte des entwickelten Spezifikationsansatzes basierend auf einer Untersuchung von modellbasierten Verfahren zur Benutzerschnittstellenentwicklung, vorhandenen Entwurfs- und Spezifikationsmöglichkeiten sowie interdisziplinären Vorgehensweisen erarbeitet. Als Ergebnis wird ein strukturiertes Vorgehensmodell vorgestellt, das die zuvor identifizierten Konsensmodelle der verschiedenen Disziplinen verknüpft. Dabei werden Zusammenhänge herausgearbeitet, welche die Vorgehensweise transparent und nachvollziehbar machen. Schließlich wird beschrieben, wie der entwickelte Ansatz im Zusammenspiel mit einem innovativen Werkzeug als interaktive visuelle Spezifikation zur Alternative für textlastige Methoden werden kann. Abschließend werden die erarbeiteten Konzepte zusammengefasst, kritisch betrachtet und Empfehlungen für eine Weiterentwicklung ausgesprochen.

Inhaltsverzeichnis

1 Einleitung **1**
1.1 Motivation und Ansatz 4
1.2 Forschungskontext und Methodik 5
1.3 Ziel der Arbeit 5
1.4 Aufbau der Arbeit 6

2 Interessengruppen und Vorgehensweisen **9**
2.1 Softwareentwicklung 9
2.1.1 Wasserfallmodell 10
2.1.2 Spiralmodell 11
2.1.3 Rational Unified Process 12
2.1.4 Agile Methoden 12
2.2 Mensch-Computer-Interaktion 13
2.2.1 User-Centered Design 17
2.2.2 Participatory Design 18
2.2.3 Contextual Design 18
2.3 Geschäftsprozessmodellierung 20
2.4 Disziplinenübergreifende Zusammenarbeit bei der Anforderungsermittlung 22
2.5 Zusammenfassung 25

3 Ansätze zur Modellierung von Benutzerschnittstellen **27**
3.1 Brückenschläge zwischen den Disziplinen 27
3.1.1 UML und UI Design 28
3.1.2 UML for Interactive Systems 29
3.1.3 UMLi - UML for Interactive Applications 30
3.1.4 Wisdom 32
3.1.5 Formal Models for UI Design 34
3.1.6 Usage-Centered Software Engineering 35
3.1.7 Bewertung der Ansätze 37
3.2 Werkzeugunterstützung 39
3.3 Zusammenfassung 41

4 Entwurfs- und Spezifikationsmöglichkeiten **43**
4.1 Spezifikation von Anforderungen 43
4.1.1 Textbasierte Spezifikationen 45
4.1.2 Modellbasierte Spezifikation 47
4.1.3 Formale Ausdrucksweisen 49
4.1.4 Informale Ausdrucksweisen 51
4.2 Techniken zur Realisierung einer interdisziplinären Spezifikation . . . 53
4.2.1 Agile Modeling 54
4.2.2 Modellierung mit Prototypen 56

4.2.3 Design Rationale 61
4.3 Zusammenfassung 63

5 Vorgehensweise und geeignete Modelle für die interdisziplinäre Spezifikation 65
5.1 Vorgehensweise während der Spezifizierung 65
5.2 Konsensmodelle und Modellierungsphasen 66
5.2.1 Problem Domain Modeling 68
5.2.2 User Modeling 70
5.2.3 Task Modeling 72
5.2.4 Interaction Modeling 76
5.2.5 User Interface Modeling 79
5.3 Zusammenfassung 81

6 Interaktive visuelle Spezifikation 85
6.1 INSPECTOR - Interdisipinary Specification Tool 85
6.2 Interaktive Modellierung 88

7 Zusammenfassung 95

A Anhang 99

Bibliography 105

Abbildungsverzeichnis

2.1 Wasserfallmodell 10
2.2 Spirallmodell 11
2.3 Rational Unified Process 12
2.4 HCI als multidisziplinäre Wissenschaft 15
2.5 Entwicklungsprozess für interaktive Systeme 17
2.6 Das Vorgehensmodell des ContextualProcess 19
2.7 Business Process und Benutzeroberfläche 21
2.8 Die involvierten Disziplinen im Entwicklungsprozess 24

3.1 UML for Interactive Systems 29
3.2 UML for Interactive Applications 31
3.3 Modellierungsprozess für UMLi 31
3.4 Arbeitsabläufe, Aktivitäten und Modelle in Wisdom 32
3.5 Der Analyserahmen für Interaktive Systeme in Wisdom 33
3.6 Informale und Formale Darstellung eines UI 35
3.7 Usage-Centered Design Prozess 36
3.8 Canonical Abstract und detaillierter Protoyp 37

4.1 Zusammenhang von Modellierung und Interpretation 48
4.2 Zyklus von Aktivitäten bei kreativen Vorgängen 51
4.3 Unterschiedliche Darstellung von Use Case Diagrammen 56
4.4 Modellierungsformen zur Exploration und Kommunikation von UI-Entwürfen 57
4.5 Verwendung von Prototypen 60

5.1 Spezifikationsprozess 66
5.2 Phasen der Modellierung und entsprechende Aufgaben 68
5.3 useagescen 69
5.4 Zusammenhang zwischen Role Maps und Persona 72
5.5 Zusammenhänge zwischen Task Map, Essential Use Case und Use Case Diagramm 75
5.6 Zusammenhänge zwischen Activity Diagramm, Sequence Diagramm, Data Flow Chart und Use Cases 77
5.7 Zusammenhang von Storyboard und Prototypen 80
5.8 Konsensmodelle der interdisziplinären Spezifikation 82
5.9 Schematischer Modellierungsablauf 83

6.1 Interaktionsmöglichkeit für skalierbare grafische Oberflächen 86
6.2 Die Skalierungsebenen in INSPECTOR 87
6.3 Die Scenario Map als Startpunkt der Modellierung 89
6.4 Die vier Skalierungsebenen dargestellt im Werkzeug 90
6.5 Zusammenhänge zwischen Modellen in INSPECTOR 91

6.6 Prototypen mit unterschiedlicher Fidelity 92
6.7 Unterschiedliche Verknüpfungsmöglichkeiten von Modellen 93

A.1 Scenario Map 100
A.2 Die Skalierungsebenen 101
A.3 Traceability 102
A.4 UI Design Layer 103
A.5 Herstellung von Relationen 104

Tabellenverzeichnis

2.1 Merkmale von Usability 16
2.2 Normierte Prinzipien der Usability 16

3.1 Bewertung der vorgestellten Vorgehensweisen 38
3.2 Bewertung von aktuellen UI Entwicklungswerkzeugen 40

4.1 Informale Artefakte für Spezifikationen 53
4.2 Prinzipien des Agile Modeling 55
4.3 Kategorisierung von Prototypen 59
4.4 Vergleich zwischen Prototypen mit unterschiedlicher Genauigkeit . . . 61

Kapitel 1

Einleitung

Benutzerschnittstellen definieren interaktive Systeme

Seit den ersten Bemühungen, die Kommunikation zwischen System und Anwender mithilfe einer grafischen Oberfläche zu verbesseren (Sutherland, 1964) ist die Bedeutung einer Benutzerschnittstelle (engl. *User Interface*) immer größer geworden. Ein *User Interface* (UI) – heutzutage fast ausschließlich in grafischer Form realisiert, wie (Strahonja und Picek, 2005) darstellen – ist mehr als nur Ein- und Ausgabefläche. Ein UI hat als Schnittstelle zwischen Benutzer und System entscheidenden Einfluss darauf, wie gut Kommunikation und Interaktion mit einem interaktiven System ablaufen (Jacob, 1986). Statt Kommandozeilen werden Oberflächen benutzt, die viele unterschiedliche Interaktionskonzepte ermöglichen. Als essenzieller Bestandteil von interaktiven Systemen beeinflusst das UI deutlich die Effektivität von installierten Applikationen (da Silva und Paton, 2000a). Somit entscheidet in erster Linie das UI darüber, ob eine Software akzeptiert oder abgelehnt wird, da Benutzer zwischen UI und System nicht unterscheiden. Nur durch ein intuitiv und leicht zu bedienendes UI werden Benutzer nicht bei der Ausübung ihrer Arbeit behindert, sondern unterstützt und gefördert, was deren Zufriedenheit und Produktivität erhöht. Dies steigert die Vertrauenswürdigkeit gegenüber Kunden und bindet sie stärker an das eigene Unternehmen (Mayhew, 2005, p. 17ff), (Bias und Mayhew, 2005, p. 26 ff).

> „The importance of the user interface to any system cannot be overstated. It is the user interface that allows system operators to see the current status of the system as well as control the system. The user interface is the first thing seen by a system operator and is therefore the most tangible aspect of the system."
>
> (Salvucci et al., 2005)

Ansprüche an interaktive Systeme steigen

Die Entwicklung von UIs ist zu einem zentralen Punkt bei der Erstellung von Software geworden, da die Verbreitung von interaktiven Systemen immer weiter zugenommen hat und der Umgang mit ihnen immer relevanter wird (Hardtke, 2004). Interaktive Systeme sind nicht mehr nur Expertenwerkzeuge, heutzutage muss nahezu jeder täglich mit ihnen interagieren. Dadurch sind nicht nur die Ansprüche an ein UI gestiegen (Rosson und Carroll, 2002), sondern auch die Anforderungen an die Erstellung von UIs. Je einfacher ein UI zu benutzen und zu verstehen ist, desto schwieriger ist dessen Entwicklung (Lozano et al., 2001). So haben Studien gezeigt,

dass 48% des Quellcodes für die Benutzeroberfläche benötigt und für dessen Implementierung nahezu 50% der gesamten Entwicklungszeit benötigt werden (Myers und Rosson, 1992).

> „UI design is recognized as a process that is [...] open (new considerations may appear at any time), iterative (several cycles are needed to reach an acceptable result), and incomplete (not all required considerations are available at design time)."
>
> (Coyette und Vanderdonckt, 2005)

Anforderungsermittlung ist komplex und kritisch

Eine besonders kritische Phase bei der Erstellung von interaktiven Systemen ist die Anforderungsermittlung (Sidoran et al., 1995). Insbesondere bei komplexen Anwendungsfeldern kann die Anforderungsanalyse nicht in einem Schritt erfolgen. Einzelne Schritte müssen wiederholt werden, da es kaum möglich ist alle Informationen, Abhängigkeiten, Rollen und Aufgaben in einem einzigen Schritt zu erfassen. Gerade beim UI, dessen Benutzbarkeit in hohem Maße von den Bedürfnissen der Benutzer, ihrem Arbeitsumfeld und ihren Aufgaben abhängt, ist eine genaue und zeitintensive Untersuchung nötig.

Einbinden von unterschiedlichen Gruppen ist wichtig

Darüber hinaus ist es von großer Bedeutung, dass Benutzer in diesen Prozess mit einbezogen werden, auch wenn es sich dabei um noch keine gängige Praxis handelt (Rashid et al., 2006). Sie sind es, die später mit dem zu entwickelnden System arbeiten und sich mit den Arbeitsabläufen, Aufgaben und Zielen auskennen (Damm et al., 2000). Neben Anforderungen von Benutzern fließen auch technische Rahmenbedingungen, betriebswirtschaftliche Prämissen und allgemeine Gestaltungsregeln ein, weshalb andere Interessengruppen wie etwa Designer, Experten der Mensch-Computer-Interaktion, Softwareentwickler sowie Manager und Mitarbeiter mit betriebswirtschaftlichem Hintergrund ebenfalls involviert sind.

Interdisziplinärer Vorgang

Die Anforderungsermittlung ist ein kooperativer Kommunikations- und Lernprozess zwischen Personen mit unterschiedlichem Wissen, Erfahrungen und Erwartungen. Das Ziel ist es, ein gemeinsames Verständnis über den Anwendungsbereich und die Anforderungen zu erlangen. Hofmann und Lehner (2001) bezeichnen dies als den wichtigsten und schwierigsten Teil eines Softwareprojekts, da es die Hauptursache für fehlgeschlagene Entwicklungen gerade im Bereich der UIs ist.

> „The most sensitive part of the analysis and planning of software systems to be developed is the development of user interfaces. This is mostly because the planning phase of a user interface is a highly communication intensive field."
>
> (Tick, 2005)

Mißverständnisse entstehen

Was Benutzer im Endeffekt tatsächlich für ihre Arbeit benötigen, unterscheidet sich oftmals gravierend von dem, was sie denken zu brauchen. Zudem haben Benutzer häufig Schwierigkeiten die Anforderungen im Detail zu formulieren, da Sprache mehrdeutig und oftmals durch Begriffe aus dem Fachjargon angereichert ist. Missverständnisse sind dabei ein häufiges Problem und eine weiterere Ursache für nicht zufriedenstellend abgeschlossene Softwareprojekte (Tick, 2005). Missverständlich

spezifizierte Anforderungen sind der Grund, weshalb dieselben Sachverhalte von beteiligten Akteuren unterschiedlich interpretiert und verstanden werden (Rashid et al., 2006). Werden Anforderungen falsch aufgefasst, dann können Schneeballeffekte entstehen, sodass alle Entscheidungen, die darauf beruhen ungültig sind, weshalb fehlerhafte Systeme umgesetzt werden (Ambler und Jeffries, 2002).

Spezifikation kommuniziert Anforderungen an das zu implementierende System

Spezifikationen werden deshalb dazu benutzt, alle Sachverhalte der zu entwickelnden Software vollständig, genau, nachvollziehbar und widerspruchsfrei darzustellen (Granic und Glavinic, 2002). Vor allem wenn kostenintensive oder selbst nicht effizient ausführbare Aufgaben in der Softwareentwicklung an spezialisierte Dienstleister abgegeben werden, um hohen Kosten, verzögerter Fertigung und geringer Rendite des eingesetzten Kapitals entgegenzuwirken, sind Spezifikationen wichtig. Spezifikationen dienen als Bindeglied zwischen Anforderungsermittlung und Implementierung und beschreiben die notwendigen Bedingungen für das umzusetzende System. Eine Spezifikation beschreibt, wie der Benutzer mit der Bedienoberfläche interagiert, aber auch, wie das UI mit dem darunterliegenden System Daten und Informationen austauscht (Jacob, 1986).

Modellierungstechniken sind unterschiedlich

Traditionell bedingt werden Softwaresysteme und das dazugehörende UI durch Experten mit unterschiedlichem Hintergrund und mit unterschiedlichen Techniken und Methoden entworfen. Beim Software Engineering werden sehr formale Methoden und Modelle bevorzugt, wohingegen Designer und Experten der Mensch-Computer Interaktion mehr zu informalen, besser verständlichen Ausdrucksweisen tendieren, die kreative Prozesse besser unterstützen. UI-Spezifikationen sind daher eher informal und enthalten hauptsächlich textuelle Beschreibungen, aber auch Zeichnungen, Grafiken, Diagramme und prototypische Entwürfe. Eine Spezifikation für ein UI kann deshalb aus mehreren Artefakten bestehen (Geiger und Müller, 1998), (McInerney und Sobiesiak, 2000), welche von den Entwicklern dann in formalere und verbindlichere Modelle umgewandelt werden. Bei Artefakten handelt es sich um Repräsentationen von Informationen, welche während des Entwicklungsprozesses entstanden sind oder die dabei verwendet wurden.

Herkömmliche Spezifikationen sind mehrdeutig interpretierbar

Herkömmliche UI-Spezifikationen werden oft durch *Styleguides* ergänzt, um genaue Designrichtlinien für die Umsetzung vorgeben zu können. Dadurch hat das Spezifikationsdokument oft einen beachtlichen Umfang (Mayhew, 1999), reicht aber dennoch nicht aus, um zufriedenstellende Ergebnisse bei der Umsetzung der Anforderungen zu erzielen, da viel Freiraum für Interpretationen entsteht. Ein weiteres Problem sind die vielen Medienbrüche, da unterschiedliche Software zum Erstellen der einzelnen Artefakte verwendet wird. Dies ist die Ursache, weshalb der Auftragnehmer oft nicht in der Lage ist, selbstständig ein benutzerfreundliches System nach den exakten Vorstellungen des Auftraggebers zu gestalten (Offergeld und Oed, 2006).

> „The gap between the requirements expressed by business managers and the functions implemented by IT remains one of the biggest sore points in enterprise application development, as well as one of the biggest causes of waste."
>
> (Zetie et al., 2005)

Ergebnisse sind unbefriedigend

Die gelieferte Software entspricht somit nicht den Bedürfnissen der Benutzer, wodurch wieder und wieder nachgebessert werden muss. Laut Ambler und Jeffries (2002) wird oftmals keine Software geliefert, wie der Auftraggeber sie verlangt, sondern sie wird so erstellt und ausgeliefert, wie es der Auftragnehmer möchte. Verhandlungen mit den Auftragnehmern müssen geführt werden, die Kosten steigen und der Zeitpunkt, zu dem das Produkt auf den Markt gebracht werden kann (*time-to-market*), verzögert sich. Der Auftraggeber kann sich deshalb nicht darauf beschränken, die System- und Benutzeranforderungen zu definieren (Memmel et al., 2007d), vielmehr müssen die unzureichenden textuellen Spezifikationen durch visuelle Spezifikationen abgelöst werden (Rashid et al., 2006), die im besten Fall interaktiv sind.

System widerspricht den Vorstellungen

1.1 Motivation und Ansatz

Kooperation von Hochschule und Industrie

Im Rahmen von BEST (*Business Excellence in Software Usability and Design*), einer Forschungskooperation der Universität Konstanz mit der Daimler AG Forschung in Ulm und basierend auf zusätzlichen Erfahrungen, bei der Zusammenarbeit mit der Dr. Ing. h.c. F. Porsche AG (Memmel und Reiterer, 2007), (Memmel und Heilig, 2007), (Memmel et al., 2007d), (Memmel et al., 2007e), ist ein Ansatz entstanden, der die bisherigen Probleme vermindern soll. Es handelt sich dabei um eine leichtgewichtige Vorgehensweise zur Modellierung, Gestaltung und Spezifikation von Benutzerschnittstellen. Dabei unterstützen ein Vorgehensmodell und eine reduzierte Modellpalette die Definition von technischen und nicht-funktionalen Anforderungen. Im Verlauf der Arbeit werden formale, semiformale sowie informale Ausdrucksweisen vorgestellt, die besonders geeignet für das interdisziplinäre Spezifizieren von Anforderungen für benutzer- und aufgabengerechten UIs sind. Prototypen, die sowohl statisch als auch interaktiv eingesetzt werden können, unterstützen dabei formellere Diagramme und tragen zur Übermittlung des *Look & Feels* bei.

Interaktive Spezifikation

Durch die Verwendung von ausführbaren Prototypen wird die Spezifikation erlebbar und das Verhalten des UI kann besser nachvollzogen werden. Verhalten kann hier sowohl zeitbezogene Veränderungen als auch interaktive Abläufe bedeuten. Somit können komplexe Sachverhalte genau dargestellt und erörtert werden und es gibt Ausdrucksformen, sodass von allen Beteiligten verstanden werden. Bei Missverständnissen können entweder der Prototyp oder die Modelle herangezogen werden, wodurch Kommunikationsprobleme ebenfalls vermindert werden können. Dies verbessert die Kooperation und Kollaboration von beteiligten Akteuren, weshalb sich die Entwicklungszeit verringert und die Qualität der Software zunimmt. Zudem kann die Evaluation des UI schon vor der Umsetzung stattfinden, und nicht mehr wie bisher kurz vor der Fertigstellung des Systems.

> „Textual requirement specifications are not enough, as users do not have the time to get deeply involved in the system and therefore need to be assisted during the specification of their requirements. Visual requirements specification seems to be well-suited for this challenge."
>
> Rashid et al. (2006)

1.2 Forschungskontext und Methodik

Konzeption und Entwicklung

Diese Arbeit bezieht sich auf aktuelle Forschungsergebnisse, die an der Universität Konstanz am Lehrstuhl Mensch-Computer Interaktion im Rahmen des Projektes *INSPECTOR* seit Oktober 2006 gemacht wurden. Basierend auf Erfahrungen, die mit modellgetriebenen Prozessen (Memmel und Heilig, 2007), (Memmel et al., 2007a) und agiler Softwareentwicklung (Gundelsweiler et al., 2004), (Memmel et al., 2007b), (Memmel et al., 2007d) gemacht wurden, ist *INSPECTOR* als Werkzeug mit agiler Methodik für die interdisziplinäre Spezifikation von Benutzerschnittstellen entstanden. Die Konzeption und Entwicklung von Werkzeug und Methode basierte dabei auf den folgenden Tätigkeiten:

- Anforderungen an Werkzeuge für die Spezifikation von Benutzerschnittstellen wurden erhoben (siehe König (2008) und Kapitel 3.2—"Werkzeugunterstützung")
- Methoden und Modelle wurden ausfindig gemacht, welche für das gemeinschaftliche Spezifizieren von Benutzerschnittstellen geeignet sind (siehe Kapitel 4—"Entwurfs- und Spezifikationsmöglichkeiten" und Kapitel 5—"Vorgehensweise und geeignete Modelle für die interdisziplinäre Spezifikation")
- Ein Werkzeug zur Unterstützung der interdisziplinären Spezifikation von Benutzerschnittstellen wurde entwickelt (siehe Geyer (2008) und Kapitel 6—"Interaktive visuelle Spezifikation")

Weitere Informationen zu dieser Arbeit sowie Videos, Screenshots und veröffentlichte Publikation sind auf der Projektseite[1] zugänglich.

1.3 Ziel der Arbeit

Das Ziel der vorliegenden Arbeit ist es, Möglichkeiten für die interdisziplinäre Spezifikation von Benutzerschnittstellen zu schaffen. Dabei werden UIs für sicherheitsrelevante Systeme nicht betrachtet, da diese eine starke Formalisierung der Methoden erfordern. Deshalb wurde der Fokus auf folgende Merkmale gelegt:

- Interdisziplinärer Ansatz - Bei der Entwicklung von interaktiven Systemen arbeiten Beteiligte aus verschiedenen Fachbereichen zusammen. Die funktionsbasierte Systemsicht muss mit der benutzerorientierten Sichtweise verbunden werden, ohne dass betriebswirtschaftliche Hintergründe vergessen werden. Es ist erforderlich, dass ein Spezifikations- und Modellierungsansatz geschaffen wird, der für die gemeinschaftliche Spezifikation geeignet ist und dadurch die Möglichkeit zur effektiven Zusammenarbeit schafft.
- Interaktive Spezifikation - Bisherige Spezifikationen sind schwer lesbar und führen zu Verständigungsschwierigkeiten, Mehrdeutigkeiten und Medienbrüchen. Ausdrucksweisen und Notationen werden verwendet, welche nicht

[1] http://hci.uni-konstanz.de/inspector

für alle Teilnehmer verständlich sind. Durch die Verbindung von Modellen mit interaktiven visuellen Artefakten soll das *Look & Feel* des UIs erlebbar werden. Dadurch können Sachverhalte in verständlicher Weise dargestellt werden und zur Unterstützung der verwendeten Modelle dienen.

- Agilität - Eine leichtgewichtige Herangehensweise, ohne die Nachteile herkömmlicher Spezifikationen, steht im Vordergrund dieser Arbeit, um die Iterationshäufigkeit bei der Entwicklung von Benutzerschnittstellen zu unterstützen. Die Folgen dieser flexiblen Herangehensweise sind eine reduzierte Entwicklungsdauer und eine erhöhte Kommunikation unter den Beteiligten. Zudem können Ergebnisse schneller erzielt und evaluiert werden.

- Traceability - Die Nachverfolgbarkeit von Anforderungen ist ein wichtiger Bestandteil einer Spezifikation. Zum einen können Abhängigkeiten unter den Anforderungen selbst erkannt werden, zum anderen müssen Anforderungen auch nach der Umsetzung der Spezifikation identifiziert werden können. Entscheidungen, die getroffen werden, bleiben nachvollziehbar und können legitimiert werden.

- Kreativität - Herkömmliche Herangehensweisen an die Erstellung von funktionalen, benutzertauglichen Benutzerschnittstellen sind unzureichend, um innovative Wege und neue Möglichkeiten der Interaktion zwischen Benutzer und System herzustellen. Um alternative Lösungen zu finden, die nicht auf konventionellen Ansätzen beruhen ist es wichtig, kreatives Arbeiten zu unterstützen.

1.4 Aufbau der Arbeit

Kapitelübersicht

In den folgenden Kapiteln werden die Vorgehensweise und geeignete Modelle zur kooperativen Erstellung von Spezifikationen vorgestellt, bei welchen Verständigungsprobleme durch erhöhte Kooperation und vereinfachte allgemeingültige Notationen vermindert werden. Das Resultat ist ein agiler leichtgewichtiger Prozess, welcher im Zusammenspiel mit einem innovativen Werkzeug besonders für interdisziplinär zusammengesetzte Gruppen geeignet ist.

Kapitel 2—"Interessengruppen und Vorgehensweisen" dient zur Vorstellung der in den Spezifikationsprozess involvierten Disziplinen und ihrer verschiedenen Aufgaben. Vorgehensweisen, die zur Erstellung von funktionaler und gebrauchstauglicher Software erforderlich sind, werden vorgestellt. Zudem wird aufgezeigt, warum das Business Engineering eine wichtige Rolle beim Entwurf von Benutzeroberflächen spielt und weshalb diese drei Disziplinen bei der Anforderungsanalyse kooperieren müssen.

Kapitel 3—"Ansätze zur Modellierung von Benutzerschnittstellen" stellt aktuelle Entwurfs- und Spezifikationsansätze zur Modellierung von Benutzerschnittstellen vor, welche als Brückenschläge zwischen den Disziplinen verwendet werden können. Diese Vorgehensweisen werden auf die Eignung zur interdisziplinären Spezifiaktion untersucht und bewertet. Hilfsmittel und Werkzeuge, die während einer Spezifikation eingesetzt werden können, sind ebenfalls aufgeführt und werden im Bezug auf ihre Eignung für die interdisziplinäre Modellierung betrachtet.

Kapitel 4—"Entwurfs- und Spezifikationsmöglichkeiten" stellt Möglichkeiten zur Spezifikation von Anforderungen vor und beschreibt die Vor- und Nachteile für interdisziplinär zusammengesetzte Teams. Es werden Techniken und Methoden präsentiert, mit denen sich die Nachteile bisheriger Spezifikationsformen ausgleichen lassen, wodurch es möglich wird, dass Anforderungsanalyse und Spezifikation durch Beteiligte aus den verschiedenen Fachgebieten gemeinsam durchgeführt werden können.

Im Kapitel 5—"Vorgehensweise und geeignete Modelle für die interdisziplinäre Spezifikation" wird die entwickelte Herangehensweise zur Modellierung und Spezifikation von UIs vorgestellt. Dabei werden Modelle identifiziert, welche von allen Disziplinen verstanden werden und welche ein gemeinsames Verständnis ermöglichen. Diese Konsensmodelle bilden den gemeinsamen Nenner, welcher es den Interessengruppen ermöglicht kooperativ zu spezifizieren.

Kapitel 6—"Interaktive visuelle Spezifikation" stellt ein interaktives Werkzeug und dessen Prinzipien vor. Es dient als Testumgebung zur Anwendung der Konsensmodelle und bildet mit diesem zusammen eine interaktive, visuelle Spezifikation. Das Werkzeug unterstützt die hierarchische Struktur der verwendeten Modelle und ermöglicht die Verbindung von abhängigen Modellen.

Diese Arbeit schließt in Kapitel 7—"Zusammenfassung" mit einer Zusammenfassung der vorgestellten Konzepte und gibt einen Ausblick auf mögliche Anknüpfungspunkte für nachfolgende Projekte.

Kapitel 2

Interessengruppen und Vorgehensweisen

Im folgenden Kapitel werden Fachrichtungen mit verschiedenen Aufgaben aufgezeigt, die bei der Entwicklung von Software bzw. deren Benutzerschnittstellen eine Rolle spielen. Es werden Vorgehensweisen vorgestellt, die zur Erstellung von funktionaler und gebrauchstauglicher Software erforderlich sind. Dabei wird auch eingegangen, warum agiler Herangehensweisen in manchen Fällen erforderlich sind. Anschließend wird aufgezeigt warum eine enge Kooperation zwischen diesen Bereichen schon bei der Anforderungsermittlung erstrebenswert ist.

2.1 Softwareentwicklung

Von Build & Fix zu strukturiertem Vorgehen

In den Anfängen der Softwareentwicklung Mitte der fünfziger Jahre wurde Software ohne strukturiertes Vorgehen umgesetzt. Hardware war teuer und Rechenzeit wurde eingeteilt, deshalb war eine intensive manuelle Fehlersuche erforderlich, bevor beim Einsatz von Software Rechenzeit verschenkt wurde. Ein System wurde vom Programmierer nach seinen eigenen Vorstellungen umgesetzt. Traten Änderungswünsche auf, wurde so lange nachgebessert, bis die Software den Ansprüchen genügte (Boehm et al., 1984). Dieses evolutionäre Build & Fix ist kein geeigneter Ansatz zur Entwicklung von komplexen interaktiven Systemen (Nunes und Cunha, 2000b). Es liegt kein strukturiertes Vorgehen zugrunde, die Dokumentation wird vernachlässigt und Nachbesserungen sind sehr kostenintensiv. Zudem kam Anfang der siebziger Jahre, durch die unverändert schlechte Qualität der erzeugten Software-Systeme, der Begriff der Software-Krise auf (Zuser et al., 2005). Als Lösungsvorschlag wurde deshalb gefordert, geeignete Methoden und Vorgehensmodelle zu entwickeln, um die Softwareentwicklung nicht mehr als Kunst, sondern als ingenieurmäßige Tätigkeit zu etablieren. Im *IEEE Standard Glossary of Software Engineering Terminology* (IEEE, 1990) wird Software Engineering wie folgt definiert:

1. Die Anwendung eines systematischen, disziplinierten und quantifizierbaren Ansatzes auf die Entwicklung, den Betrieb und die Wartung von Software, das heißt, die Anwendung von Prinzipien des Ingenieurwesens auf Software.

2. Die Untersuchung von Ansätzen, welche wie (1) definiert sind.

Deshalb wurden strukturierte Herangehensweisen entwickelt, die dabei helfen sollten, funktionsfähige Software in akzeptabler Zeit und innerhalb eines bestimmten finanziellen Rahmens zu erstellen. Nachfolgend findet sich deshalb ein Überblick über Prozesse, die im Software Engineering (SE) eine Rolle spielen.

2.1.1 Wasserfallmodell

Sequenzielles (iteratives) Wasserfallmodell

Eine erste strukturierte Herangehensweisen war das sequenzielle Wasserfallmodell (siehe Abb. 2.1), bei dem der Entwicklungsprozess in überschaubare, zeitlich und inhaltlich begrenzte Phasen eingeteilt wurde (Royce, 1987). Anforderungsanalyse und Entwurf spielten eine wichtigere Rolle und das Vorgehen wurde besser organisiert. Dieses Wasserfallmodell, das auch iterative Schritte vorsieht, bietet einen einfachen Weg, die Softwareentwicklung zu steuern und zu planen. Iterationen können durchgeführt werden, wenn in einem späteren Schritt erkannt wurde, dass die Informationen, die vom Schritt zuvor geliefert wurden, unvollständig sind oder den aktuellen Ansprüchen nicht genügen (Dix et al., 2003).

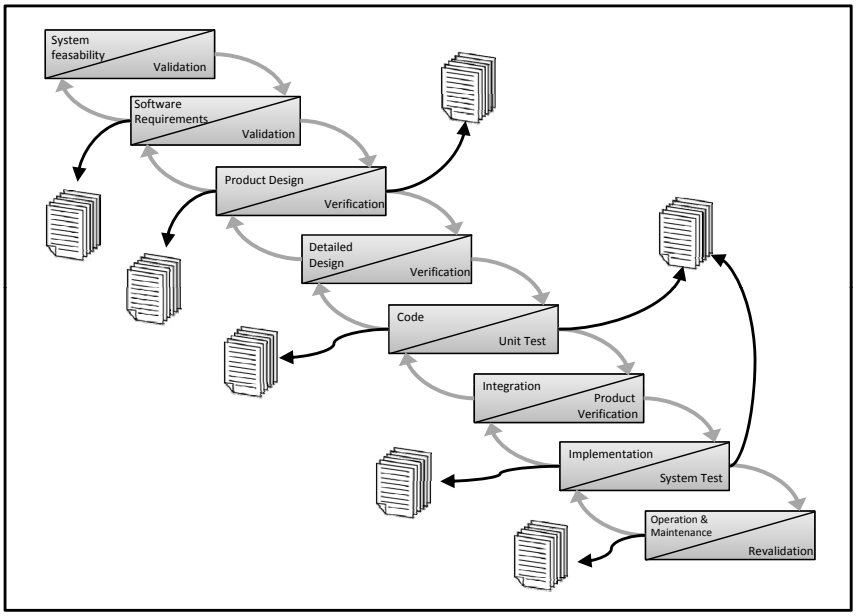


Abbildung 2.1: Softwareentwicklung nach dem Wasserfallmodell (Royce, 1987)

Allerdings beruht das Wasserfallmodell sehr stark auf Dokumentation. Nach jedem Schritt wird ein Dokument erstellt, das als Meilenstein dient und für die weitere Entwicklung in der nächsten Phase notwendig ist. Die Folge ist eine langsam vorangehende Entwicklung mit geringer Produktivität, die zudem eine schlechte Skalierbarkeit aufweist (Boehm, 2006), (Düchting et al., 2007). Um den Entwicklungsvorgang zu beschleunigen waren deshalb Herangehensweisen gefragt, die schneller Ergebnisse liefern und in den folgenden Kapiteln vorgestellt werden.

2.1.2 Spiralmodell

Objektorientierte Programmiersprachen und UML

Das Aufkommen von objektorientierten Programmiersprachen machte es möglich wiederverwendbare Software zu entwickeln. Verbesserte Programmierumgebungen, die schnellere Entwicklung von Prototypen (engl. *Rapid Prototyping*) und leistungsfähigere Hardware trugen ebenfalls zur Steigerung der Produktivität bei (Brooks, 1987). Hinzu kam die große Akzeptanz der *Unified Modeling Language* (UML) in den 1990er Jahren als standardisierte Sprache für die objektorientierte Modellierung von interaktiven Systemen (Phillips und Kemp, 2002), (Booch et al., 2005). Software wurde immer mehr zu einem wettbewerbsentscheidenden Faktor und der Bedarf an kurzen Produkteinführungszeiten machte eine Abkehr vom Wasserfallmodell notwendig. Gefragt waren nicht mehr sequenzielle Verfahren, sondern Vorgehensweisen bei denen Anforderungsanalyse, Entwurf und Entwicklung nahezu parallel ablaufen (Boehm, 2006). Zudem kamen erste Ansätze und Theorien auf, die den Benutzer stärker in die Entwicklung von Software einbezogen. Wie Strahonja und Picek (2005) darstellen, wird dieser Aspekt bei der Softwareentwicklung vernachlässigt oder sogar ignoriert (siehe Kapitel 2.2—"Mensch-Computer-Interaktion").

Inkrementelles und iteratives Spirallmodell

Das Spiralmodell, das aus dem Wasserfallmodell hervorging, wurde von Boehm (1988) entwickelt. In diesem Modell spielen Iterationen, inkrementelle Zyklen, eine regelmäßige Risikoanalyse und der Entwurf von Prototypen eine wichtige Rolle (siehe Abb. 2.2). Durch die sich wiederholenden Phasen kann auf geänderte Anforderungen reagiert werden und durch die Beurteilung der Risiken können auch große Projekte erfolgreich abgeschlossen werden (Düchting et al., 2007).

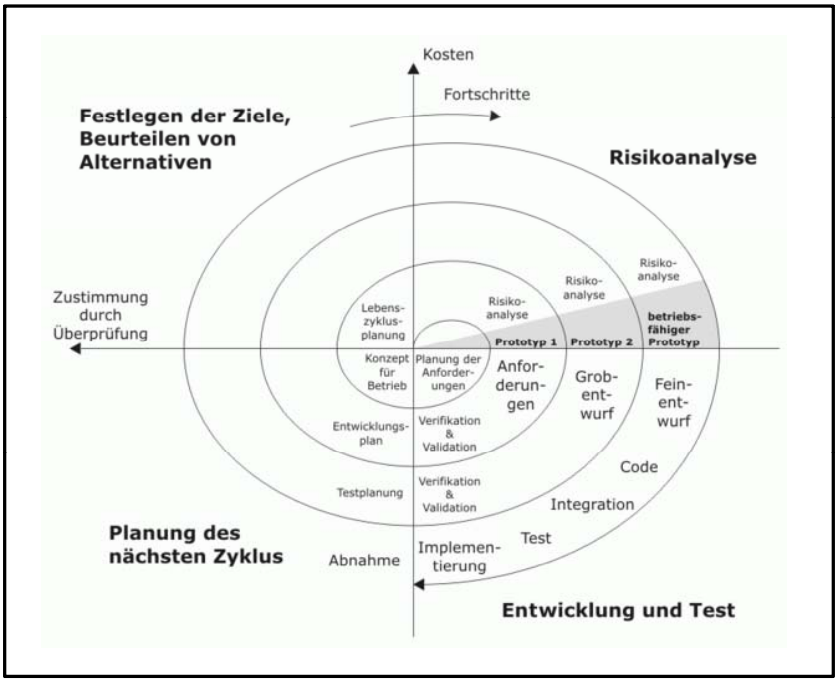


Abbildung 2.2: Phasen und Ablauf des Spiralmodells (Boehm, 1988)

2.1.3 Rational Unified Process

Rational Unified Process

Der etwas später entwickelte *Rational Unified Process* (RUP) (siehe Abb. 2.3) ist ebenfalls ein iterativer Entwicklungsprozess, dessen Modelle auf der UML-Notation beruhen. (Jacobson et al., 1999). Das Ziel des RUP ist es sicherzustellen, das Software von hoher Qualität innerhalb eines gewissen Zeitplanes und Budgets hergestellt wird. Nach jeder Iteration findet eine Überprüfung statt, in der geklärt wird, ob alle Ziele erreicht wurden. Falls nicht, werden die Gründe ermittelt, die danach bei der nächsten Iteration als Eingangswert besondere Berücksichtigung finden. Charakteristisch für diesen Prozess sind die iterativen Schritte, ausgiebige Testfälle, die Berücksichtigung von Geschäftsprozessen (engl. *Business Process*) und eine stärkere Einbindung von Benutzern (Phillips und Kemp, 2002). Prototyping spielt ebenfalls eine wichtige Rolle, da die Gestaltung der Benutzerschnittstelle und die Bedürfnisse der Endnutzer im Vordergrund stehen (Zuser et al., 2005). Zudem werden im Gegensatz zu den bisher vorgestellten Prozessen betriebswirtschaftliche Hintergründe mit einbezogen, weshalb die Geschäftsprozessmodellierung neben der Anforderungsermittlung zu Beginn sehr wichtig sind. Allerdings ist der RUP sehr schwerfällig und beruht auf einer starken Formalisierung des Prozesses, weshalb er sich nicht für jedes Anwendungsgebiet eignet. Aufgrund dessen wird im folgenden Kapitel eine Vorgehensweise beschrieben, die sich besser anpassen lässt und schneller Ergebnisse liefert.

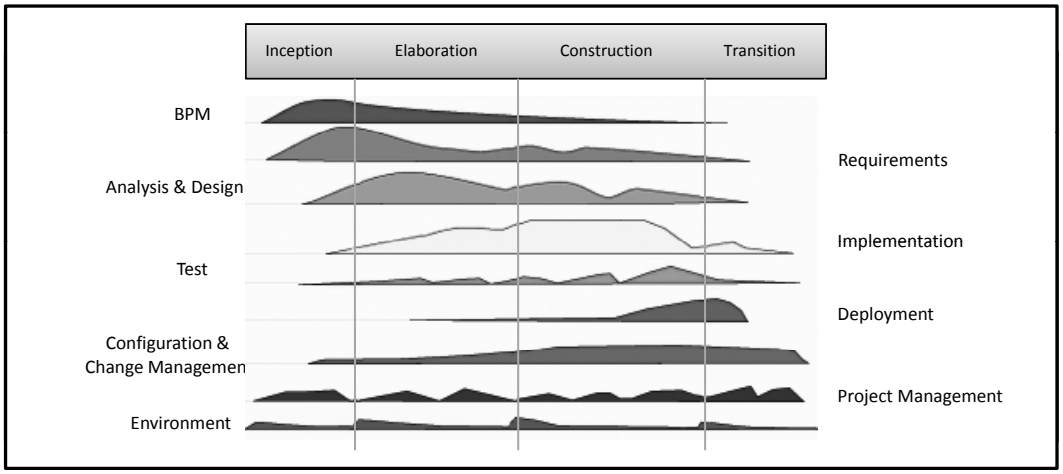


Abbildung 2.3: Der Rational Unified Process (Jacobson et al., 1999)

2.1.4 Agile Methoden

Agilere Softwareentwicklung

Der Trend zu einer schnelleren Anwendungsentwicklung hat weiter zugenommen. Informationstechnologien, Organisationen, Wettbewerbssituationen und das Umfeld der Unternehmen sind einem schnelleren Wandel unterworfen. Dies führte dazu, dass die schwergewichtigen und stark formalisierten Prozessmodelle mit ihren detaillierten Spezifikationen und Dokumenten nicht mehr den erhofften Nutzen liefern konnten (Boehm, 2006). Sie sind für Projekte mit sich ständig ändernden Anforderungen unangebracht. Die Folge waren leichtgewichtige Verfahren (engl. *Agile Methods*), wie etwa eXtreme Programming (XP) (Beck und Andres, 2004) oder Scrum (Schwaber und Beedle, 2001) Ende der neunziger Jahre. Sie richteten sich nach dem

Agile Manifesto, einer Doktrin, in welcher folgende vier Hauptziele definiert wurden:

- Individuen und Interaktionen haben eine höhere Bedeutung als Prozesse und Werkzeuge
- Funktionierende Programme haben eine höhere Bedeutung als ausführliche Dokumentationen
- Zusammenarbeit mit Kunden hat eine höhere Bedeutung als das Aushandeln von Verträgen
- Reaktionen auf Änderungen haben eine höhere Bedeutung als das Befolgen eines Plans

Iterativ und Inkrementell

Diese *Agile Methods* (AM) folgen ebenfalls einem iterativen und inkrementellen Softwareentwicklungsansatz. Die Iterationslängen sind kürzer und es wird auf ausführliche Dokumentation verzichtet. Stattdessen liegt der Fokus stärker auf Quantität und Qualität von Software. Die Zusammenarbeit innerhalb des Entwicklerteams durch bessere Kommunikation und Kollaboration macht es möglich, auf ausufernde Dokumentation zu verzichten (Düchting et al., 2007).

Flexible Vorgehensweisen

Um neue und innovative Produkte zu erzeugen, ist es notwendig, dass Vorgehensweisen Kreativität und innovatives Arbeiten unterstützen. Deshalb können Prozesse, welche bereits im Vorfeld alle Aktivitäten definieren und bei denen alle Anforderungen erhoben sein müssen, bevor die Entwicklung startet, dies nicht gewährleisten. Viele Anforderungen werden erst während der Entwicklungsphase sichtbar, weshalb viele Projekte scheitern (Larman, 2003). Techniken der AM nehmen darauf Rücksicht und können sich ändernden Anforderungen und Bedingungen während des Prozesses anpassen (Düchting et al., 2007). Allerdings eignen sich AM nicht unbedingt für jedes Projekt. Vor allem bei großen Entwicklerteams muss beachtet werden, dass die Anforderungen an Kommunikation und Kollaboration innerhalb des Projektteams im geforderten Maß umgesetzt werden können.

Da die Belange der späteren Nutzer durch SE nicht in ausreichendem Maße berrücksichtig werden, gibt es eine weitere Interessengruppe, die involviert werden muss und im nächsten Abschnitt vorgestellt wird.

2.2 Mensch-Computer-Interaktion

Erhöhte Bedeutung der Benutzerschnittstelle

Seit den Anfängen der Computernutzung hat sich nicht nur in technischer Hinsicht einiges getan. Softwarebasierte Systeme sind heute weitverbreitet und nicht mehr nur Expertenwerkzeuge. Einfache Schnittstellen wurden ersetzt durch graphische Oberflächen, wodurch komplexe Interaktionen möglich sind. UIs sind nicht mehr nur auf Computer begrenzt, sondern nahezu allgegenwärtig, weshalb es immer wichtiger wurde, die Möglichkeiten der Interaktion intuitiv und klar für den Benutzer zu halten (Preece et al., 1994). Das UI dient sowohl der Kontrolle als auch zur Steuerung des Systems, weshalb besondere Aufmerksamkeit beim Spezifizieren von Anforderungen für UI nötig ist (Hardtke, 2004), (Rosson und Carroll, 2002).

Vorgehensweisen nicht ausreichend

Nicht nur McCoy (2002) hat festgestellt, dass die herkömmlichen Methoden des Software Engineering nicht ausreichen, um dies sicherzustellen. Oftmals ist die Software in funktioneller Hinsicht von hoher Qualität, entspricht aber nicht den Aufgaben und Bedürfnissen der Benutzer. Dies liegt vor allem daran, dass die Belange der Nutzer innerhalb der Vorgehensweisen des SE eine untergeordnete Rolle spielen. So wird die Modellierung von UIs in vielen Methoden der Softwareentwicklung unterlassen, ist eher ein Beiprodukt oder geschieht am Ende der Softwareentwicklung (Dix et al., 2003, p. 235), (Pyla et al., 2004), (Strahonja und Picek, 2005). Diese Herangehensweise stammt noch aus den Zeiten, als das UI lediglich eine textbasierte Kommandozeile war, die von Experten dazu benutzt wurde, Funktionen der Software aufzurufen und zu starten (Bowen, 2005).

Geringe Akzeptanz

Dieser Ansatz ist für heutige Softwareentwicklungen nicht mehr tragbar. Es führt dazu, dass die Benutzer nicht mit dem System zurechtkommen und die Nutzung ablehnen, wodurch die Funktionalität belanglos wird (Bevan, 1999). Die Arbeitsschritte laufen langsam ab, das Ziel wird nur umständlich erreicht und der Benutzer wird immer mehr demotiviert, was sich schlussendlich in einer verringerten Akzeptanz des Systems äußert (Memmel et al., 2007a). Da es aus der Sicht des Benutzers keinen Unterschied zwischen UI und System gibt (Hix und Hartson, 1993), stehen oder fallen Akzeptanz und Absatz eines Produktes mit dessen Benutzeroberfläche (Douglas et al., 2002).

Verstärktes Einbeziehen des Benutzers

Um eine bessere Akzeptanz zu erreichen, ist es deshalb nötig, dass die Bedürfnisse des Benutzers schon beim Entwurf eines Systems und nicht nach oder während der Entwicklung einbezogen werden (Lozano et al., 2001). Mitte der achtziger Jahre kam deshalb der Forschungsbereich der Mensch-Computer Interaktion (engl. *Human-Computer Interaction*) auf. Es wurde erkannt, dass die Interaktion zwischen Mensch und Computer nicht nur durch gutes Design beeinflusst wird.

> „Human-Computer Interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them."
>
> (ACM SIGCHI, 1992, p. 6)

Multidisziplinäre Wissenschaft

Human-Computer Interaction (HCI) ist eine multidisziplinäre Wissenschaft, die viele Fachgebiete einbezieht und nicht auf technische oder computerbezogene Belange reduziert werden kann (siehe Abb. 2.4). Die Methoden und Techniken der HCI basieren auf einem Mix aus vielen Disziplinen, besonderen Einfluß haben dabei die Bereiche Informatik, Psychologie, Sozialwissenschaften sowie (Software-) Ergonomie (Preece et al., 1994, p. 50).

Zielsetzung bei der Förderung von Usability

Grundlegendes Ziel der HCI ist die Förderung der Gebrauchstauglichkeit (engl. *Usability*) von Benutzerschnittstellen. Dazu ist es notwendig, funktionelle, benutzbare und gut konzipierte Bedienoberflächen für interaktive Systeme zu schaffen, wodurch sie leicht zu handhaben sind und es dem Benutzer erlauben, seine Arbeit konzentriert und produktiv zu erledigen (Granic und Glavinic, 2002), (Hix und Hartson, 1993), (Constantine und Lockwood, 1999a). Nach Ansicht von Nielsen (1994, p. 23) definiert sich Usability durch 5 Kriterien: *Learnability, Efficiency, Memorability, Errors und Satisfaction* (siehe Tab. 2.1). Durch geeignete Berücksichtigung dieser Kriterien kann das primäre Ziel – die Verbesserung der Arbeitsleistung von

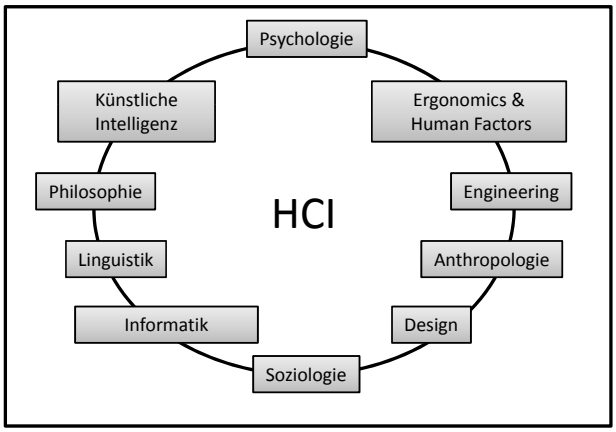


Abbildung 2.4: Übersicht über Fachgebiete, welche in die Mensch-Computer Interaktion involviert sind (Lozano et al., 2001)

Anwendern – erreicht werden (Hix und Hartson, 1993, p. 23). Weitere Definitionen und Anmerkungen zu *Usability* können z. B. bei Whiteside et al. (1988), Carroll und Rosson (1985) und Shackel (1986) entnommen werden.

Standardisierte Normen für die Realisierung

Diese Definition von Usability ist für die praktische Anwendung allerdings zu mehrdeutig und es können keine konkreten Anforderungen für Usability ermittelt werden (Jokela et al., 2003). Die Normen für die *Benutzerorientierte Gestaltung interaktiver Systeme* (ISO/IEC, 1999) und die *Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten* (ISO/IEC, 1998) haben sich deshalb als wichtige Bewertungsmaßstäbe für Usability etabliert. Besondere Berücksichtigung sollte dabei das Kapitel über die Grundsätze der Dialoggestaltung in ISO/IEC 9241 erfahren, da dort wichtige Prinzipien aufgestellt werden (siehe Tab. 2.2). Beide Normen spielen deshalb eine wichtige Rolle, um eine hohe Akzeptanz der Benutzer für die Schnittstelle und damit das System zu erzeugen. Neben *User Experience* (Alben, 1996) wirken Aspekte des *Joy-of-Use* (Reeps, 2004) ebenfalls unterstützend, weil sie eine besondere Bindung des Benutzers zum System hervorrufen.

Abgrenzung zu Interaction Design

Im Gegensatz zu *Interaction Design* (ID) beschränkt sich HCI auf das Design, die Evaluation und die Implementierung von interaktiven Computersystemen für den menschlichen Gebrauch. Der Geltungsbereich ist größer und unabhängig vom Anwendungsbereich, da sich ID generell mit Schnittstellen zu Menschen beschäftigt (Sharp et al., 2007). ID hat sich dagegen als Sammelbegriff für zahlreiche Bereiche herausgebildet, welche sich mit Design beschäftigen (z. B. *User Interface Design*, *Software Design*, *Product Design*, *Web Design*, *Experience Design* und *Interactive System Design*).

Im Bereich des HCI haben sich eine Reihe von Ansätzen entwickelt, von denen einige im Folgenden vorgestellt werden. Die Ansätze unterscheiden sich in ihren Methoden, basieren aber meist auf der Integration von Anwendern in den Entwurfsprozess. Zudem werden mehrere Iterations- und Verfeinerungsstufen mit anschließenden Evaluationen (Weinberg und Stephen, 2002), (Gould und Lewis, 1985) durchgeführt.

Kriterium	Beschreibung
Erlernbar	Ist die Bedienung von Aufgaben auch beim ersten Mal einfach zu beherrschen?
Effizient	Wie schnell und produktiv können Aufgaben ausgeführt werden?
Nachvollziehbar	Wie gut können Arbeitsschritte ausgeführt werden, die schon länger nicht mehr getätigt wurden?
Fehleranfällig	Wie viele Fehler werden bei der Erfüllung einer Aufgabe gemacht und wie einfach können diese wieder behoben bzw. rückgängig gemacht werden?
Zufriedenstellend	Wie ansprechend ist die Nutzung?

Tabelle 2.1: Merkmale von Usability (Nielsen, 1994)

Prinzip	Beschreibung
Aufgabenangemessenheit	Die Arbeitsaufgabe des Benutzers ist effektiv und effizient zu erledigen. -> Sinnvolle Funktionalität, Komfort, Nützlichkeit
Selbstbeschreibungsfähigkeit	Der nächste Schritt ist unmittelbar verständlich, entweder durch Rückmeldung des Systems oder auf Anfrage des Benutzers. -> Offensichtlichkeit, Verständlichkeit, Explorationsfähigkeit
Steuerbarkeit	Die Interaktion mit dem System ist durch den Benutzer in Richtung und Geschwindigkeit beeinflussbar. -> Kontrolle beim Benutzer, Adaptivität
Erwartungskonformität	Konsistenz ist wichtig, zudem müssen die Merkmale des Benutzers berücksichtigt werden. -> Konsistenz, Feedback, Transparenz
Fehlertoleranz	Trotz fehlerhafter Eingaben und mit minimalem Korrekturaufwand kann das Arbeitsergebnis des Benutzers erreicht werden. -> Fehlermanagement, Fehlermeldungen
Individualisierbarkeit	Individuelle Anpassungen an die Erfordernisse der Arbeitsaufgabe und an die Fähigkeiten und Vorlieben des einzelnen Benutzers sind möglich. -> Wahl der Sprache, der Dialogtechnik, der Form des Dialogs
Lernförderlichkeit	Der Benutzer wird beim Erlernen des Dialogsystems unterstützt und angeleitet. -> Relevante Lernstrategien, Lernunterstützung

Tabelle 2.2: Normierte Prinzipien der Usability (ISO/IEC, 1998)

2.2.1 User-Centered Design

Vorherrschende Philosophie

Ein erster Ansatz, den Benutzer stärker in den Entwurfsprozess zu integrieren, ist User-Centered Design (UCD), das ebenfalls unter dem Begriffen *Human-Centered Design* (HCD) oder *Usability Engineering* (UE) bekannt ist (Norman und Draper, 1986), (Pyla et al., 2004). UCD hat sich zu einer vorherrschenden Philosophie innerhalb der HCI entwickelt, weil interaktive Systeme heute nicht mehr nur von Experten benutzt werden (Constantine und Lockwood, 2003).

Die Benutzer sind keine Spezialisten, die technikaffin sind und sich mit der darunterliegenden Technologie auskennen, sondern Personen, die Softwaresysteme anwenden, um ihre tägliche Arbeit zu erledigen. Das System sollte im Stillen arbeiten, da technische Aspekte für diese Anwendergruppe uninteressant sind und sie sich voll auf ihre Aufgaben konzentrieren wollen (Beyer und Holtzblatt, 1997).

Phasen des User Centered Design

Da Benutzer eine zentrale Bedeutung spielen, handelt es sich beim UCD Vorgehensmodell (siehe Abb. 2.5) um einen an den späteren Benutzern orientierten iterativen Prozess (Preece et al., 1994, p. 50) mit vier Aktivitäten. Nach einer Planungsphase wird der erste Durchlauf gestartet und der Nutzungskontext festgelegt. Dies beinhaltet z. B. die Identifikation von Interessengruppen (engl. *Stakeholdern*) und die Aufgaben, welche mit dem System erledigt werden sollen. Danach geht es im zweiten Schritt um die Erhebung von Anforderungen, die erfüllt werden müssen. Sie werden entweder durch die Benutzer definiert oder sind aus unternehmerischer Sicht erforderlich.

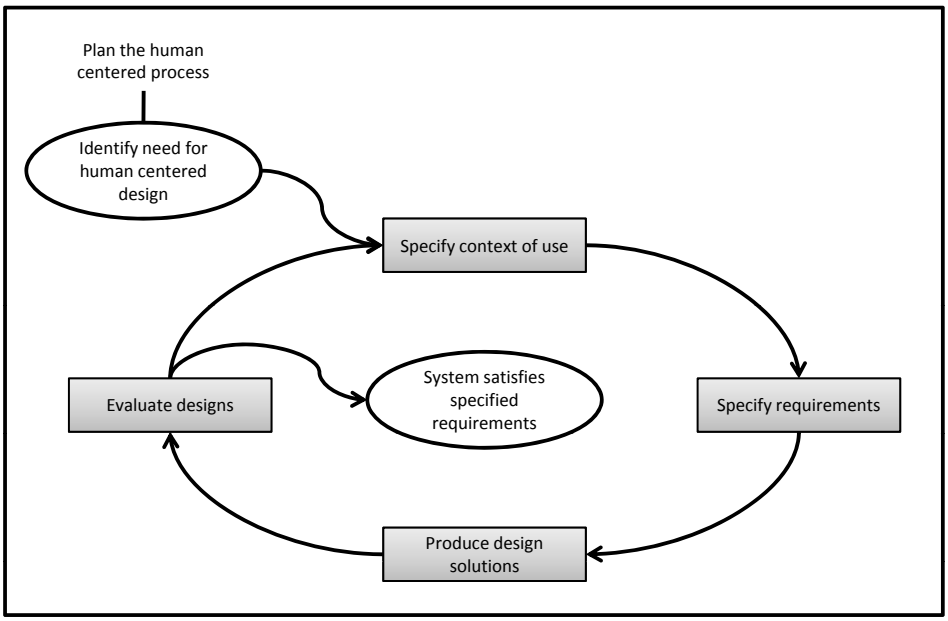


Abbildung 2.5: Entwicklungsprozess für interaktive Systeme (ISO/IEC, 1999)

Die nächste Phase befasst sich mit dem Entwurf von Lösungsmöglichkeiten, welche immer weiter verfeinert werden. Anschließend findet im letzten Schritt die Evaluation des Entwurfs gegen die Anforderungen statt. Der Prozess endet, wenn das System alle erhobenen Anforderungen erfüllt. Falls dies nicht der Fall ist, wird eine neue Iteration durchgeführt. Die wichtigsten Methoden des UCD sind iteratives Design, Usability Evaluationen, Task Analysen, Experten Reviews und Feldstudien (Vredenburg et al., 2002).

2.2.2 Participatory Design

Benutzer werden sehr stark einbezogen

Participatory Design (PD) ist ein Ansatz, bei dem die späteren Benutzer stärker in die Entwurfsphase und in die zu treffenden Entscheidungen mit einbezogen werden. Benutzer sind nicht nur Informanten, die Hinweise über Anforderungen liefern, sondern Initiatoren (Chin et al., 1997). Aus Sicht des PD sind die Benutzer die primäre Quelle für Innovationen, Entwurfsideen und Informationen während eines Entwicklungsprozesses. Dadurch wird sichergestellt, dass der Nutzer mit dem System interagieren kann, da er die Funktionalität der Software versteht und die Darstellung nachvollziehen kann. Deshalb ist klar, dass PD nur mit wirklichen Nutzern und nicht mit imaginären abstrakten Personen funktionieren kann. Eine wichtige Rolle während des Entwurfs spielt dabei kollaboratives Prototyping. Weitere Informationen zu PD sind in der Literatur bei Schuler und Namioka (1993), Kensing und Blomberg (1998), Muller und Kuhn (1993) sowie Muller (2003) zu finden.

2.2.3 Contextual Design

Merkmale der Benutzer werden untersucht

Das von Beyer und Holtzblatt (1997) entwickelte *Contextual Design* (CD) beinhaltet einige Ideen aus PD und UCD (Blomquist und Arvola, 2002). Sie führen an, dass die Arbeit der späteren Nutzer komplex und sehr detailreich ist. Dies macht es Entwicklerteams schwer, den Arbeitsablauf der Anwender niederzuschreiben oder darüber zu diskutieren, weil sie nicht über die Fähigkeiten für einen detaillierten Einblick verfügen. Die Mitglieder des Entwicklerteams benötigen deshalb Kenntnisse in den Methoden der Ethnografie, genauer gesagt in den Techniken der Anthropologie (Simonsen und Kensing, 1997), (Muller und Kuhn, 1993).

Die Anthropholgie ist eine Methodik, die es ermöglicht, menschliches Handeln zu analysieren. Dadurch kann ein besseres Verständis für die Aufgaben und Ziele von Personen entwickelt werden und wie dies durch interaktive Systeme unterstützt werden kann.

> „The respect for the uniqueness of each use situation, the concern for work design rather than mere computer system design, and the use of user-accessible design techniques and expressions all closely relate contextual design to participatory design."
>
> (Löwgren, 1995)

Modellbasierter Ansatz

Im CD werden Benutzer ebenfalls in allen Phasen des Entwurfsprozesses mit einbezogen. Graphische Notationen werden verwendet, da die Dokumentation der Arbeitsabläufe mit Text sehr schwierig ist und um Entwicklern die Analyse zu erleichtern. Deshalb kann bei CD auch von einem modellbasierten Ansatz gesprochen werden. Das Vorgehensmodell gliedert sich in mehrere Abschnitte auf (siehe Abb. 2.6).

Im ersten Schritt namens *Contextual Inquiry* werden Bedürfnisse, Wünsche und Arbeitsweisen der späteren Anwender genauer untersucht. Interviews unmittelbar am Arbeitsplatz sind für diesen Vorgang eine gängige Technik. Es soll eine Diskussion gestartet werden, bei der eine gemeinsame Auswertung der Arbeitsvorgänge

zwischen Interviewer und Benutzer entsteht. Nachfolgend werden die gesammelten Daten während des *Work Modeling* in Modelle überführt, weil Text oder Sprache als Ausdrucksformen nicht ausreichen und abstraktere Notationen notwendig sind (Beyer und Holtzblatt, 1997). In der Phase des *Work Model Consolidation* werden die erstellten Modelle verfeinert und zusammengefasst. Dadurch ergibt sich aus der Sicht der einzelnen Anwender eine gemeinsame Darstellung, welche den Ansprüchen aller Benutzern gerecht wird. Über die Stufen *Work Redesign* und *User Environment Design* werden Verbesserungsmöglichkeiten und neue Lösungen gesucht. Diese werden dann in den folgenden Abschnitten und im Zusammenspiel mit den späteren Anwendern in mehreren Iterationen erstellt. Dazu werden papierbasierte Prototypen verwendet, welche daraufhin einer Evaluation unterzogen werden. Die Implementierung erfolgt erst, nachdem der letzte Schritt im CD-Prozess abgeschlossen wurde und sich alle Beteiligten auf einen Umsetzungsentwurf geeinigt haben.

Neben funktionalen und benutzerbezogenen Anforderungen müssen ebenfalls betriebswirtschaftliche Aspekte bei der Entwicklung von Software beachtet werden. Im nächsten Abschnitt wird eine weitere Interessengruppe beschrieben, die ein Interesse an funktionsfähiger Software hat, mit welcher sich Aufgaben effizient und effektiv lösen lassen.

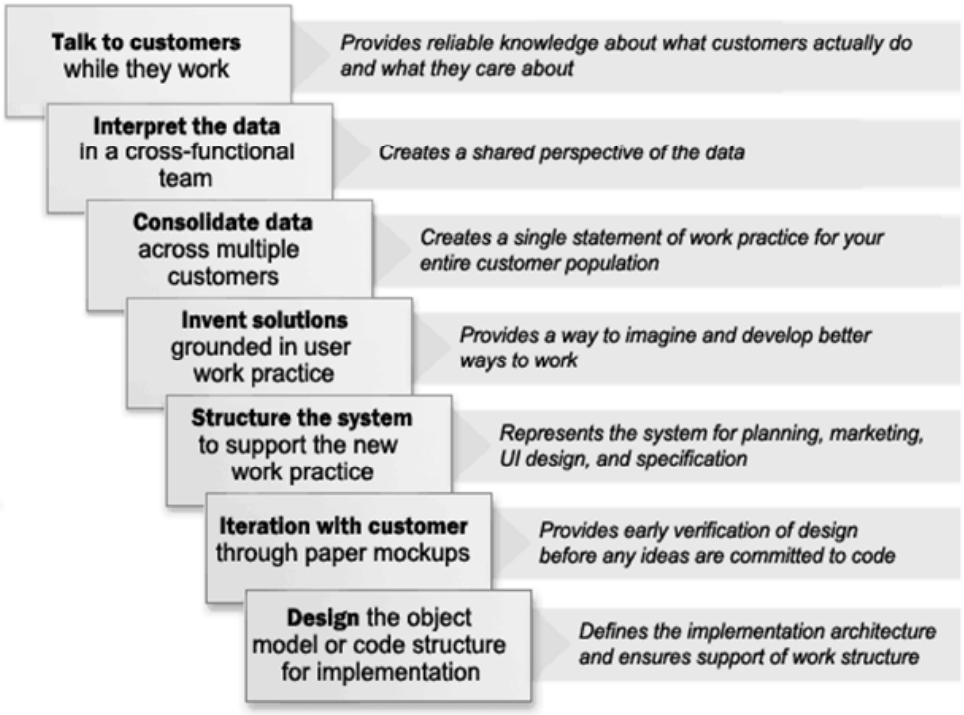


Abbildung 2.6: Der Contextual Process als Vorgehensmodell für das Contextual Design (Beyer und Holtzblatt, 1997)

2.3 Geschäftsprozessmodellierung

Nutzung von informationstechnischem Potential

Das Business Engineering (BE) ist eine Managementdisziplin, welche das Ziel hat, das Potenzial von Technologien im Bereich der Informations- und Datenverarbeitung für das Unternehmen zu nutzen. Dabei wird betriebswirtschaftliches und informationstechnisches Wissen zusammengelegt, um Konzepte und Instrumente zur Planung, Durchführung und Kontrolle von Geschäftslösungen abzuleiten (Österle, 1995), (Österle und Blessing, 2003).

Optimierung von Geschäftsprozessen

Die Geschäftsprozessmodellierung (engl. *Business Process Modeling*) spielt dabei eine wichtige Rolle. Wie Smith (2006) darstellt, ist sie ein geeignetes Mittel, um Probleme und Schwachstellen in Arbeitsabläufen zu identifizieren und zu lösen. Das Ziel ist die Verbesserung von Effektivität und Effizienz bei Prozessen und Abläufen innerhalb eines Unternehmens, um Kosten, Qualität, Service und Zeit zu optimieren. Mithilfe des *Business Process Modeling* (BPM) ist es möglich, geschäftliche Vorgänge zu optimieren, neu zu strukturieren und zur Veranschaulichung modellhaft darzustellen. So können Unternehmen konkurrenzfähig bleiben und sich schnell an geänderte oder neu aufkommende Bedingungen anpassen (Aversano et al., 2005).

Interaktive Systeme und BPM sind verknüpft

Wie Gruhn und Wellen (1998) zeigen, stehen die meisten Geschäftsprozesse in direktem Zusammenhang zu interaktiven Systemen. Geschäftsprozesse sind zwar grundsätzlich unabhängig von der späteren Umsetzung, häufig beinhalten sie aber die Automatisierung von Vorgängen durch Informationssysteme (Oestereich et al., 2003). BPM und Softwaresysteme stehen also in direktem Zusammenhang, da interaktive Systeme zur Unterstützung von Geschäftsprozessen verwendet werden.

Vereinfachte strukturierte Darstellung

Ein *Business Process* (BP) ist abstrakt und zeigt, *was* bewerkstelligt wird, aber nicht, *wie* dies geschieht (Sousa et al., 2008a). Er kann informale, semiformale und formale Darstellungen zur Beschreibung und Visualisierung beinhalten (Oestereich et al., 2003). Dadurch wird es möglich, den schematischen Ablauf eines Geschäftsprozesses (engl. *Business Process*) in einer strukturierten Form abzubilden. Das Erstellen von BPs führt zu einem besseren Verständnis der aktuellen Situation, zeigt Verbesserungsmöglichkeiten auf und hat darüber hinaus auch dokumentatorischen Charakter (Sousa et al., 2008b).

Umgestaltung von Geschäftsprozessen

Ein weiterer Aspekt sind Veränderungen im Geschäftsumfeld, die sich auch immer auf den *Business Process* (BP) auswirken. Deshalb sollte *Business Process Reengineering* (BPR), also die Anpassung der aktuellen Modelle, regelmäßig durchgeführt werden, um ineffiziente und überholte Vorgänge zu identifizieren und zu überarbeiten.

Rückverfolgbarkeit bei Änderungen ist wichtig

Wird die Anpassung eines Geschäftsprozesses nötig, so zieht dies auch immer eine Angleichung des Systems nach sich. Nur dann ist gewährleistet, dass der Vorgang nach einer Modifizierung wieder unterstützt wird. Im Umkehrschluss ziehen Änderungen am System demnach auch immer eine Anpassung der Geschäftsprozesse nach sich (Aversano et al., 2005).

Das UI ist abhängig von Geschäftsprozessen

Welche wichtige Rolle Benutzerschnittstellen bei interaktiven Systemen spielen, wird nochmals von Sauve et al. (2006), Sukaviriya et al. (2007) und Sousa et al. (2008b) erwähnt. Sie zeigen auf, wie eng ein BP mit interaktiven Systemen und deren UI zusammenhängt. Durch die Analyse von Geschäftsvorgängen werden Verbesserungspotenziale erkannt und dadurch auch erste Anforderungen an interaktive

Systeme erhoben (Gruhn und Wellen, 1998).

BPM kann deshalb zum frühen Bestandteil eines Entwicklungsprozesses für interaktive Systeme und dessen UI gezählt werden, da die gesammelten Informationen innerhalb eines BP oftmals die Vorstufe für detailliertere Anforderungsanalysen sind (siehe Abb. 2.7).

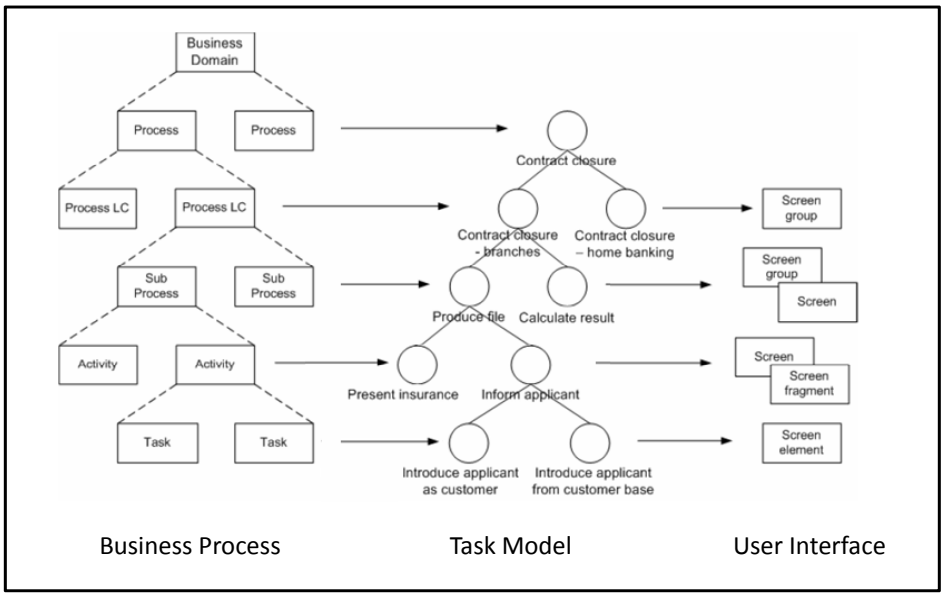


Abbildung 2.7: Zusammenhang von Business Process und Benutzeroberfläche (vgl. (Sousa et al., 2008a))

Prototypen als Kommunikations- und Kooperationsmittel

BPs sind oft zu abstrakt, nicht repräsentativ und nicht detailliert genug um sämtliche Interaktionen abzubilden. Zudem kann das Verhalten von interaktiven Systemen nicht genau beschrieben werden (Sousa et al., 2008a). Sukaviriya et al. (2007) schlagen deshalb vor, komplexe Vorgänge am System zum besseren Verständnis durch informale UI Prototypen zu visualisieren. Gerade im Umgang mit Kunden oder Benutzern kann ein besseres Verständnis für Interaktionen und Abläufe innerhalb eines BP erzeugt werden.

> „Geschäftsprozessmodellierer und Softwareentwickler sind meistens verschiedene Personen, die traditionell mit verschiedenen Beschreibungssprachen, Methoden und Werkzeugen arbeiten. [...] Oft werden die Ergebnisse der Geschäftsprozessmodellierung von Softwareentwicklern nicht übernommen oder - wegen des speziellen für sie nicht verfügbaren Werkzeuges - gar nicht gewürdigt."
>
> (Oestereich, 2005)

Keine gängige Praxis

Trotzdem werden Geschäftsprozesse viel zu selten als Grundlage für die Entwicklung eines UI herangezogen (Sousa, 2008a). Die Folge sind zeitintensive Besprechungen, bei welchen überprüft werden muss, ob das UI mit den zugrunde liegenden Geschäftsprozessen übereinstimmt. Dies führt zu Verzögerungen für die spätere Realisierung und ist hinderlich für die Wartung und Anpassung von BP und interaktiven Systemen.

2.4 Disziplinenübergreifende Zusammenarbeit bei der Anforderungsermittlung

Identifikation von Anforderungen

Bevor interaktive Systeme mit hoher Komplexität entwickelt werden, ist es erforderlich, einen genaueren Einblick in die Problemdomäne und den späteren Anwendungsbereich zu erhalten. Ohne Analyse mit der Programmierung zu starten und das zu entwickelnde System dann nach und nach den Bedürfnissen anzupassen, ist keine gute Lösung für komplexe Systeme (Boehm et al., 1984). Es werden zwar relativ schnell Ergebnisse erzielt, doch bis das Produkt fertiggestellt ist, den Bedürfnissen entspricht und eingesetzt werden kann, ist eine kosten- und zeitintensive Überarbeitung notwendig. Interaktive Systeme setzen sich aus mehreren Bestandteilen, wie etwa Software und Hardware zusammen, weshalb es wichtig ist, dass neben spezifischen Anforderungen auch implizite Bedingungen für alle Komponenten erkannt werden (Rupp, 2007, p. 131).

Ziele der Entwicklung werden definiert

Deshalb wird zuerst eine Anforderungsermittlung (engl. *Requirements Engineering*) durchgeführt und nicht einfach mit der Umsetzung begonnen. Dabei wird der Nutzungskontext und die Anwendungsdomäne betrachtet und die Frage beantwortet, „was“ entwickelt werden soll und „weshalb“ (Zave und Jackson, 1997), (Davis, 1993). Es handelt sich um eine entscheidende Phase, da die Ziele und damit der Erfolg der späteren Entwicklung bestimmt wird. Zweck des *Requirements Engineering* (RE) ist es, eine vollständige, widerspruchsfreie und eindeutige Beschreibung von Anforderungen zu erhalten (Haumer, 2000). Diese werden dokumentiert und dienen dann als Grundlage für die anschließende Umsetzung.

Vollständige, widerspruchsfreie und eindeutige Beschreibung

Im *IEEE Standard Glossary of Software Engineering Terminology* (IEEE, 1990) wird die Anforderungsermittlung und Anforderungen wie folgt festgelegt:

Die Anforderungsanalyse ist:

1. Der Prozess zur Untersuchung von Bedürfnissen einer Person, welcher zur Definition von System-, Hardware- oder Software-Anforderungen führt.

2. Der Prozess zur Untersuchung und Verfeinerung von System-, Hardware- oder Software-Anforderungen.

Anforderungen sind:

1. Eine Bedingung oder Fähigkeit, die von einer Person zur Lösung eines Problems oder zur Erreichung eines Ziels benötigt wird.

2. Eine Bedingung oder Fähigkeit, die ein System oder eine Systemkomponente erfüllen oder besitzen muss, um einen Vertrag, eine Norm oder ein anderes, formell bestimmtes Dokument zu erfüllen.

3. Eine dokumentierte Darstellung einer Bedingung oder Fähigkeit, welche in (1) oder (2) genannt sind.

Was und nicht *Wie* wird ermittelt

Beachtet werden muss, dass Anforderungen keine Verweise auf die Realisierung beinhalten sollten, denn dadurch werden möglicherweise vielversprechende Ansätze zur Verwirklichung bereits im Vorfeld ausgeschlossen (Pohl, 1997). Es wird beschrieben „was" umgesetzt werden soll und nicht „wie" es realisiert wird . Es gibt etliche Techniken und Methoden, um Anforderungen zu ermitteln, z. B. durch Fragebögen, Interviews oder die Untersuchung von Arbeitsabläufen. Anforderungen können nach Robertson und Robertson (1999) und Haumer (2000) in drei Bereiche unterteilt werden, eine Einteilung hängt aber vom Blickwinkel des Betrachters ab (Pohl, 1997):

1. Anforderungen, die die Funktion des Systems und die Interaktion mit den Benutzern betreffen (*Functional or Behavioural Requirements)*.
2. Anforderungen, die definieren wie (1) realisiert werden soll und etwas über die Eigenschaften des Systems aussagen (*Nonfunctional or Nonbehavioural Requirements*) wie z. B. im Bezug auf Ausfallsicherheit, Antwortzeiten oder Wartungsfreundlichkeit.
3. Randbedingungen, die das System als Ganzes betreffen und die (1) und (2) begrenzen (*Constraints or Global Requirements*) wie z. B. Vorgaben für die zu verwendende Programmiersprache.

Anforderungen ändern sich

Ein weiterer Punkt, der beachtet werden muss, ist die Tatsache, dass sich Anforderungen im Laufe des Entwicklungsprozesses ändern und entwickeln (Nuseibeh und Easterbrook, 2000), (Cao und Ramesh, 2008). Dadurch können sie sich als inkompatibel, überholt und widersprüchlich erweisen (Minocha, 1999). Wie Sommerville (2005) darstellt, ist dies unausweichlich und z. B. darauf zurückzuführen, dass sich neue Erkenntnisse und Einsichten erst während des Entwicklungsprozesses ergeben oder dass sich das betriebliche Umfeld und die Rahmenbedingungen ändern.

Traceability ist wichtig

Deshalb muss es möglich sein, die zugrunde liegenden Anforderungen identifizieren und anpassen zu können, weshalb die Verbindungen zwischen Anforderung und Umsetzung nachvollziehbar sein müssen. Diese Rückverfolgbarkeit (engl. *Traceability*) von Anforderungen ist dabei sowohl für den Entwicklungsprozess, als auch für das fertige System bzw. dessen Wartung unbedingt erforderlich.

Buxton (2003) listet auf, dass schlechte Qualität, geringe Innovation, hohe Kosten und verspätete Auslieferung das Ergebnis vieler Softwareentwicklungen kennzeichnen. Dies ist darauf zurückzuführen, dass Projekte zu früh gestartet werden, ohne dass eine ausführliche Analyse aller Anforderungen durchgeführt wurde.

> „Requirements traceability refers to the ability to describe and follow the life of a requirement, in both a forwards and backwards direction."
>
> (Gotel und Finkelstein, 1994)

Analyse der Anwendungsdomäne

Bevor erste Anforderungen erhoben werden, ist deshalb eine genaue Analyse der Anwendungsdomäne nötig, um eine möglichst wertungsfreie Darstellung der aktuellen Situation zu bekommen (Finkelstein, 1993). Dazu gehört neben der Abgrenzung der Aufgabenstellung auch die genauere Erfassung der Projektziele sowie

die Prüfung von Durchführbarkeit, Aufwand, Nutzen und Risiken. Es wird versucht, ein besseres Verständnis des Aufgabenbereichs, der Abhängigkeiten und des späteren Systemaufbaus zu bekommen. Zusätzlich fließen bisherige Erfahrungen, Standards und Regeln sowie Expertenwissen mit ein. Mit einer fundierten Kenntnis der aktuellen Situation, der Probleme und Optimierungspotenziale können anschließend die Ziele der Entwicklung definiert und Benutzer ermittelt werden.

Interdisziplinäre Zusammenarbeit

Wenn der Umsetzung zugestimmt werden kann, so wird die Anforderungsanalyse gestartet, bei welcher Beteiligte aus den jeweiligen Fachrichtungen zusammenarbeiten müssen, um die Ansprüche und Abläufe zu klären (Paech und Kohler, 2003). Zudem werden konkrete Benutzer- und Aufgabenanforderungen ermittelt. Abbildung 2.8 zeigt das Zusammenwirken der drei Disziplinen im Laufe eines Entwicklungsprozesses.

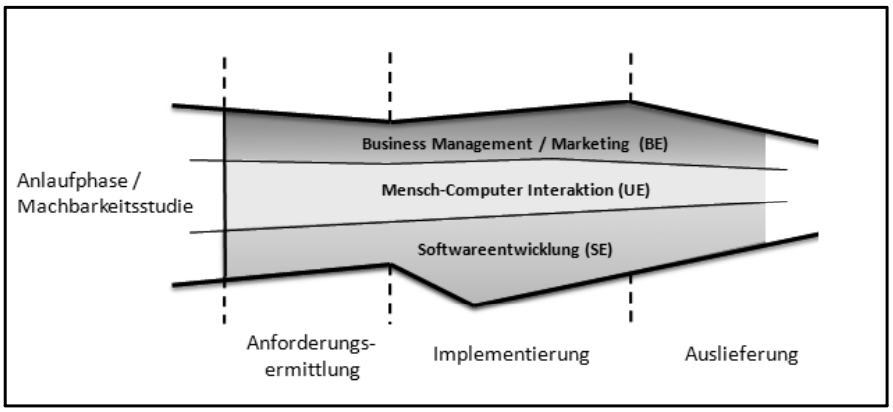


Abbildung 2.8: Das Zusammenwirken der verschiedenen Disziplinen und ihr approximativer Arbeitsumfang und -aufwand während der Softwareentwicklung in Anlehnung an Buxton (2003)

Entwicklungsprozess beinhaltet Anforderungsanalyse

Ist dieser Abschnitt abgeschlossen und die Anforderungen erhoben, dienen diese als Spezifikation für die - intern oder extern erfolgende - Umsetzung. Wird ein externer Dienstleister hinzugezogen, so übernimmt dieser die Implementierung und liefert am Ende ein funktionierendes System, das anschließend validiert wird. Sind nicht alle Anforderungen erfüllt oder werden Probleme identifiziert, so muss der Dienstleister nachbessern. Gegen Ende dieses Prozesses verringert sich der Arbeitsaufwand, weil das System ausgeliefert werden kann und nur noch letzte Änderungen eingepflegt werden müssen.

> „Many delivered systems do not meet their customers's requirements due, at least partly, to ineffective Requirements Engineering."
>
> (Nuseibeh und Easterbrook, 2000)

Anforderungsanalyse ist die schwierigste Phase

Bei der Anforderungsanalyse handelt es sich um die schwierigste Phase bei der Erstellung von Software, da hier die Weichen für die spätere Umsetzung gestellt werden. Wie Brooks (1987) feststellt, können die dort getroffenen Entscheidungen im späteren Verlauf nur sehr schwer abgeändert werden. Es gilt deshalb diesen Schneeballeffekt und die damit verbundenen hohen Kosten zu verhindern, weshalb eine enge interdisziplinäre Zusammenarbeit bereits in diesem Stadium wichtig ist.

2.5 Zusammenfassung

Interdisziplinäre Kooperation

Anforderungen an die Benutzerschnittstelle haben auch Auswirkungen auf den funktionalen Teil des Systems und darüber hinaus auf dessen Architektur (Bass und John, 2001). Die funktionale systembasierte Sichtweise des SE muss aus diesem Grund mit der benutzerorientierten Sichtweise des HCI verbunden werden. Zudem dürfen die betriebswirtschaftlichen Hintergründe nicht unbeachtet bleiben, weshalb die drei Disziplinen UE, SE und BE bereits während der Anforderungsanalyse kooperieren müssen (siehe Abb. 2.8).

Alle Fachbereiche sind wichtig für die Erstellung von qualitativ hochwertigen, innovativen, funktionalen und gebrauchstauglichen Systemen. Keine Disziplin kann alleine erfolgreich sein, vor allem nicht, wenn der vorgesehene Rahmen für Zeit und Budget nicht überschritten werden soll. Wie Memmel und Reiterer (2008c) zeigen, ist es nur dann möglich, benutzer- und aufgabengerechte UIs zu entwickeln, wenn die Akteure aus allen Interessengruppen in der Lage zur kollaborativen Zusammenarbeit sind.

Kapitel 3

Ansätze zur Modellierung von Benutzerschnittstellen

In diesem Kapitel werden Entwurfs- und Spezifikationsmethoden für Benutzerschnittstellen vorgestellt, die sowohl funktionale als auch benutzerbezogene Sichtweisen beinhalten. Im Anschluß findet eine Bewertung statt, mit der festgestellt werden soll, ob diese Herangehensweisen mit den formulierten Zielen dieser Arbeit vereinbar sind. Es wird besonderen Wert darauf gelegt, ob diese Vorgehensweisen von interdisziplinären Teams für die Spezifikation von Benutzerschnittstellen verwendet werden können. Anschließend gibt es einen Überblick über aktuelle Werkzeuge, welche bei der Modellierung von UIs verwendet werden können, um herauszufinden ob es bereits geeignete Hilfmittel zur Erstellung von Spezifikationen gibt.

3.1 Brückenschläge zwischen den Disziplinen

Trennung von System und UI

Im folgenden Kapitel werden aktuelle Modellierungsansätze für die Spezifikation von UIs vorgestellt, welche versuchen die Kluft zwischen funktionaler und benutzerorientierter Sichtweise zu überbrücken. Bei der Entwicklung von interaktiven Systemen ist die Trennung von UI und Applikation eine weitverbreitete und lösungsorientierte Herangehensweise. Konzepte wie das *Seeheim Model* (Pfaff, 1985), das *Arch Model* (SIGCHI Bull., 1992) oder das *Model-View-Controller* (MVC) *Design Pattern* (Goldberg und Robson, 1983) unterstützen dieses Vorgehen. Dadurch ist es möglich, dass die Aufgaben verteilt werden können und verschiedene Modellierungstechniken beim Entwurf Verwendung finden (Bass und John, 2001), (Bowen und Reeves, 2007).

Präzise Definitionen durch formale Modelle

Die Modellierungsansätze dieses Kapitels beinhalten sowohl formale als auch informale Modelle. Formale Techniken reichen von strikten mathematischen Modellen bis hin zu weniger formalen Notationen und Diagrammtechniken (Furniss et al., 2005) und besitzen eine genau definierte Semantik (Paternò, 1999) (siehe Kapitel 4.1.3—"Formale Ausdrucksweisen"). Dadurch erlauben sie es den Entwicklern, die Entwürfe des Systems an den formal definierten Anforderungen präzise und korrekt zu überprüfen (Bowen und Reeves, 2007). Sie lassen sich einteilen in solche, die für die Konzeption und Konstruktion des Systems geeignet sind und andere, die

besser für den Entwurf des Verhaltens angebracht sind. Erstere sind besonders zur Beschreibung der sogenannten Applikationslogik eines Systems geeignet und um die funktionalen Aspekte sowie das zugrunde liegende Systemverhalten zu modellieren.

Spezifikation von Architektur und Verhalten des Systems

Beispiele für formale Notationen zum Beschreiben von Systemen sind z.B Die Sprache Z (ISO/IEC, 2002), eine Prädikatenlogik erster Stufe, die objektorientierte Variante *Object-Z* (Carrington et al., 1990), und die Strukturdiagramme der UML. Beispiele für die Spezifikation von Interaktionsdesign und des Verhaltens von UIs sind z. B. *State Transition Diagramme* (Dix et al., 2003, p. 582), *User Action Notation* (UAN) (Hartson et al., 1990), *Finite State Machines* (FSM) (Seshu, 1963), *ConcurTaskTrees* (CCT) (Paternò, 1999) sowie die Verhaltensdiagramme der UML.

Informale Modelle für kreative Vorgänge

Im Gegensatz dazu sind informale Techniken besonders geeignet für kreative Vorgänge, da sie mehrdeutige Interpretationen zulassen und somit Handlungsspielraum für Entscheidungen vorhanden ist (siehe Kapitel 4.1.4—"Informale Ausdrucksweisen"). Dies empfiehlt sich gerade zu Beginn des Entwicklungsprozesses, wenn es erforderlich ist, ein erstes Verständnis der Anforderungen zu erhalten und noch nicht klar ist, wie das System auszusehen hat (Paternò und Volpe, 2005). Deshalb werden informale Techniken bevorzugt in der HCI eingesetzt. Nicht genau definiert ist hingegen, wie sich formale und informale Methoden unterscheiden. Antworten auf diese Frage unterscheiden sich je nach Hintergrund des jeweiligen Autors (SE, HCI, Künstliche Intelligenz, etc, ...). Für eine nähere Definition von informalen Methoden wird deshalb die Definiton von Bowen und Reeves (2007) herangezogen. Informal in diesem Zusammenhang bedeutet nicht, dass diese Methoden unstrukturiert sind oder aus dem Stegreif durchgeführt werden, sondern dass die Artefakte, die sie erzeugen, informaler Natur sind. Detailliertere Erklärungen zu den Unterschieden zwischen informalen und formalen Techniken sind in Kapitel 4.1.2—"Modellbasierte Spezifikation" zu finden.

3.1.1 UML und UI Design

UML nicht ausreichend für das UI Design

Wie in Kapitel 2.1—"Softwareentwicklung" erwähnt wurde, hat sich UML zur allgemeinen Norm als Notationssprache für die modellbasierte Entwicklung für Software im Bereich des *Software Engineering* etabliert, da funktionale Aspekte sehr gut abgebildet werden können (da Silva und Paton, 2000b), (Glinz, 2000), (Paternò, 2001), (Nunes und Cunha, 2001), (Trætteberg, 2002b). Deshalb scheint die UML-Notation auf den ersten Blick sehr gut für die Modellierung von UIs geeignet zu sein, da keine neue Notation entwickelt werden muss. Bei der Entwicklung von UML wurde jedoch wenig Wert auf Modelle gelegt, die den Entwurf und die Entwicklung von interaktiven Komponenten eines Systems unterstützen. Struktur und Verhalten von Systemen kann sehr gut modelliert werden (da Silva und Paton, 2001), allerdings ist UML weniger geeignet für das Modellieren von UIs wie z. B. bei Glinz (2000), Markopoulos und Marijnissen (2000), Nunes und Cunha (2000a), da Silva und Paton (2000b), Paternò (2001), Ambler und Jeffries (2002), da Silva und Paton (2003) oder Almendros-Jimenez und Iribarne (2005) nachzulesen ist.

„There is a common misconception that the same models developed to support the design of the application internals are also adequate to support interaction design, leveraging the usability aspects of the applications."

(Nunes und Cunha, 2000a)

3.1.2 UML for Interactive Systems

UML in Verbindung mit Scenarios und Task Models

Paternò (2001) beschreibt in seinem Ansatz *UML for Interactive Systems* nicht nur verwendbare Modelle, sondern einen systematischen Entwicklungsprozess zur Modellierung von UIs (siehe Abb. 3.1). Er führt an, dass die Kommunikation zwischen System und Benutzer nicht in geeigneter Form in UML dargestellt werden kann. Um diese Einschränkung zu umgehen, ergänzt er die *Use Case* Diagramme der UML mit *Scenarios* und *Task Models*. Die Use Case Diagramme werden zur Modellierung von systeminternen Verhalten benutzt. *Task Models* sind in modellbasierten Ansätzen für den Entwurf von Benutzerschnittstellen weit verbreitet (Elnaffar und Graham, 1999). Sie beschreiben die Aktivitäten, die zum Erreichen von Benutzerzielen durchgeführt werden müssen und können den Zugriff des Benutzers auf funktionale Aspekte des Systems darstellen. *Task Models* werden nicht nur bei der Entwicklung von Interaktiven Applikationen und dem Entwurf von UIs angewendet, sondern können auch bei der Anforderungsanalyse, bei Evaluationen oder zu Dokumentationszwecken benutzt werden. *Scenarios* sind eine etablierte Technik aus dem Umfeld des HCI, die das Verständnis von interaktiven Systemen verbessern (Carroll und Rosson, 1990), (Sutcliffe, 2003). Sie liefern informale Beschreibungen über einen spezifischen Anwendungsfall bei der Benutzung des Systems.

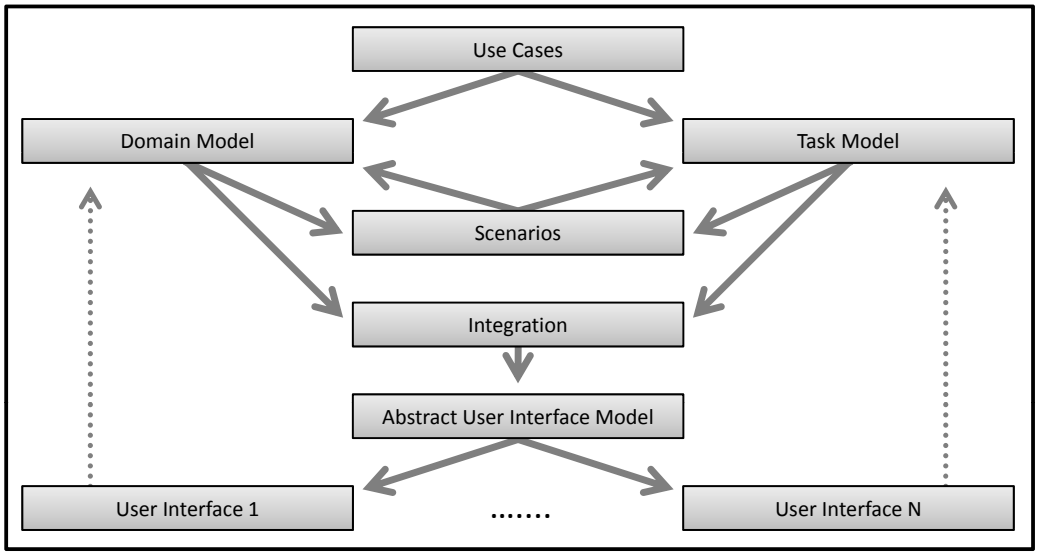


Abbildung 3.1: Prozess zur Modellierung von Benutzerschnittstellen nach Paternò (2001)

CCT zur Visualisierung von Task Models

Als Nächstes werden die *Task Models* mithilfe von *ConcurTaskTrees* (CCTs) generiert, um die Interaktionen des Benutzers mit dem System darzustellen. CCT sind eine Spezifikationstechnik, mit der es möglich ist, Aufgaben hierarchisch aufzugliedern und grafisch in Baumform darzustellen. Zudem können zeitliche Abhängigkeiten und eine Reihe von weiteren (optionalen) Attributen wie z. B. Rollen oder zu erreichende Ziele angegeben werden. Zusätzlich kommen *Scenarios* als informale Beschreibungen spezieller Anwendungsfälle hinzu.

Parallele Modellierung

Parallel zur Modellierung der Aufgaben läuft die Modellierung des Anwendungsbereiches (engl. *Domain Modeling*) mit UML ab. Das Ziel ist, alle Objekte, die das Arbeitsgebiet beeinflussen und deren Beziehungen untereinander zu identifizieren. Anschließend können Informationen zwischen Domain und *Task Model* zusammengeführt werden, um die Erstellung eines ersten abstrakten UIs zu ermöglichen. Aus diesem werden danach konkretere UIs in mehreren Varianten entwickelt.

Iterativer Prozess

Der Prozess läuft iterativ ab, sodass die Ergebnisse der einzelnen Schritte laufend überarbeitet werden. Paternò (2001) stellt ein Vorgehensmodell vor, mit welchem System und UI parallel entwickelt werden können. Durch die Kombination von UML, *Task Models* bzw. CCT und *Scenarios* wird es möglich, das Verhalten des Systems, die Aufgaben der Benutzer, die Interaktion zwischen Benutzer und System sowie mehrere Varianten des UIs zu spezifizieren.

3.1.3 UMLi - UML for Interactive Applications

Abstrakte Komponenten nicht darstellbar

Auf ein Problem bei der Verwendung von UML zur Spezifikation von UIs wird von da Silva und Paton (2000b) hingewiesen. Es fehlt eine Modellierungsform, welche die Struktur von UIs in frühen Phasen besser beschreiben kann, als dies im Moment mit Klassen- und Objektdiagrammen möglich ist. Durch die frühe Verwendung von detaillierten visuellen Komponenten (engl. *Widgets*), wie etwa *Buttons*, *Radiobuttons* oder *Listviews*, rückt die Gestaltung in die Nähe einer Implementierung. Die Gefahr, sich schon frühzeitig auf ein bestimmtes Aussehen und Verhalten der Oberfläche festzulegen, ist sehr groß. Deshalb ist eine abstraktere Notation vor allem in den frühen Phasen des Entwurfs nötig. Die *Unified Modeling Language for Interactive Applications* (UMLi), eine minimale Erweiterung von UML, wurde deshalb entwickelt

UML for Interactive Applications

um den Entwurf von UIs besser zu unterstützen (da Silva und Paton, 2000a), (da Silva und Paton, 2001). Es kann die Applikation in UML modelliert werden und parallel dazu das UI in UMLi.

Ergänzung der UML

Um das Verhalten des UI aufzuzeigen, wird auf eine Erweiterung des von UML bekannten *Activity* Diagramme zurückgegriffen. Dadurch kann die Ausführung der einzelnen Aktivitäten genauer als mit herkömmlichen UML Aktivitätsdiagrammen modelliert werden (siehe - Abb. 3.2). Für die Modellierung von abstrakten Benutzerschnittstellen kann das *User Interface Diagram* zur Darstellung von visuellen Komponenten verwendet werden. Es ermöglicht die Darstellung von visuellen Komponenten. Anstatt konkret auf das Aussehen und die Anordnung dieser Interaktionsobjekte einzugehen, wird mehr Wert auf die Darstellung von Ein- und Ausgabemöglichkeiten gelegt. Dadurch wird es einfacher, Interaktionen zwischen Benutzer und System unabhängig von der Realisierung festzulegen.

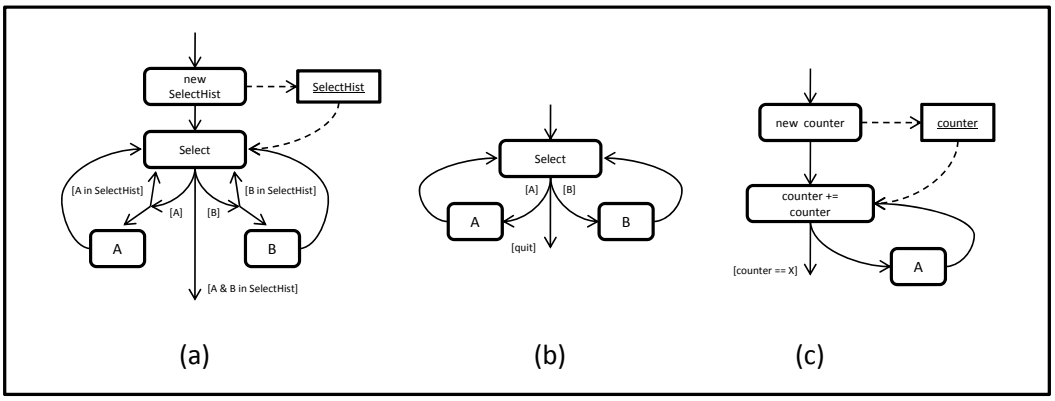


Abbildung 3.2: Erweitertes *Activity Diagram* um Verhaltensweisen darzustellen, deren Abfolge irrelevant ist (a), die optional (b) oder mehrmals durchgeführt werden können (c) (da Silva und Paton, 2000a).

Acht Phasen die parallel zur Systementwicklung ablaufen

Die Modellierung von Benutzeroberflächen mit UMLi gliedert sich in 8 Stufen auf (siehe - Abb. 3.3) und muss in den Prozess zur Modellierung des Systems mit UML eingegliedert werden (da Silva und Paton, 2000a). Begonnen wird mit dem *User Requirement Modeling*, bei welchem Benutzeranforderungen durch die Erstellung von *Use Case* Diagramme und *Scenarios* ermittelt werden. Im nächsten Schritt, der *Interaction Object Elicitation* werden die verwendeten interaktiven Komponenten des UIs mithilfe von *Scenarios* bestimmt. Über die *Interaction Activity Identification* wird festgestellt, in welchen Fällen es zu einer Interaktion zwischen Benutzer und System kommt.

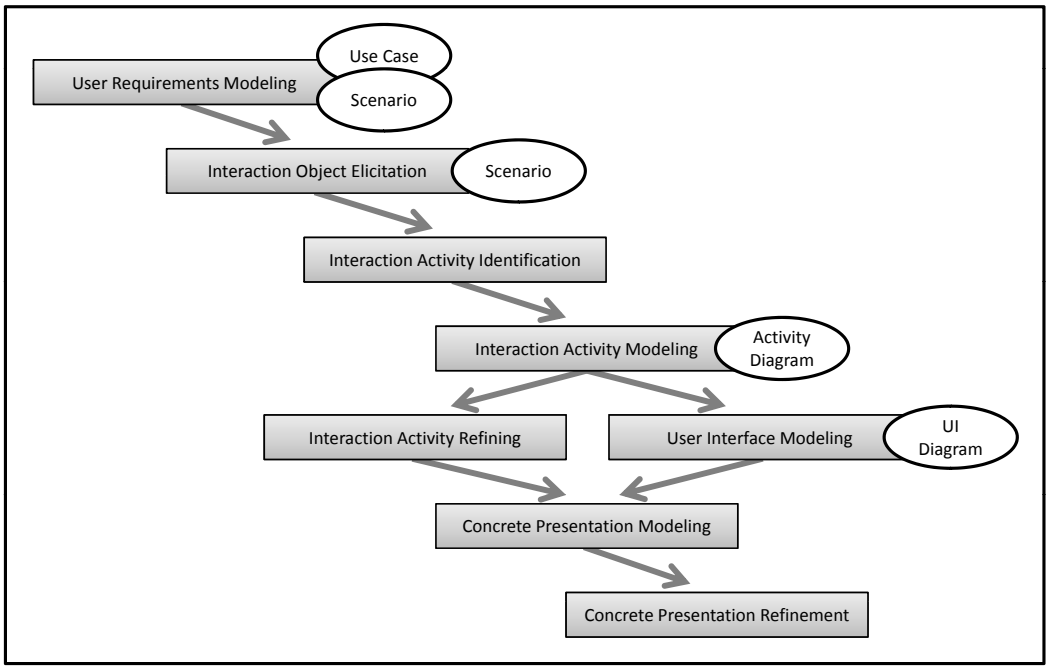


Abbildung 3.3: Modellierungsprozess von UI mit UMLi nach da Silva und Paton (2000a)

Mithilfe von *Activity* Diagrammen werden diese Interaktionen während des *Interaction Activity Modelings* konkretisiert. Die nächsten beiden Abschnitte laufen gleichzeitig ab, wodurch zum einen die *Activity* Diagramme verfeinert werden und zum anderen durch die Verwendung von UI Diagrammen erste abstrakte und konzeptionelle Modelle des UI erstellt werden. In der Phase des *Concrete Presentation Modeling* werden die abstrakten Modelle durch konkrete Widgets ersetzt, wobei durchaus mehrere Varianten auf eine abstrakte Komponente abgebildet werden können. Im letzten Schritt wird das erzeugte UI unter ergonomischen Gesichtspunkten und durch die Verwendung von *Styleguides* verfeinert, um bessere Gebrauchstauglichkeit zu gewährleisten.

Modellierung unabhängig von der Realisierung

Durch die Verwendung der UMLi Notation kann die Gefahr umgangen werden, sich frühzeitig auf ein konkretes *Look & Feel* zu versteifen. Interaktive Komponenten können zwar auch schon in den abstrakten Prototypen vorkommen, ihre spätere Realisierung bleibt aber bis in die späten Phasen offen. Mit der Einführung von erweiterten Aktivitätsdiagrammen wird die Modellierung von komplexen UIs sehr viel einfacher und verständlicher, als dies mit der UML-Notation möglich wäre (da Silva und Paton, 2000a).

3.1.4 Wisdom

Wisdom Ansatz

Ein ähnliches Ziel verfolgen (Nunes und Cunha, 2000a) mit ihrem *Whitewater Interactive System Development with Object Models (Wisdom)* Ansatz, der speziell für kleine Entwicklerteams konzipiert wurde. Bei Wisdom handelt es sich um einen leichtgewichtigen UCD-Ansatz für die Softwareentwicklung von interaktiven Systemen, der zum Teil auf UML basiert. Es werden einige Notationen der UML verwendet, die von neuen Modellen erweitert werden. *Wisdom* besteht aus drei Arbeitsabläufen, sechs Aktivitäten und sieben Modellen (siehe - Abb. 3.4).

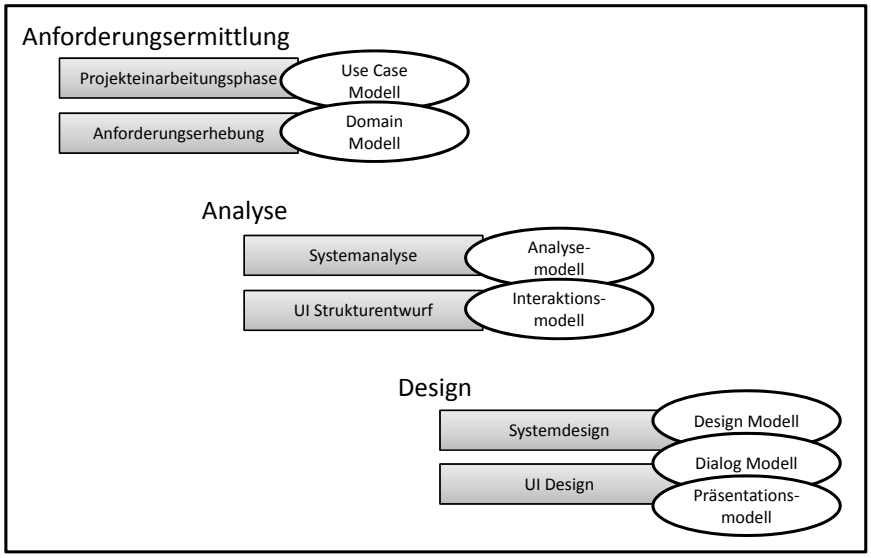


Abbildung 3.4: Arbeitsabläufe, Aktivitäten und Modelle in Wisdom nach Nunes und Cunha (2000b)

Anforderungsermittlung

Im ersten Arbeitsablauf werden erhobene Anforderungen durch *Use Case* Diagramme und Aktivitätsdiagramme ausgedrückt. Verwendet werden *Essential Use Cases* (Constantine und Lockwood, 1999a, 103ff), eine Sonderform der *Use Case* Diagramme. Durch die Nutzung von *Essential Use Cases* können Aufgaben des Benutzers komplett, aussagekräftig und eindeutig definiert werden (Nunes und Cunha, 2000b).

Analysephase

In der nächsten Phase werden die bisher erhobenen Anforderungen durch die Verwendung von Analyse- und Interaktionsmodell verfeinert (siehe - Abb. 3.5). Das Analysemodell entspricht der bisherigen Sichtweise, die durch UML darstellbar ist und wird zur Modellierung der Systemstruktur verwendet. Dadurch können Funktionalitäten, Schnittstellen und internes Verhalten dargestellt werden. Beim Interaktionsmodell handelt es sich um eine äquivalente Ergänzung. Es besteht aus Präsentations- und Dialogdimension und dient zur Darstellung von Interaktionen mit dem Benutzer.

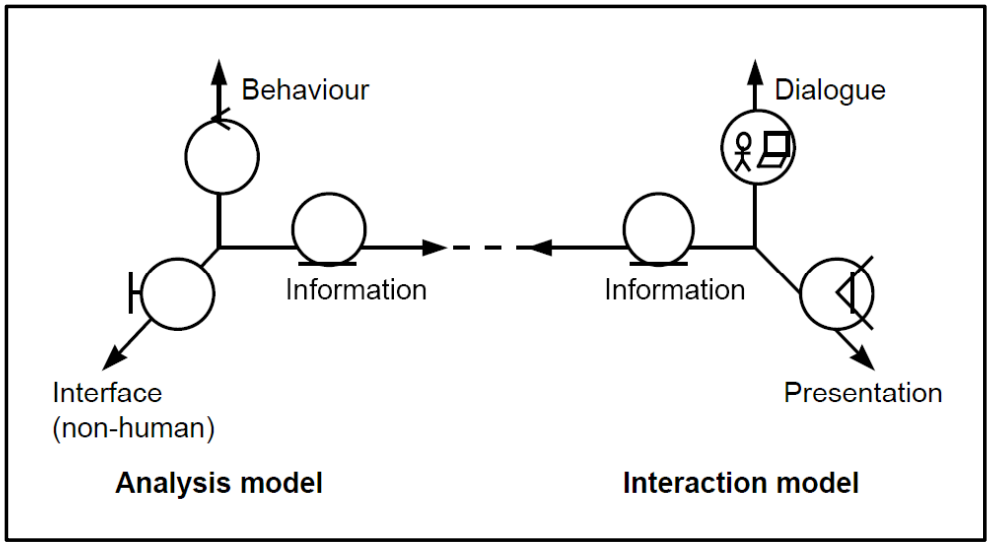


Abbildung 3.5: Der Analyserahmen für Interaktive Systeme nach Nunes und Cunha (2001)

Präsentations- und Dialogdimension

Der Fokus der Präsentationsdimension liegt auf der Modellierung und Strukturierung der einzelnen Widgets, welche die Interaktion mit dem Benutzer möglich machen. Über die Dialogdimension wird der Aufbau der einzelnen Applikationsdialoge spezifiziert, welche für die Interaktion zwischen Benutzer und System notwendig sind. Realisiert werden kann dies z. B. mit CCTs (Nunes und Cunha, 2000a). Die Verbindung zwischen Analysemodell und Interaktionsmodell wird durch die Informationsdimension dargestellt. Dadurch kann ein Datenaustausch stattfinden und es besteht die Möglichkeit auf die selben Informationen zuzugreifen.

Designphase

In der letzten Phase, dem Design wird die Architektur und die Form des UIs genauer beschrieben. Die bisherigen Modelle werden erneut verfeinert und sukzessiv optimiert. Durch die Verwendung von Prototypen mit unterschiedlichem Detailgrad können Entwurfsentscheidungen getroffen und evaluiert werden. Daneben fließen auch erste Randbedingungen ein, welche die spätere Umsetzung in Hard- und Software betreffen.

Leichtgewichtige Vorgehensweise

Nunes und Cunha (2001) stellen mit *Wisdom* eine leichtgewichtige Vorgehensweise zur Entwicklung von interaktiven Systemen vor. Durch die Verwendung von Modellen, die einfach zu lernen und leicht zu benutzen sind, kann der Entwicklungsprozess schnell durchlaufen werden. Dies wird durch einen reduzierten Umfang an Modellen zusätzlich unterstützt.

Charakteristisch für *Wisdom* sind laut Nunes und Cunha (2000b) folgende Punkte:

- Ein evolutionärer, benutzerorientierter Entwicklungsprozess, der sich auf die schnelle Entwicklung von Prototypen stützt, wodurch er prädestiniert für die Konstruktion von interaktiven Systemen ist.
- Eine Notation, die auf einer vereinfachten Teilmenge der UML aufbaut und die Modellierung von funktionalen und nicht-funktionalen Anforderungen unterstützt.
- Eine pragmatische Herangehensweise, für die ein flexibles Team und eine erhöhte Kommunikation zwischen Vertragspartnern, Benutzern und Entwicklern benötigt werden.

3.1.5 Formal Models for UI Design

Systemmodellierung in Z

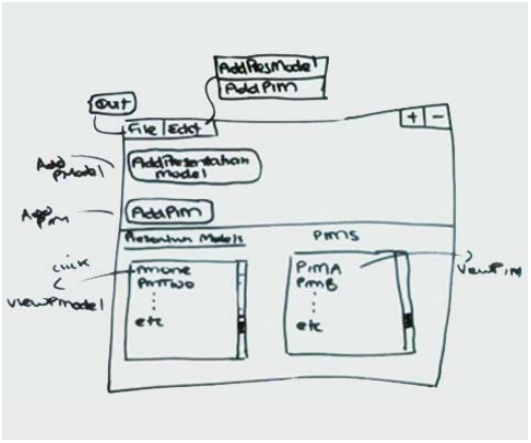
Eine mögliche Kombination von informalen und formalen Vorgehensweisen wird von Bowen und Reeves (2007) vorgeschlagen. Dabei werden die informalen Artefakte und Informationen, welche bei einem UCD-Prozess entstehen können, in formaler Weise spezifiziert. Der Entwurfsvorgang wird aufgeteilt, wodurch System und UI getrennt voneinander modelliert werden können. Für die Überführung von informalen Artefakten in formale Modelle werden *Presentation Model* (PM) (siehe - Abb. 3.6) und *Presentation and Interaction Model* (PIM) eingeführt. Bowen (2005) schlägt für die formale Modellierung von statischen und dynamischen Aspekten eines Systems die Sprache *Z* vor. Zusätzlich werden FSMs und die *µChart* Notation verwendet.

Informale werden in formale Modelle überführt

Das PM ist ein Modell, das die Bedeutung eines informalen Artefaktes festhält und für die formale Beschreibung des Designs verwendet werden kann. Dadurch ist es z.B. möglich, informale Prototypen in formaler Notation zu überführen. Das Verhalten des UIs wird in FSMs dargestellt. FSMs erlauben die Modellierung von dynamischen Sachverhalten und haben im Vergleich zu anderen formalen Methoden eine geringe Komplexität. Dadurch wird das Verhalten auch für Beteiligte ohne tiefergehende Kenntnisse nachvollziehbar und besser verständlich.

Verhalten und Desing des UI wird über PIMs festgehalten

Das PIM dient als Container für die gerade erwähnten PMs und FSMs und eignet sich deshalb zur Definition von Verhalten und Entwurf des UIs. Da aber auch PIMs begrenzt formal sind, schlägt sie die Überführung von beiden Notationen in die formalere Notation *µChart* (Reeve, 2005) vor. *µChart* verwendet zum Ausdrücken von Logik und Semantik ebenfalls die von *Z* bekannte Notation und ist darüber hinaus auch in der Lage, PIMs darzustellen. So können PIM und *Z* in *µChart* kombiniert und formal festgehalten werden, wodurch Verhalten und Struktur von System und UI dargestellt werden kann.




```
PModel
MainWin Widgetname
FileMenu Quit MenuItem EditMenu AddPMMenuItem
AddPIMMenuItem MaxWin MinWin AddPMButt
AddPIMButt PMList PIMList

Category
Container ActCtrl SValSel

Behaviour
   QuitAppUI_OpenAddPMDecs UI_OpenAddPIM
   UI_MaxWindow UI_MinWindow UI_OpenViewPM
   UI_OpenAddPMDecs UI_OpenAddPIM
   UI_OpenViewPM
MainWin is
   (FileMenu,Container, ())
     (QuitMenuItem, ActCtrl, (Quit App))
   (Edit Menu,Container, ())
     (AddPMMenuItem, ActCtrl, (UI_OpenAdd PMDecs))
     (AddPIMMenuItem, ActCtrl, (UI_OpenAddPIM))
     (MaxWin, ActCtrl, (UI_MaxWindow))
     (MinWin, ActCtrl, (UI_MinWindow))
     (AddPMButt, ActCtrl, (UI_OpenAddPMDecs))
     (AddPIMButt, ActCtrl, (UI_OpenAddPIM))
     (PMList, SValSel, (UI_OpenViewPM))
     (PIMList, SValSel, (UI_OpenViewPIM))
```

Abbildung 3.6: Entwurf eines UI als Skizze und in formale Notation als *Presentation Model* (Bowen und Reeves, 2007)

Dadurch wird es möglich, Artefakte und Informationen, die während des informalen UCD Designprozesses gesammelt werden, in formaler Weise festzuhalten und die herkömmlichen Vorgehensweisen für die Softwareentwicklung parallel zum Entwicklungsprozess für UIs durchzuführen. Bevor die Implementierung startet, werden die informalen Artefakte formalisiert und können in den formalen Entwicklungsprozess eingebunden werden. Die Trennung in System und UI kann dadurch überwunden werden und es ist gewährleistet, dass das System präzise und formal spezifiziert werden kann.

3.1.6 Usage-Centered Software Engineering

Usage-Centered Design Process

Ein weiteres Beispiel für einen leichtgewichtigen Ansatz für die Spezifikation von UIs wird von Constantine und Lockwood (1999a) vorgestellt. Diese Vorgehensweise hat deutliche Einflüsse aus dem UCD und steht damit im Gegensatz zu den bisher präsentierten Verfahren, die ihren Hintergrund in der klassischen Softwareentwicklung haben. Das Ziel des *Usage Centered Design Process* ist, die Anzahl und Komplexität der verwendeten Artefakte auf ein absolutes Minimum zu reduzieren.

3 Kernmodelle

Der Prozess für die Spezifikation des UI setzt sich aus *Role Models*, *Task Models* und *Content Models* zusammen. In diesen drei Kernmodellen werden die Informationen, die während der Anforderungsermittlung erhoben wurden, genauer spezifiziert. (siehe - Abb. 3.7).

Rollen anstatt Benutzer

User Roles werden verwendet, um die Eigenschaften, Bedürfnisse und Motivationen der Benutzer darstellen zu können. Es werden keine Benutzer modelliert, sondern die Rollen, die sie annehmen können. Dadurch wird kein umfassender Entwurf vorgenommen und die Modellierung auf die wichtigsten Aspekte eingeschränkt. Zudem kann dieser Vorgang mit hypothetischen Benutzern durchgeführt werden. Über

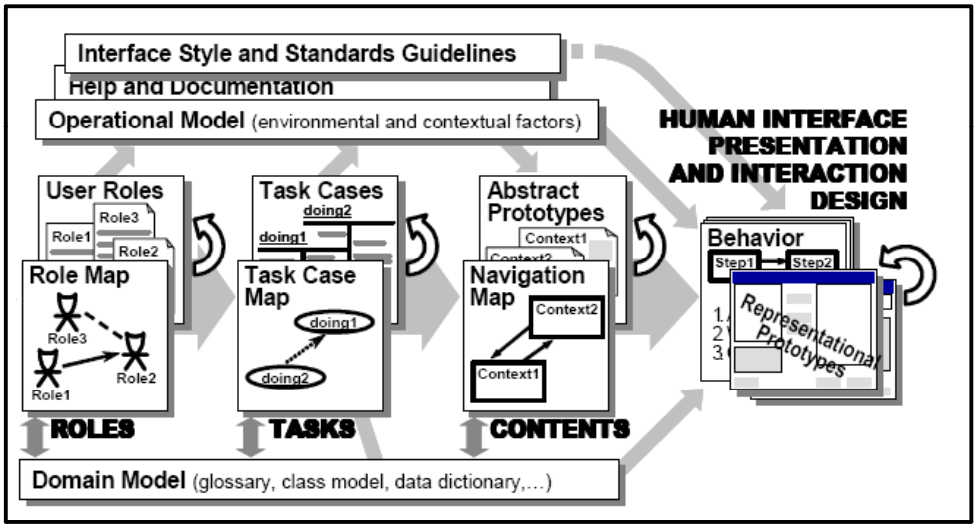


Abbildung 3.7: Der Usage-Centered Design Prozess nach Constantine et al. (2003)

Role Maps können die Zusammenhänge zwischen den einzelnen Rollen genauer analysiert werden. Anstatt Aktionen des Benutzers und die Reaktionen des Systems zu spezifizieren, werden *Task Models* verwendet, um die Aufgaben und die damit verbundenen Intentionen der Nutzer darzustellen. Dadurch wird die Modellierung abstrakter und lässt mehr Freiraum für den Entwurf zu. Die *Task Models* werden durch die bereits vorgestellten *Essential Use Cases* dargestellt und über *Use Case Maps* strukturiert. Die Architektur des UIs und der Aufbau der einzelnen Komponenten wird im *Content Model* spezifiziert. Dies geschieht sehr abstrakt und zunächst unabhängig von der späteren Realisierung mit *Canonical Abstract Protoypes* (siehe - Abb. 3.8). Diese Prototypen spielen eine wichtige Rolle, da sie die Verbindung zwischen den *Task Models* und den detaillierten Prototypen darstellen (Constantine, 2003).

Canonical Abstract Prototypes

Für die Entwicklung eines kompletten Systems können diese drei Kernmodelle erweitert werden. So schlagen Constantine und Campos (2005) für die Modellierung der funktionalen Aspekte die Verwendung von *System Actors* und *System Use Cases* vor. *System Actors* sind keine wirklichen Benutzer, sondern interne oder externe Objekte, die über *System Use Cases* auf Funktionen zugreifen. Aus diesen beiden Modellen können Klassen- und Sequenzdiagramme entwickelt werden, welche dann zusammen mit den detaillierten Prototypen in die Spezifikation des Systems einfließen. Geschäftsregeln oder betrieblichen Vorgaben können über *Business Rule Models* eingebunden werden, Datenflüsse innerhalb des Systems sind mit *Data Models* darstellbar.

Flexibler erweiterbarer Prozess

Da es sich beim *Usage-Centered Design* um einen modellgetriebenen Prozess handelt, ist er für die reine Spezifikation von UIs weniger geeignet. Modellgetriebene Ansätze, bei denen eine automatische Erstellung des UIs aus den Modellen erfolgt, sind nur bedingt dazu geeignet, Applikationen mit innovativen Interaktionsformen zu erstellen. Da die Umsetzung der einzelnen Modelle durch die vorhandenen Vorlagen begrenzt wird, können oft nur herkömmliche, *WIMP (Windows, Icons, Menus and Pointing device)* basierende Anwendungen erstellt werden.

Modellgetriebenen Prozess

Modelle mit geringerer Komplexität

Zudem ist die iterative Verfeinerung vor allem für die Modelle und nicht für das Design vorgesehen (Constantine und Lockwood, 2003). Allerdings verfolgen sie einen interessanten Ansatz, da der Modellumfang auf die wesentlichen Artefakte reduziert wird, wodurch sich die Komplexität verringert. Dadurch wird die Modellierung auch für Beteiligte nachvollziehbar. Zudem ist diese Vorgehensweise sehr flexibel an die vorhandenen Gegebenheiten anpassbar. Der Prozess kann deshalb sehr strukturiert, mit vollem Modellumfang, aber auch in agiler Form mit minimaler Modellpalette verwendet werden (Constantine, 2001), (Constantine und Lockwood, 2002), (Constantine und Lockwood, 2003).

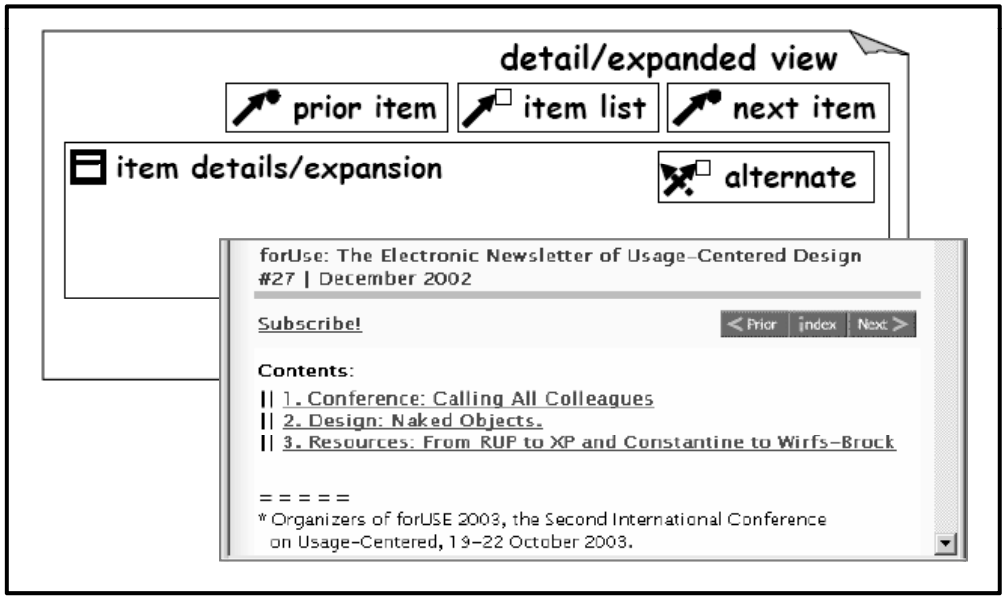


Abbildung 3.8: Canonical Abstract Protoype (hinten links) und automatisch generierter Prototyp (vorne rechts) (Constantine, 2003)

3.1.7 Bewertung der Ansätze

Bewertung der Spezifikationsansätze

Funktionale Anforderungen werden von allen Herangehensweisen unterstützt, da sie aus dem Umfeld des SE stammen. Es wird versucht, die vorhandenen Notationen mit der Sichtweise des HCI zu kombinieren, um die Modellierung des UIs zu ermöglichen. Geschäftsprozesse können mit den selben Notationen erstellt werden, die auch in der SE Verwendung finden (Oestereich, 2005), (Holt, 2005). Um festzustellen, ob die vorgestellten Vorgehensweisen für eine interdisziplinäre Erstellung von Spezifikationen tatsächlich geeignet sind wurde eine Bewertung durchgeführt (siehe Tab. 3.1). Dabei wurden die Ziele dieser Arbeit als Kriterien herangezogen:

- Wie gut wird gemeinschaftliches Arbeiten unterstützt. Es ist wichtig, dass einfache und nachvollziehbare Modelle Verwendung finden, sodass ein gemeinsames Verständnis erzeugt wird und interdisziplinäres Arbeiten möglich ist.
- Wie gut wird kreatives Arbeiten unterstützt. Es muss zu Beginn der Entwicklung möglich sein, mehrere Entwürfe in schneller Abfolge zu erzeugen, sodass innovative Ansätze nicht unentdeckt bleiben.

- Wie flexibel ist Herangehensweise und wie schnell können Ergebnisse erzielt werden.
- Wie gut lassen sich Anforderungen nach der Spezifikation identifizieren.
- Die Simulation des UI vermittelt das *Look & Feel* der Oberfläche. Verhalten und Aussehen entsprechen dabei dem erwünschten System.

Gemeinschaftliche Kooperation

Ein wichtiges Kriterium ist, ob es allen Beteiligten ermöglich wird, am Spezifikationsprozess teilzunehmen und ob die verwendeten Modelle von allen anwendbar und verständlich sind. Dies ist nur der Fall, wenn alle Akteure die Bedeutung der verwendeten Symbole verstehen und die Modellierung mit ihnen nachvollziehen können (Bowen et al., 1995). Nur so ist es möglich, dass die Entscheidungsfindung unterstützt wird und dass die Ergebnisse als Referenzmaterial für die spätere Umsetzung geeignet sind. Laut Barbosa et al. (2004) sind dazu am besten Vorgehensweisen geeignet, die einen geringen Umfang an Modellen beinhalten. So können lediglich das *Usage-Centered Design* und mit Abstrichen *Wisdom* bei diesem Punkt überzeugen, da sie eine reduzierte Modellpalette und einfache, verständliche Modelle verwenden.

Kriterien	UML for interactive Systems	UMLi	Wisdom	Formal Models for UI Design	Usage Centered Design
Kooperations-fähigkeit	○	○	◒	○	●
Kreativität	◒	◒	◒	◒	◒
Agilität	○	○	◒	○	●
Verfolgbarkeit von Anforderungen	○	○	○	○	○
UI-Simulation	○	○	○	○	○

○ ausreichend bzw. befriedigend ◒ gut ● exzellent

Tabelle 3.1: Eignung der vorgestellten Vorgehensweisen für eine interdisziplinäre Spezifikation

Kreativität

Bei der Unterstützung von kreativen Vorgängen schneiden alle Vorgehensweisen ähnlich ab. Informale Techniken, die Mehrdeutigkeit und Kreativität unterstützen, werden in den frühen Phasen nur bei den *Formal Models for UI Design* verwendet. Um ein gutes UI erstellen zu können, werden künstlerische Fähigkeiten benötigt (Phanouriou, 2000), die durch geeignete Spezifikations- oder Modellierungsformen unterstützt werden müssen. Durch die Bereitstellung von konventionellen Herangehensweisen ist es deshalb nicht möglich, neue Wege und Möglichkeiten der Interaktion zu erschließen und einen „Tunnelblick“ zu vermeiden.

Agilität

Starre Prozesse, die sehr stark formalisiert sind, eignen sich weniger für die gemeinsame Spezifikation. Besser geeignet sind leichgewichtige Herangehensweisen wie das *Usage-Centered Design*, das an die Bedürfnisse der aktuellen Entwicklung angepasst werden kann. Zudem basiert dieser Ansatz auf einer reduzierten Anzahl an

verwendeten Modellen, weshalb sich Beteiligte aus allen Disziplinen schneller in die Modellierung einarbeiten können und Ergebnisse früher vorhanden sind.

Traceability

Um Abhängigkeiten zwischen Anforderungen und UI erkennen zu können, ist es wichtig, dass eine Beziehung unter den erhobenen Anforderungen hergestellt werden kann. Zudem müssen Anforderungen auch nach der Umsetzung erkannt und identifiziert werden können, sodass Entscheidungen nachvollziehbar bleiben. Diekte Beziehungen zwischen den Anforderungen werden kaum von keiner Vorgehensweise unterstützt, weshalb es schwierig ist Gründe für Entscheidungen nachzuvollziehen.

Simulation des UIs wird nicht unterstützt

Abstrakte und detaillierte Darstellungen werden von allen Vorgehensweisen unterstützt, allerdings sieht kein Ansatz die Simulation des UIs vor. Es ist zwar möglich interaktives Verhalten formal zu modellieren und auch das Aussehen über Prototypen festzulegen, aber ein detailliertes *Look & Feel* kann dadurch nicht hergestellt werden. Laut Carr (1996) ist die wichtigste Eigenschaft einer Spezifikation, den Entwurf gegenüber anderen zu kommunizieren. Dies kann mit formalen Modellierungsformen kaum erreicht werden, da diese nur bedingt für interdisziplinäre Teams mit unterschiedlichen Kenntnissen, Fähigkeiten und Wissenständen geeignet sind (Dix, 1995b). Besser kann dies durch eine ausführbare interaktive Spezifikation, z.B. durch animierte Prototypen, realisiert werden. Anstatt in Spezifikationsdokumenten nachschlagen zu müssen, ist es bei Unklarheiten möglich, den Prototyp auszuführen, um Fragen zu Darstellung oder Verhalten klären zu können (Rudd et al., 1996).

Wichtige Eigenschaften fehlen

Als Ergebnis lässt sich festhalten, dass die vorgestellten Vorgehensweisen alle für die Modellierung eines UIs geeignet sind. Es fehlen aber wichtige Eigenschaften, die es auch Nichtfachleuten ermöglichen am Spezifikationsprozess teilzunehmen. Zudem ist problematisch, dass auch weiterhin Spezifikationsdokumente erstellt werden müssen, um die Ergebnisse der Modellierung für alle *Stakeholder* verständlich zu machen. Die Spezifiaktion bleibt also textbasiert und es ist schwierig einzelne Anforderungen oder Beweggründe zu identifizieren (siehe Kapitel 4.1.1—"Textbasierte Spezifikationen").

3.2 Werkzeugunterstützung

Bewertung der Werkzeug-unterstützung

Neben der Betrachtung von Methoden zur Spezifikation ist es darüber hinaus auch wichtig zu untersuchen, welche Werkzeuge für den Entwurf von UIs vorhanden sind und wie gut diese die einzelnen Phasen bei der Erstellung unterstützen (Bock, 2007). Da es heutzutage eine Vielzahl an Hilfsmitteln für die Entwicklung von UIs gibt (SIGGRAPH, 1987), (Myers, 1995), (Memmel et al., 2007c), (Memmel und Reiterer, 2008b) wurde eine Kategorisierung in fünf Gruppen vorgenommen. Diese Einteilung basiert auf den Feststellungen von Myers (1995) sowie Campos und Nunes (2007). Anschließend wurden die Eignung und die Möglichkeiten dieser Werkzeuge bewertet (siehe Tab. 3.2). Die aufgestellten Kriterien entsprechen dabei wichtigen Funktionalitäten, die bei einem interdisziplinären Entwurf von UIs notwendig sind (vgl. Geyer (p. 25 2008, p. 53ff)). Für einen detaillierteren Überblick wird auf die Ergebnisse von König (2008) verwiesen.

Kategorisierung / Kriterien	Grafische Spezifikation Axure , Irise	Visualisierungs-werkzeuge MS Visio	CASE Tools Rational Rose	Bildbearbeitungs-programme Adobe Photohop	Entwicklungs-werkzeuge MS Expression Blend, MS Visual Studio
Design und Kreativität	◒	◒	○	●	◒
Modellierung und Konzeption	○	●	●	○	○
Prototyping und Simulation	●	○	○	○	◒
Präsentations- und Exportfähigkeit	●	◒	○	◒	○
Spezifikations- und Mitteilungsfähigkeit	●	○	○	○	◒
Kollaborations-fähigkeit	◒	○	○	○	◒
Verfolgbarkeit von Anforderungen	○	○	◒	○	○

○ ausreichend bzw. befriedigend ◒ gut ● exzellent

Tabelle 3.2: Bewertung von aktuellen UI Entwicklungswerkzeugen basierend auf Geyer (2008) und König (2008)

Kreativität wird eingeschränkt

Traditionelle Entwicklungsumgebungen wie z. B. MS Visual Studio[1] erlauben die Entwicklung von UIs über einen *GUI-Builder*. Da die Erzeugung von Programmcode im Vordergrund steht, sind die Gestaltungsmöglichkeiten und kreatives Arbeiten eingeschränkt. Durch die Verbindung mit XML-basierten Beschreibungssprachen wie etwa der *Extensible Application Markup Language* (XAML) oder *USer Interface eXtensible Markup Language* (UsiXML) und die Verwendung von entsprechenden Entwicklungsumgebungen (z. B. MS Expression Blend[2]) hat sich dies verbessert. Die Vorteile von Bildbearbeitungssoftware wie z. B. Adobe Photoshop[3] liegen eindeutig bei der Erzeugung von Grafiken, allerdings schneiden sie bei allen anderen Kriterien schlecht ab.

Schwierig zu bedienen

CASE-Tools wie etwa Rational Rose[4] erlauben die Erzeugung von UIs direkt aus den zuvor erstellten Modellen. Sie sind deshalb sehr gut geeignet für die Modellierung und Konzeption von UML-Diagrammen, aber nicht für die adäquate Unterstützung von Entwurfsprozessen (Memmel et al., 2007e), (Memmel und Reiterer, 2008b). Kreatives Arbeiten ist bei modellgetriebenen Vorgehensweisen und Werkzeugen nur bedingt möglich, weil die Erzeugung von UIs auf vorhandene *Widgets* beschränkt ist (Damm et al., 2000), (Mackay et al., 2003). Dasselbe gilt auch bei der Benutzung von *GUI-Buildern* für die Erstellung von Oberflächen (Lecolinet, 1998). Wie Kemerer (1992), Iivari (1996) sowie Chervany und Lending (1998) zeigen, sind diese Werkzeuge nicht sehr verbreitet und auch wenn sie vorhanden sind, werden sie nur in geringem Maße eingesetzt. Dies liegt unter anderem auch daran, dass diese Software kompliziert und schwer zu bedienen ist, wie Jarzabek und Huang (1998) festgestellt haben.

[1] http://www.microsoft.com/germany/msdn/vstools/
[2] http://www.microsoft.com/expression/
[3] http://www.adobe.com/products/photoshop/
[4] http://www-01.ibm.com/software/de/rational/design.html

Visualisierungswerkzeuge haben ihre Stärken ebenfalls bei der Unterstützung von Modellierungstätigkeiten und versetzen den Anwender darüber hinaus auch in die Lage, UIs zu entwerfen. Allerdings ist es nicht möglich, interaktive Prototypen zu erzeugen und die Möglichkeiten zur Modellierung sind begrenzt.

> „ [...] the interactivity supported by multimedia design tools is very limited, usually to only basic mouse clicks, and so is support for creating functionality and tying in data."
>
> (Petrie und Schneider, 2006)

Möglichkeit zur Modellierung ist eingeschränkt

Die besten Ergebnisse liefern die Werkzeuge für die Erstellung von interaktiven, grafischen Spezifikationen wie etwa Irise[5] und Axure[6]. Sie unterstützen viele der erforderlichen Funktionen wie z.B. die Simulation des UIs, wodurch die Mitteilungsfähigkeit der Spezifikation eine ganz neue Qualität erreicht. Zudem sind sie einfach zu bedienen, sodass UI-Prototypen schnell erzeugt werden können (Memmel et al., 2007c). Allerdings werden Modellierungsmöglichkeiten nur sehr eingeschränkt unterstützt und auch die Nachverfolgbarkeit von Anforderungen ist nicht besonders gut gelöst.

Werkzeugketten sind notwendig

Bei diesem Vergleich fällt auf, dass kein Werkzeug in allen Punkten überzeugen kann. Somit ist es notwendig, Werkzeugketten für die Spezifikation von UIs zu bilden, was für die Erstellung einer interaktiven Spezifikation nicht förderlich ist. Deshalb ist es wichtig, dass Werkzeuge und Modellierungsformen aufeinander abgestimmt sind, da eine direkte Abhängigkeit besteht. Ansonsten drohen Medienbrüche, ein Mix an Modellen und Verständigungsproblemen unter den Beteiligten.

3.3 Zusammenfassung

Vorgehensweisen nicht besonders gut geeignet

In diesem Kapitel wurden Ansätze gezeigt, die für die Modellierung und Spezifikation von UIs geeignet sind. Anschließend wurden diese Vorgehensweisen auf ihre Eignung für interdisziplinäres Spezifizieren untersucht. Da der Fokus beim Entwurf dieser Vorgehensweisen auf andere Prioritäten gelegt wurde, sind diese dazu nicht optimal geeignet. Zwar kann das interaktive Verhalten des UIs modelliert werden, allerdings ist es nicht möglich, ein detailliertes *Look & Feel* darzustellen. Zudem ergibt sich weiterhin ein Medienbrüche, wodurch Nachvollziehbarkeit und Verständnis von Entscheidungen für *Stakeholder* behindert wird. Außerdem wird deutlich, dass diese Verfahren versuchen, etablierte Techniken aus den Disziplinen HCI und SE zu kombinieren, weil System und UI sehr stark voneinander abhängig sind.

Fehlende Werkzeugunterstützung

Abgesehen von der Betrachtung der vorhandenen Vorgehensweisen ist es auch wichtig zu sehen, ob es geeignete Werkzeuge gibt, die es erlauben, UIs zu spezifizieren. Die Werkzeuge für die grafische Spezifikation (z. B. Axure und Irise) überzeugen in vielen Punkten, unterstützen jedoch keine Modellierung. Somit sind

[5] http://www.irise.com
[6] http://www.axure.com

Entwicklerteams beim Spezifizieren von UIs weiterhin auf Werkzeugketten angewiesen. Wenig geeignete Vorgehensweisen und fehlende Werkzeugunterstützung legitimieren deshalb unseren Forschungsansatz.

> „[...] to address concerns from other disciplines such as HCI during design and development, we need to look for alternative solutions instead of focusing on extending UML to incorporate elements of HCI design."
>
> (Barbosa et al., 2004)

Kapitel 4

Entwurfs- und Spezifikationsmöglichkeiten

In diesem Kapitel werden Möglichkeiten zur Spezifikation von Anforderungen vorgestellt. Dabei wird beleuchtet, welche Vor- und Nachteile vorhanden sind und ob sie für die interdisziplinäre Spezifikation eingesetzt werden können. Im Anschluß werden Möglichkeiten und Techniken vorgestellt, durch die bisherige Spezifikationsformen ergänzt werden können, sodass die Anforderungsanalyse und Spezifikation für UIs interdisziplinär durchgeführt werden kann.

4.1 Spezifikation von Anforderungen

Interdisziplinärer Vorgang und kooperativer Lernprozess

Bei der Anforderungsermittlung handelt es sich um einen interdisziplinären Vorgang, bei dem es auf die Bedürfnisse von Personen und deren unterschiedliche Blickwinkel auf die Problemstellung ankommt (Hofmann und Lehner, 2001). Haumer (2000) bezeichnet die Anforderungsermittlung als einen kooperativen Lernprozess, bei dem es wichtig ist, dass alle, die in dieses Projekt involviert oder davon betroffen sind, ihren Standpunkt darlegen können und aktiv mitwirken. Viele Entwicklungsvorgänge schlagen fehl, weil dies nicht beachtet wird und der Fokus nicht auf dem Anwendungsbereich und den involvierten Akteuren, sondern hauptsächlich auf technischen Belangen liegt (Holtzblatt und Beyer, 1995).

> [Requirement Engineering is, d.V.] „a systematic process of developing requirements through an iterative cooperative process of analysing the problem, documenting the resulting observations in a variety of representation formats, and checking the accuracy of the understanding gained."
>
> (Loucopoulos und Karakostas, 1995, p. 13)

Kooperation und Kollaboration

Da Anforderungen inkompatibel, überholt, ungenau und widersprüchlich sein können, ist eine genaue Untersuchung durch das Projektteam nötig (Minocha, 1999). Nur wenn eine gute Kooperation und Kollaboration unter den Beteiligten herrscht,

ist es möglich, die richtigen Entscheidungen zu treffen (Hofmann und Lehner, 2001). Können sich alle auf eine Anforderung einigen, wird sie übernommen, ansonsten verworfen oder nach Alternativen gesucht.

Textbasierte Spezifikation

Weil Anforderungen meistens nur implizit als Skizze oder als Gesprächsprotokoll vorliegen, werden sie in einem weiteren Schritt in eine genaue Spezifikation überführt und dokumentiert (Sommerville, 2005). Dies findet üblicherweise durch textuelle Beschreibungen statt, es fließen aber auch Modelle und Prototypen mit in die Spezifikation der Anforderungen ein. Dabei muss besonders beachtet werden, dass Ausdrucksweisen gewählt werden, welche möglichst von allen *Stakeholdern* verstanden werden. Sie machen eine anschließende Prüfung und Bewertung durch die Akteure erst möglich und erlauben eine weitere Verfeinerung.

Laufend duchgeführte Validierung und Verifizierung

Die Validierung und Verifizierung findet während des gesamten Prozesses statt und gewährleistet, dass das Spezifikationsdokument vollständig und korrekt bleibt. Wichtig ist, dass die Ergebnisse und Entscheidungen, die getroffen werden, ebenfalls festgehalten werden, um die Nachvollziehbarkeit von Entscheidungen sicherzustellen. Nach Boehm et al. (1984) befasst sich die Validierung mit der Fragestellung, ob das richtige System umgesetzt wird, während die Verifizierung dafür zuständig ist, dass das System richtig realisiert wird. Nach Hofmann und Lehner (2001) haben erfolgreiche Projektteams im Durchschnitt drei Prozessdurchläufe nötig um nützliche Resultate zu erzielen. Problematisch ist dabei, dass Anforderungen sich während der Entwicklung ändern können, was während der Umsetzung berücksichtigt werden muss.

Anforderungen müssen dokumentiert werden

Anforderungen sind definiert als dokumentierte Repräsentationen einer Bedingung oder einer Eigenschaft, die von einem System benötigt werden, um eine Problemstellung zu lösen (siehe Kapitel 2.4—"Disziplinenübergreifende Zusammenarbeit bei der Anforderungsermittlung"). Die Notwendigkeit der Dokumentation ist also bereits in der Definition einer Anforderung berücksichtigt. Daraus folgt, dass alle gesammelten Informationen, die während der Anforderungsermittlung gesammelt werden, erst durch die Dokumentation und Spezifikation zu Anforderungen werden.

Die Dokumentation von Anforderungen muss nach Pohl (2007) folgende Ansprüche erfüllen:

- Vollständigkeit - Alle Anforderungen müssen erfasst sein.
- Konsistenz - Anforderungen müssen widerspruchsfrei und ohne Konflikte sein.
- Änderungsfreundlichkeit - Änderungen und Ergänzungen können einfach durchgeführt werden.
- Nachvollziehbarkeit - Alle Anforderungen müssen verständlich dokumentiert werden.

Dokumentation und Spezifikation der Anforderungen

Eine Spezifikation ist das primäre Ergebnis der Anforderungsanalyse. In ihr sind alle wichtigen Informationen und Anforderungen in dauerhafter Form aufgelistet, die später vom umgesetzten System erfüllt werden müssen. Dies stellt sicher, dass alle Anforderungen der Interessengruppen dokumentiert und berücksichtigt werden. Idealerweise versetzt eine Spezifikation alle Akteure in die Lage, die Zusammenhänge schnell zu verstehen und die Bedürfnisse der Interessengruppen nachvollziehen zu können (Hofmann und Lehner, 2001). Außerdem wird die Kommunikation zwischen Beteiligten gefördert, da Anforderungen reflektiert und gegebenenfalls in Frage gestellt werden.

Wissensbasis für die Umsetzung und Vertragsgrundlage

Eine Spezifikation dient als Wissensbasis, um das bisher Umgesetzte schon während der Entwicklung verifizieren zu können. Auch für die Anforderungsanalyse von Nachfolgesystemen kann die Spezifikation des bestehenden Systems hilfreich sein. Daneben hat das Spezifikationsdokument als Vertragsgrundlage zwischen Auftraggeber (engl. *Client*) und Auftragnehmer (engl. *Supplier*) eine weitere wichtige Funktion.

Textbasierte und informale Darstellung

Um die Spezifikation für alle beteiligten Akteure im Entwicklungsprozess leicht zugänglich zu machen, werden Spezifikationen üblicherweise durch Text ausgedrückt (Bock, 2007). Zudem können Änderungen leicht eingepflegt werden, da keine weiteren Voraussetzungen erfüllt werden müssen, um Anforderungen lesen oder schreiben zu können. Welche zusätzlichen Artefakte Spezifikationen enthalten können, ist nicht genau festgelegt oder standardisiert. Dies können z.B. Grafiken, Prototypen und Modelle sein. Um diese Artefakte zu erstellen, werden neben Werkzeugen für die Textverarbeitung auch Bildbearbeitungssoftware, Animationswerkzeuge und Modellierungswerkzeuge verwendet (Geiger und Müller, 1998), (Carr, 1996), (McInerney und Sobiesiak, 2000), (Memmel et al., 2007a). Beispiele für textbasierte Vorlagen und Anweisungen zum Verfassen von Spezifikationsdokumenten für den englischsprachigen Raum sind z.B. das *System Requirement Specification Document* (IEEE, 1998a) bzw. *Software Requirement Specification Document* (IEEE, 1998b) oder das *Volere Requirements Specification Template*[1] (Robertson und Robertson, 1999).

Vorlagen für Spezifikationen

Lasten- und Pflichtenheft

Im deutschsprachigen Raum besteht die Spezifikation aus einem Lastenheft und einem Pflichtenheft (Geiger und Müller, 1998). Im unverbindlicheren Lastenheft werden die Forderungen an die Lieferungen und Leistungen des Auftragnehmers gegenüber dem Auftraggeber bzw. Kunden beschrieben. Das danach folgende Pflichtenheft, welches vom Auftragnehmer erstellt wird, konkretisiert die Anforderungen und enthält bereits detaillierte Realisierungsvorgaben.

4.1.1 Textbasierte Spezifikationen

Formalen Spezifikationsformen ungeeignet

Markopoulos und Marijnissen (2000) legen dar, dass es keine gemeinhin akzeptierte *"best practice"* zur Durchführung einer Spezifikation gibt. Es gibt zwar etliche strukturierte Methoden, welche für die Spezifikation von UIs verwendet werden können, wie Carr (1996) feststellt, bisher konnte sich aber keine in der Praxis durchsetzen (vlg. Kapitel 3—"Ansätze zur Modellierung von Benutzerschnittstellen"). Dies liegt seiner Meinung nach vor allem daran, dass diese formalen Spezifikationsformen für die Beteiligten schwierig zu lesen, zu erstellen und zu verstehen sind.

[1] http://www.systemsguild.com/GuildSite/Robs/Template.html

Textbasierte Spezifikation

Weil es bei der Verwendung von formalen textbasierten Spezifikationssprachen (wie z.B. Z) zu Kommunikationsproblemen kommt (Frank, 1995), werden stattdessen textbasierte Spezifikationen auf natürlichsprachiger Basis erstellt (Geiger und Müller, 1998), (Mayhew, 1999), (McInerney und Sobiesiak, 2000), (Pyla et al., 2004), (Memmel et al., 2007a), (Bock, 2007). Die Vorteile sind offensichtlich: Texte können komplexes Verhalten beschreiben und es ist möglich, Informationen in einem einfach lesbaren, druckbaren, änderbaren und austauschbaren Format festzuhalten (Trætteberg, 2002a). Zur näheren Erläuterung werden Ergänzungen durch Grafiken, Tabellen oder Diagramme vorgenommen (Geiger und Müller, 1998), (Zhu und Gorton, 2007).

Interpretationsspielraum

Problematisch ist allerdings, dass Anforderungen, die durch Text festgehalten werden, mehrdeutig sind und falsch verstanden werden können, da es erheblichen Interpretationsspielraum gibt (Memmel et al., 2007a). Text ist kein adäquates Mittel, um Anforderungen der Benutzer zu spezifizieren (Rashid et al., 2006), vor allem nicht in natürlichsprachiger Ausführung (Horrocks, 1999). Die Kommunikationsfähigkeit wird eingeschränkt, da die einzelnen Interessengruppen verschiedene "Sprachen" sprechen und Fachausdrücke unterschiedliche Bedeutungen haben können (Moore, 2003). Daneben ist es schwierig das *Look & Feel* des UIs darzustellen, da das Aussehen zwar durch Abbildungen dargestellt werden kann, das gewünschte interaktive Verhalten jedoch nur schwer aus den abstrakten Beschreibungen entnommen werden kann.

Kein standardisiertes Vorgehen und Medienbrüche

Ein weiteres Problem ergibt sich aus den verwendeten Werkzeugen. Wie Bock (2007) in einer Studie zeigt, wird beim Erstellen dieser informalen, textbasierten Dokumente überwiegend gewöhnliche Bürosoftware eingesetzt. Dass die Anwender auf Standardsoftware zurückgreifen, lasst sich darauf zurückführen, dass diese Werkzeuge weit verbreitet sind, sie sich mit diesen Werkzeugen auskennen und dass Artefakte relativ einfach mit ihnen erstellt und ausgetauscht werden können. Somit werden Anforderungen zwar erhoben und festgehalten, die Beziehungen zwischen Anforderungen bleiben aber unklar und sind schwierig zu verwalten, da ein koordiniertes und standardisiertes Vorgehen fehlt (Memmel et al., 2008b). Das Ergebnis dieser verketteten Werkzeuge sind unterschiedliche Formate und eine erschwerte Zusammenarbeit, da nicht alle Beteiligten dieselben Anwendungen favorisieren (Memmel et al., 2007a).

Anforderungen nicht nachvollziehbar

Da die Entwickler andere Modelle und Werkzeuge verwenden, müssen die Beschreibungen umgewandelt werden. Dies bleibt ein schwieriges Unterfangen, weil Präzision und Exaktheit der festgehaltenen Sachverhalte unter der Umwandlung leiden, auch wenn es Möglichkeiten zur Konvertierung dieser oftmals proprietären Formate gibt (Bock, 2007), (Ambler, 2003). Dadurch sind Anforderungen nicht transparent und nachvollziehbar, wodurch die Realisierung des geforderten UIs behindert wird (Memmel und Reiterer, 2008c).

Traceability und Transparenz fehlen

Die Auswirkungen von Designentscheidungen können nur mit Verzögerung festgestellt werden, weil die Möglichkeit zum schnellen Wechsel zwischen Anforderungen und Entwurf nicht gegeben ist. Dies begünstigt Inkonsistenzen und Widersprüchlichkeiten, da Anforderungen nicht identifiziert werden können, wodurch Änderungen und Aktualisierungen schwer durchführbar werden. Zudem können Kontroversen oder Fragen zur Umsetzung nur schwer legitimiert werden, wenn Anforderungen nicht ersichtlich sind.

Erhöhte Kosten und fehlerträchtige Systeme

Mehrdeutigkeiten bei textuellen Spezifikationen und Medienbrüche begünstigen, dass Widersprüche erst während der Umsetzung erkannt werden, wodurch es zu kostenintensiven Nachbesserungen und vielfachen Iterationen während der Umsetzung kommt (Memmel et al., 2007a), (Memmel und Reiterer, 2008a). Im schlimmsten Fall kommt es dann zu dem von Offergeld und Oed (2006) beschriebenen Szenario, bei welchem der Auftragnehmer nicht in der Lage ist, ein benutzerfreundliches System nach den exakten Vorstellungen des Auftraggebers zu gestalten.

4.1.2 Modellbasierte Spezifikation

Reduktion auf Relevantes

Um komplexe Problemstellungen lösen zu können, wird auf die Technik des Modellierens zurückgegriffen. Dadurch wird es möglich, existierende Begebenheiten aus einem Bezugssystem auf eine abstraktere Modellebene (Damm et al., 2000) abzubilden. Es werden wesentliche Merkmale und relevante Attribute hervorgehoben, wohingegen bestimmte Eigenschaften des Bezugssystems bewusst vernachlässigt werden (Jacobson et al., 1999, p. 22). Dadurch kann der Überblick erhalten werden und es ist nicht nötig, sich mit Details zu beschäftigen (Paternò und Volpe, 2005).

Interpretation fördert das Verständis

Diese Verallgemeinerung fördert den Reflexionsprozess, da Interpretationen nötig sind, um Modelle von der Modellierungsebene zurück ins Bezugssystem überführen zu können (siehe Abb. 4.1). Dadurch wird allen Beteiligten ein besseres Verständnis und eine bessere Verständigung ermöglicht (Damm et al., 2000), (Osterweil, 2007).

Komplexe Problemstellungen werden lösbar und die Erstellung von innovativen Lösungsansätzen wird begünstigt. Modelle können für die Analyse, Spezifikation und Dokumentation von interaktiven Systemen verwendet werden, wobei der Modellierungsvorgang meistens iterativ durchgeführt wird.

Rumbaugh et al. (1998) führt folgende Vorteile auf, die die Nutzung von Modellen mit sich bringt:

- Die Bewertung von mehreren verschiedenen Entwürfen ist relativ einfach und noch vor der eigentlichen Umsetzung möglich.
- Modelle ermöglichen die Aufteilung einer Problemstellung in mehrere Bereiche, die unabhängig voneinander untersucht werden können.
- Die Herstellung von funktionsfähigen und benutzbaren Produkten wird gefördert.
- Verschiedene Blickwinkel auf eine Problemstellung werden ermöglicht, wodurch Informationen gesammelt, untersucht, gefiltert, organisiert und bearbeitet werden können.
- Es wird ermöglicht, domänenspezifisches Wissen und Anforderungen präzise festzuhalten, sodass das Verständnis und die Verständigung von Beteiligten sich verbessern.
- Komplexe Systeme können durch abstrakte und einfachere Modelle dargestellt werden. So können mögliche Auswirkungen und Abhängigkeiten bereits im Vorfeld erkannt werden.

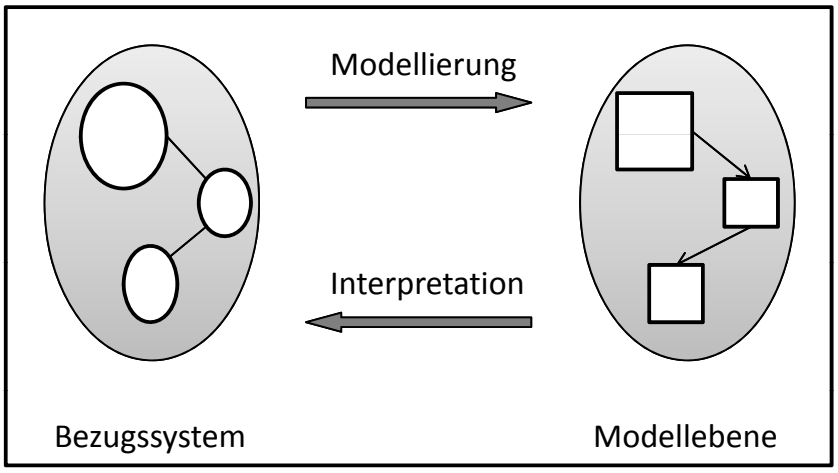


Abbildung 4.1: Zusammenhang von Modellierung und Interpretation (Damm et al., 2000).

> „We use models in our work precisely because they actually speed up development and help us get to a superior solution more quickly. Good models clarify design issues and highlight tradeoffs, so decisions can be resolved rapidly. Perhaps most importantly, models help us to know what to build., which really speeds things up because any time spent building the wrong system or creating unnecessary features is time completely wasted."
>
> (Constantine, 2000)

Unterschiedliche Abstraktionsgrade

Modelle haben sowohl deskriptiven als auch präskriptiven Charakter. Im ersten Fall bilden sie etwas nach, im zweiten Fall dienen sie als Vorbild. Üblicherweise werden Modelle mit unterschiedlichen Abstraktionsgraden bei der Softwareentwicklung verwendet (Nunes und Cunha, 2000b). Durch die Verwendung von einfachen abstrakteren Modellen wird die Sondierung von unterschiedlichen Entwurfsvarianten und Ideen in frühen Phasen möglich. In den späteren Phasen werden detaillierte und präzisere Modelle verwendet, um eine genaue Beschreibung durchführen zu können.

Erhöhung der Kommunikation

Ein weiterer Vorteil bei der Verwendung von Modellen ist, dass sie nicht nur zur Verbesserung der Kommunikation beitragen, sondern auch zur Dokumentation von Ergebnissen eingesetzt werden können. Wie Constantine (2000) berichtet, wird selbst unter höchstem Termindruck nicht auf die Modellierung verzichtet, weil sich dadurch Zeit sparen lässt.

> „Using models as part of user interface development can help capture user requirements, avoid premature commitment to specific layouts and widgets, and make the relationships between an interface's different parts and their roles explicit."
>
> (da Silva und Paton, 2003)

4.1.3 Formale Ausdrucksweisen

Abstraktion und Präzision

Wie im vorherigen Kapitel 3.1—"Brückenschläge zwischen den Disziplinen" gezeigt wurde, ist es grundsätzlich möglich, UIs mit formalen Methoden zu modellieren. Durch die Verwendung von Modellen wird auf Einzelheiten verzichtet, wodurch es möglich wird, zu erkennen "Was" umgesetzt werden soll, anstatt sich darauf zu konzentrieren "Wie" dies geschieht (Khazaei und Roast, 2001). Doch neben der Möglichkeit zur Abstraktion haben formale Methoden noch einen weiteren Vorteil. Durch die Verwendung von genau definierter Syntax und Semantik ist es möglich, Sachverhalte und funktionalen Aspekte eindeutig zu überprüfen. Die Präzision und Sorgfalt, die nötig ist, um eine formale Spezifikation zu erstellen, unterstützt das Auffinden von Widersprüchen und Mehrdeutigkeiten in interaktiven Systemen (Fields et al., 1994), (Duke und Harrison, 1995). Funktionale Schwächen können dadurch bereits vor der Umsetzung erkannt und beseitigt werden. Darüber hinaus wird die Analyse und Umsetzung von effektiver, effizienter und zuverlässiger Software gefördert (Hussey und Carrington, 1996), (Trætteberg, 2002a). Wie Hussey (2000) darstellt, können auch Probleme mit der Gebrauchstauglichkeit identifiziert werden. So können Fehlbedienungen identifiziert werden, was besonders bei sicherheitsrelevanten Systemen wichtig ist.

Vorteile von formalen Methoden

Nachweis von Fehlern

Formale Methoden garantieren aber keine fehlerfreie Software (Bowen et al., 1995). Wie Jackson (1998) anführt, kann die Verwendung von formalen Methoden zwar das Vorhandensein von Fehlern zeigen, aber nicht gewährleisten, dass keine Fehler existieren.

> „Just because a program is *"proven correct"* [...] you cannot be sure that it will do what you intend."
>
> (Smith, 1985)

Einschränkungen

Ein weiteres Problem ergibt sich aus dem Anwendungsgebiet. Bei der Modellierung des Interaktionsverhaltens zwischen Mensch und Computer ist eine präzise Überprüfung nicht möglich, da zwar funktionale Aspekte, aber kein menschliches Verhalten mithilfe von formalen Verfahrensweisen verifiziert werden kann (Smith, 1985).

> „The computer and its software are formal [...] but the system of which the computer is a part, and the purposes it is intended to serve in the world, are informal."
>
> (Jackson, 1998)

Kommunikationsprobleme

Für interdisziplinäre Teams sind Vorgehensweisen, die auf strikte Formalität setzen nur bedingt geeignet (Strahonja und Picek, 2005), auch deshalb, weil ein großes Maß an Expertise verlangt wird. Lern- und Kommunikationsprozesse zwischen den Interessengruppen werden behindert, da nur Experten und Trainierte mit formalen Vorgehensweisen zurechtkommen (Trætteberg, 2002a). Effektive Softwareentwicklung wird durch nicht oder kaum verwendungsfähige Notationen verhindert (Khazaei und Roast, 2001) und selbst innerhalb der SE-Gemeinde ist es umstritten, wie nützlich formale Methoden wirklich sind (Bowen et al., 1995).

> „Grammar or diagram approaches [...] are clumsy for direct manipulation interface definition, because they cannot conveniently cope with the variety of permissible actions and visual feedback the system provides."
>
> (Shneiderman, 1998)

Präzision am Anfang nicht vorteilhaft

Im Übrigen sind formale Herangehensweisen zu Beginn eines Entwurfsprozesses kaum verwendbar, da die Vorstellungen über die spätere Umsetzung noch mehrdeutig, unvollständig, wechselhaft und inkonsistent sind (Damm et al., 2000), (Paternò, 2003).

> „Difficulties [...] increase with the formality of the model, as more formal representations require the designers to provide more abstractions and to more precisely break their thoughts up. "
>
> (Jarczyk et al., 1992)

Interaktionsverhalten im Vordergrund

Zudem liegt der Fokus meist mehr auf der Beschreibung des Interaktionsverhaltens, als auf der Darstellung des UIs selbst (Bowen, 2005). Im Hinblick auf *User Experience*, *Corporate Identity* und *Coporate Design* wird dies der gestiegenen Bedeutung des UIs nicht gerecht, weil nicht gewährleistet wird, dass das spezifizierte System ästhetischen Ansprüchen genügt (Dix, 1991).

Formale Ansätze nicht geeignet

Die Vorteile von formalen Methoden kommen aufgrund des Anwendungsgebietes und der Zusammensetzung des Teams nicht richtig zum Tragen; die Nachteile behindern innovatives Arbeiten, Kommunikation und Kooperation (Constantine, 2000). Deshalb sind formale Ansätze oftmals nicht geeignet, um HCI und Usability Experten bei ihrer Arbeit zu unterstützen (Bowen, 2005), weshalb Dix (1995b) es als unwahrscheinlich einschätzt, dass formale Ansätze beim UI-Entwurf jemals weit verbreitet sein werden.

Monk et al. (1993) führen an, dass der potenzielle Nutzen von formalen Notationen bei der Modellierung zwar erkannt wird, es aber fünf generelle Hemmnisse für deren weitere Verbreitung gibt:

1. Es ist ein beträchtlicher Lernaufwand nötig, um eine Notation so zu beherrschen, dass ein produktiver Einsatz möglich ist.
2. Es gibt kaum Vorgehensweisen, welche beschreiben, wie eine Notation konkret eingesetzt werden kann und soll.
3. Formale Sprachen sind arbeitsintensiv und schon bei Projekten mit mittlerem Ausmaß unzweckmäßig.
4. Die verfügbaren Modellierungsformen sind nicht für den Entwurf von allen Aspekten tauglich.
5. Anforderungen (und dadurch auch Spezifikationen) ändern sich kontinuierlich während der gesamten Projektlaufzeit.

Trotzdem werden strukturierte Ansätze benötigt, die es ermöglichen, interaktive Systeme zu spezifizieren und zu verifizieren (Jacob, 1983), (Constantine und Lockwood, 1999a), (Blomkvist, 2005), (Memmel et al., 2008c). In Anlehnung an die Motive des *User-Centered Design* wird nach Ansicht von Monk et al. (1993) deshalb ein *„Designer-Centered Design"* benötigt, da die Entwickler besser bei der Modellierung unterstützt werden müssen.

4.1.4 Informale Ausdrucksweisen

Informale Ansätze bieten Spielraum

Informale Darstellungen basieren nicht auf strikt vorgegebenen Regeln und unterstützen neben der Möglichkeit zur Abstraktion auch Mehrdeutigkeiten, Unbestimmtheiten und Ungenauigkeiten (Landay und Myers, 1995), (Gross und Yi-Luen Do, 1996). Sie lassen Handlungsspielraum für Entscheidungen zu und unterstützen deshalb kreatives Vorgehen, weshalb sie sich besonders für die Ideenfindung und das Ausloten der Interaktionsmöglichkeiten eignen (van Welie, 2001). Dies ist besonders geeignet zu Beginn des Entwicklungsprozesses, ob es erforderlich ist ein erstes Verständnis der Anforderungen zu erhalten (Paternò und Volpe, 2005). Darüber hinaus sind informale Veranschaulichungen direkt manipulierbar und können schnell erstellt werden, ohne dass Rücksicht auf Darstellung oder Formalitäten genommen werden muss.

Kreative Vorgänge werden unterstützt

Wie Mackay et al. (2003) feststellen, wird auf Modelle, die einen eher unfertigen und rauhen Charakter besitzen, nicht zuviel Aufmerksamkeit verschwendet und sie werden eher Änderungen unterzogen als bei detaillierten Darstellungen. Dies ist gerade bei kreativen Entwurfsprozessen wichtig, bei denen sich die Phasen zwischen Gestaltung, Bewertung und Reflexion abwechseln (Schön, 1983), (Winograd und Flores, 1987), (Brown et al., 2008) (siehe Abb. 4.2).

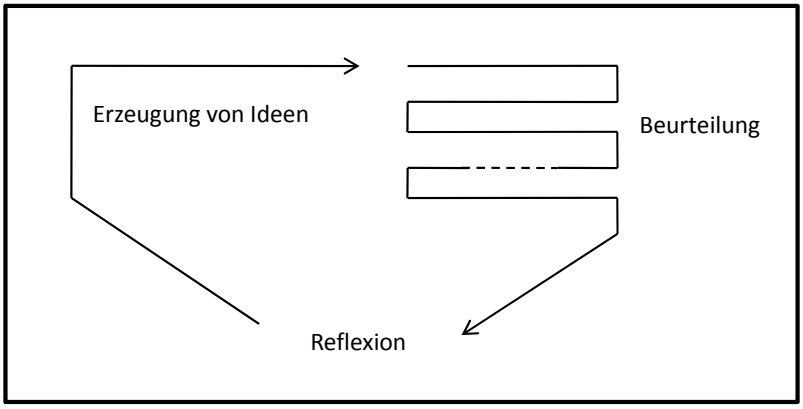


Abbildung 4.2: Aktivitätenzyklus bei kreativen Vorgängen (Brown et al., 2008)

Vage und mehrdeutig

Die Erstellung von konkreten Entwürfen in frühen Phasen benötigt viel Zeit und zwingt die Entwickler durch die detaillierte Darstellung, einmal eingeschlagene Richtungen beizubehalten (Hudson, 1994), (Landay und Myers, 1995), (Paternò, 2003). Anstatt sich durch die Verwendung von präzisen Ausdrucksmitteln bereits in einem frühen Stadium auf eine Realisierung festzulegen, ist es vielversprechender, sich durch vage und ungenaue Ausdrücksmöglichkeiten erst einmal voranzu-

tasten und später auf detailliertere Darstellungsformen zurückzugreifen. So wird es möglich, auch Komponenten zu integrieren, die nicht auf herkömmlichen *Widgets* basieren (Lecolinet, 1998). Im Gegensatz zu formalen Modellen, die den Fokus mehr auf präzise, vollständige und endgültige Darstellungen legen (Trætteberg, 2002a), wird die Ungeschliffenheit von informalen Repräsentationen als Vorteil betrachtet (Landay und Myers, 1995). Sie wirken stimulierender und weniger einschränkend, weil diese Darstellungen ungewiss bleiben, sich auf nichts Konkretes festlegen und Freiraum für Kreativität lassen (Trætteberg, 2002a).

> „If design representations are to be used as *"typical examples"* and *"paradigm cases"* they must be effortless to relate to the current work and design context, and hence should be concrete and informal and not abstract and formal models."
>
> (Ehn, 1992)

Skizzen auf Papier sind verbreitet

Für die Modellierung von ersten Entwürfen werden üblicherweise Skizzen oder *Sketches* verwendet (Coyette und Vanderdonckt, 2005). Es kommt dabei vor allem auf die schnelle Erzeugung von unterschiedlichen Varianten und die Darstellung von Ideen an (Zenka und Slavik, 2004), weniger auf die detaillierte Definition des *Look & Feel* (Hong et al., 2001), (Kimmond, 1995), (Landay und Myers, 2001). Auf Grund von fehlenden Werkzeugen und der besseren Flexibilität wird dies hauptsächlich auf Papier durchgeführt (Newman und Landay, 2000). *Sketches* sind einfach nachzuvollziehen und erlauben die Evaluation von strukturellen Aspekten schon in sehr frühen Phasen (van Welie, 2001). Wegen der Möglichkeit zur einfachen und schnellen Änderung von *Sketches* wird kollaboratives Arbeiten unterstützt (Coyette und Vanderdonckt, 2005). Durch ihren „unfertigen" Zustand tritt ihr konzeptioneller Charakter deutlich hervor, weshalb es für Entwickler einfach ist, sich nicht auf die Verfeinerung, sondern auf Iterationen der *Sketches* zu konzentrieren (Coyette und Vanderdonckt, 2005).

> „It is important to iterate quickly in the early part of the design process because that is when radically different ideas can and should be generated and examined."
>
> (Landay und Myers, 1995)

Leicht anzuwenden und einfach zu verstehen

Daneben bieten informale Modelle einen nicht zu unterschätzenden Vorteil bei gemeinschaftlich durchgeführten Entwürfen, gerade wenn auch Personen teilhaben, die unerfahren beim Gebrauch von formalen Methoden sind. Durch ihren fassbaren und anschaulichen Charakter unterstützen sie den Dialog, die Entscheidungsfindung und die Validierung von Beschlüssen bei kollaborativen Entwurfsprozessen (Trætteberg, 2002a). Gerade auch weil Vorwissen nicht berücksichtigt werden muss, sind sie leicht verständlich und einfach zu erzeugen (Mackay et al., 2003), (Barbosa et al., 2004), (Memmel et al., 2007c).

Kategorisierung	Artefakte
Diagramme	SiteMaps, MindMaps, Information Hierachies, Content Maps, Flow Charts
Ablaufpläne	Storyboards, Storybooks, Navigation Charts
Prototypen	Schematics, Mockups, Prototypes
Dokumente	Specifications, Guidelines, Styleguides, Scenarios

Tabelle 4.1: Informale Artefakte für Spezifikationen nach Newman und Landay (2000), Bailey et al. (2001) und Paternò und Volpe (2005)

Wie Ambler und Jeffries (2002) oder auch Offergeld und Oed (2006) aufzeigen, kann die Spezifikation deshalb oft nicht richtig interpretiert werden und Anforderungen werden missverstanden bzw. falsch interpretiert. Dadurch ist es nicht möglich, das gewünschte System zu erstellen. Die Folge ist ein Produkt, das verspätet ausgeliefert wird, kostenintensiv nachgebessert werden muss und über schlechte Qualität sowie geringe Innovation verfügt.

> „So we need to elaborate a representation or a modeling language that fosters interdisciplinary discussion, represents the designers' vision of the essence of the system being designed and thus serves as a reference of what needs to be done, to which every professional must commit."
>
> (Barbosa et al., 2004)

4.2 Techniken zur Realisierung einer interdisziplinären Spezifikation

Bridging the Gaps

In Kapitel 3.1—"Brückenschläge zwischen den Disziplinen" wurden Vorgehensweisen vorgestellt, welche die Modellierung des UIs ermöglichen indem sie die Disziplinen HCI und SE kombinieren. Wie Paech und Kohler (2003) zeigen, beginnt dieses *Bridging the Gaps* aber bereits bei der Anforderungsermittlung, weshalb es zusätzlich erforderlich ist, dass die Disziplin der Geschäftsprozessmodellierung ausreichend berücksichtigt wird. Erst wenn Akteure aus allen Interessengruppen in der Lage zur kollaborativen Zusammenarbeit sind, ist es möglich benutzer- und aufgabengerechte UIs zu entwickeln (Memmel und Reiterer, 2008c).

Geeignete Notationen müssen gefunden werden

Deshalb müssen Modelle und Notationen gefunden werden, die von allen Interessengruppen verstanden werden. Informale Modelle sind für das Verständnis sehr gut geeignet, aber nicht ausreichend, um funktionale Aspekte darzustellen. Gerade in späteren Phasen des Entwurfs werden deshalb formellere Darstellungen benötigt, um Funktionalität, strukturelle Aspekte und interne Vorgänge in geeigneter strukturierter Form gegenüber den späteren Entwicklern zu kommunizieren. (Monk et al., 1993), (Trætteberg, 2002a), (McInerney und Sobiesiak, 2000), (Monk et al., 1993), (Memmel et al., 2007c). Die Spezifikation wird sonst zu einer *Black Box*, die wenig Rückschlüsse auf den inneren Aufbau und die internen Abläufe zulässt.

Mix an Modellen

Um eine kollaborative Spezifikation der verschiedenen Interessengruppen bei interaktiven Systemen zu ermöglichen, ist deshalb ein Mix von formalen und informalen Modellen nötig (Paternò, 1999). Informale Modelle sollen Kreativität, Kollaboration und Kommunikation zwischen den Beteiligten möglich machen, formale Darstellungen hingegen unterstützen die Analyse von funktionalen Anforderungen, ermöglichen Reflexion und sorgen für Eindeutigkeit (Trætteberg, 2002b).

> „One issue that needs to be addressed is how formally based system modelling can be integrated with the less formal requirements specification techniques typically found in industry."
>
> (Fields et al., 1994)

4.2.1 Agile Modeling

Agiler Ansatz

Um die Modellierung einfacher zu gestalten und einen Brückenschlag zwischen den Disziplinen zu ermöglichen, ist es notwendig, Notationen zu finden, welche einfach anzuwenden sind. Wie Gundelsweiler et al. (2004), Memmel et al. (2007b), Memmel et al. (2007d) und Ferreira et al. (2007) gezeigt haben ist dies durch die Verwendung von flexiblen Techniken wie z.B. *Agile Modeling* möglich. Da Vorgehensweisen mit agilen Methoden für das RE (Lopez-Nores et al., 2006), (Düchting et al., 2007) und die Disziplinen SE (Larman, 2003), BPM (Petersen und Wiil, 2008) und HCI (Constantine und Lockwood, 1999a), (Constantine, 2001) vorhanden sind und auch bereits Anwendung finden, kann von einem geeigneten und erprobten Konzept ausgegangen werden.

Generell gilt, dass diese agilen Methoden auf leichtgewichtigen, iterativen Prozessen basieren, welche sehr gut an aktuelle Bedürfnisse angepasst werden können (Mackay et al., 2003). Anstelle von aufwendigen Verfahren mit intensiver Dokumentation werden Abläufe verwendet, bei denen viel Wert auf erhöhte Kommunikation und Zusammenarbeit gelegt wird (Memmel et al., 2007a), (Cao und Ramesh, 2007).

Einfache Modellierungsformen

Speziell für interdisziplinäre Teams ist das von Ambler und Jeffries (2002) vorgestellte *Agile Modeling* sehr gut geeignet. Ihre Philosophie basiert auf der Annahme, dass die besten Ergebnisse bei der Softwareentwicklung dann erzielt werden, wenn das Team nicht zu viele Experten beinhaltet und wenn so wenig wie möglich modelliert wird. Sie ziehen deshalb Arbeitsgruppen vor, welche sich durch Personen aus unterschiedlichen Arbeitsgebieten zusammensetzen. Da nicht alle Teilnehmer die Fähigkeiten und Kenntnisse für eine präzise Modellierung besitzen, müssen einfache, allgemeingültige Modellierungsformen verwendet werden (Ambler, 2003). *Agile Modeling* ist dabei keine Vorgehensweise, sondern ein Ansatz, der es möglich macht Prozesse und Modelle flexibel zu gestalten, sodass gemeinschaftliches Arbeiten von interdisziplinären Teams möglich wird.

Verkleinerte Modellpalette

Die Prinzipien des *Agile Modeling*[2] (siehe Tab. 4.2), ermöglichen deshalb Notationen auf eine Untermenge zu beschränken[3]. Begründet wird dies unter anderem dadurch, dass es nicht möglich ist, mit einer Notation allen gestellten Ansprüchen gerecht zu

[2] http://www.agilemodeling.com/principles.htm
[3] http://www.agilemodeling.com/artifacts/

werden. Jede Modellierungsart hat ihre Stärken und Schwächen. Es sind deshalb auch bei präzisen und detaillierten Methoden Kompromisse nötig. So ist es z.B. mit der UML möglich, 80 % der Bedürfnisse mit 20 % der Modelle darzustellen (Ambler, 2003), weshalb nicht alle Notationen unbedingt nötig sind. Die Verwendung von Modellen mit möglichst geringer Komplexität wird unter anderem auch von Constantine et al. (2003) vorgeschlagen.

Prinzip	Beschreibung
Assume Simplicity	Die einfachste Lösung ist die beste Lösung.
Embrace Change	Anforderungen und Standpunkte verändern sich mit der Zeit.
Enabling the Next Effort is Your Secondary Goal	Zukünftige Entwicklungen und Erweiterungen der Software müssen beachten werden.
Incremental Change	Einfache Modelle reichen, bei Bedarf können sie inkrementell entwickelt oder verworfen werden.
Maximize Stakeholder ROI	Stakeholder investieren viel, - Zeit, Geld, etc. - deshalb sollte das Maximum aus der Zusammenarbeit erzielt werden.
Model With a Purpose	Das Ziel und die Absicht des Modellierens sollte stehts im Vordergrund stehen. Modelle die für besseres Verständnis sorgen benötigen z.B. andere Detaillgrade wie Modelle für die Dokumentation.
Multiple Models	Verwendung von mehreren verschiedenen Modellen und Anpassung der Modellpalette an das Projekt.
Quality Work	Qualitätiv hochwertige Arbeit hat höchste Priorität.
Rapid Feedback	Enge Zusammenarbeit, gute Kommunikation und schnelle Rückmeldungen sind wichtig.
Travel Light	Die Aktualisierung von Modellen kostet Zeit, weshalb genau überlegt werden sollte, welche Modelle überhaupt (noch) notwendig sind.
Working Software Is Your Primary Goal	Das Ziel ist nicht die Erstellung von Artefakten, Dokumentationen oder Modellen, sondern qualitativ hochwertige Software.

Tabelle 4.2: Die Prinzipien des Agile Modeling

Modelle sollen gerade ausreichend genug sein

Aus diesem Grund werden Modelle verwendet die "gerade ausreichend genug" sind, unabhängig davon, ob es sich um Skizzen, UML *Statecharts* oder Datenbankmodelle handelt. Um seinen Zweck zu erfüllen, muss ein agiles Modell deshalb nicht detailliert sein und den exakten Notationsregeln entsprechen (siehe Abb. 4.3). Nicht das Modell ist das Ziel, sondern der Vorgang der gemeinsamen Ideenfindung während der Modellierung. Durch die erhöhte Kollaboration und Kooperation während des Modellierens können die Beteiligten verstehen, was erreicht werden soll, auch wenn das Modell nicht den Regeln der jeweiligen Notation entspricht.

> „An agile model is just barely good enough - it meets its goals and no more. [...] Simple tools are easy to work with, inclusive (my stakeholders can be actively involved with modeling), and flexible, and they're not constraining."
>
> (Ambler, 2003)

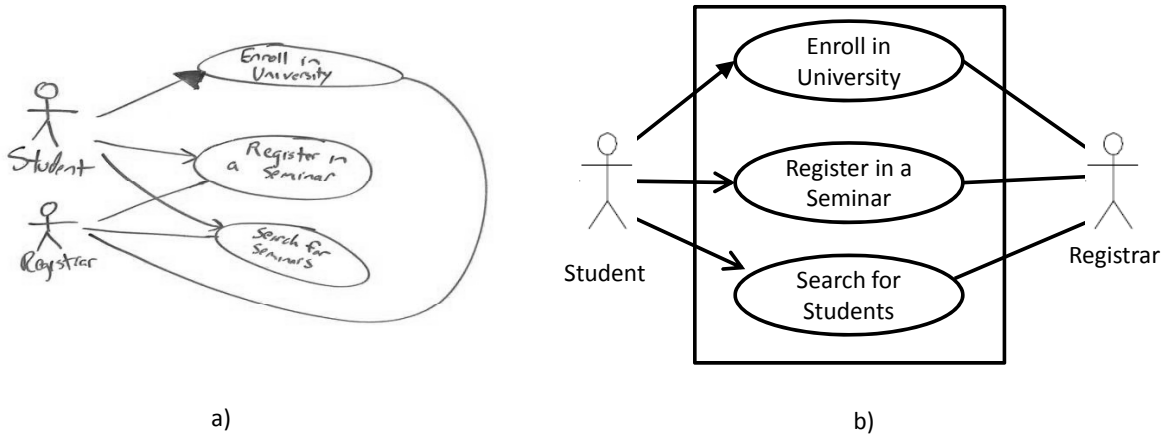


Abbildung 4.3: Kongruente Darstellungen eines Use Case Diagramms als Skizze nach den Prinzipien des Agile Modeling (a) und in UML Notation (b) (vgl. Ambler (2006))

Resultierende Fehler gering halten

Da auch mit formalen Beschreibungen keine fehlerfreie Software erstellt werden, kann (siehe Kapitel 4.1.3—"Formale Ausdrucksweisen", ist es nicht unbedingt notwendig, präzise Modellierungssprachen zu verwenden. Stattdessen ist es besser, Beschreibungen zu verwenden, die garantieren, dass der resultierende Fehler klein genug ist, um ein ausreichend verlässliches System zu erstellen (Jackson, 1998). Dabei ist es notwendig, dass das jeweilige Anwendungsgebiet bewertet wird, denn bei Benutzerschnittstellen für sicherheitsrelevante Systeme gelten andere Prämissen als bei gewöhnlichen Internetseiten des *World Wide Web*.

Anzahl und Detailgrad der Modelle einschränken

Werden die Prinzipien des AM auf den Vorgehensprozess und auf Notationen, wie z.B. das eher formale UML, angewandt, so wird es möglich, die Anzahl und den Detailgrad der Modelle einzuschränken. Durch die Verwendung von geeigneten Modellierungsmethoden und intensive Kommunikation ist eine Spezifikation des UIs aber trotzdem möglich (Memmel et al., 2007a), (Memmel und Reiterer, 2008c). Dies ist gerade bei interdisziplinären Vorgängen nötig, um eine wirkliche Zusammenarbeit zu erreichen.

4.2.2 Modellierung mit Prototypen

Prototypen verbessern die Kommunikation

Modelle und textbasierte Spezifikationen alleine reichen nicht aus, um den Entwurf gegenüber anderen zu kommunizieren, da sie es nicht im ausreichenden Maße schaffen, ein Gefühl zu vermitteln, wie das System aussieht und sich verhalten soll. Darstellungen, die für das Explorieren und die Kommunikation von Entwürfen geeignet sind in dem Bewertungsraster in Abbildung 4.4 zu sehen. Für einfache Entwürfe, die zu Beginn der Entwicklung erstellt werden eignen sich Storyboards, Skizzen und Text, während in späten Phasen, bei denen bestimmte Entscheidungen schon feststehen, Prototypen für die Modellierung favorisiert werden.

Bei der Zusammenarbeit von verschiedenen Interessengruppen und in den Disziplinen RE (Sidoran et al., 1995), (Cao und Ramesh, 2008), SE (Zhang et al., 2004), (Gunaratne et al., 2004), HCI (Nielsen, 1992), (Bäumer et al., 1996), (Mayhew, 1999) und BPM (Okawa et al., 2007), (Sukaviriya et al., 2007) finden Prototypen Verwendung, weshalb sie sich besonders zur Unterstützung der interdisziplinären Spezi-

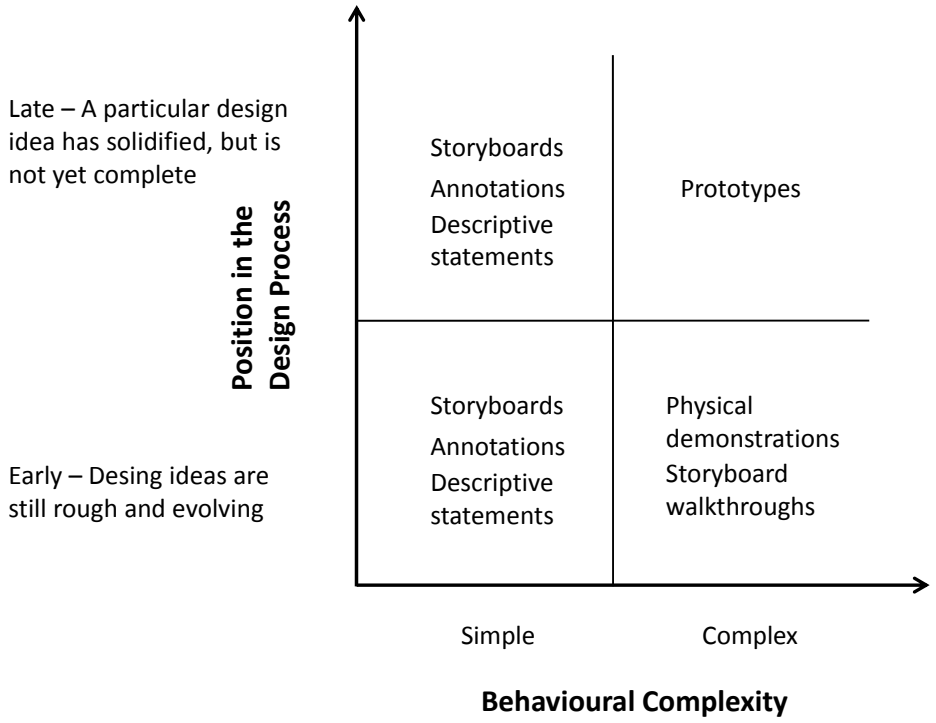


Abbildung 4.4: Einordnungsraster von verschiedenen Modellierungsformen, für frühe und späte Entwicklungsphasen (vgl. Bailey et al. (2001))

fikation von UIs eignen (Blomkvist, 2005), (Memmel et al., 2007a). Für ein besseres Verständnis und zur Förderung der Kommunikation sind Prototypen deshalb ein geeignetes Mittel und ein weiterer Brückenschlag zwischen den Disziplinen (Memmel et al., 2007c).

> „Within some innovation environments, prototypes effectively become the media franca of the organization: the essential medium for information, interaction, integration, and collaboration. "
>
> (Schrage, 1996)

Prototypen sind unbedingt notwendig

Für leichtgewichtige und flexible UI-Prozesse sind Prototypen eine Notwendigkeit. Sie sorgen für eine bessere Kommunikation und Kooperation unter den *Stakeholdern*, da sie die abstrakten Konzepte konkret und erlebbar machen. Die Problemstellung verliert an Komplexität und es wird ein effektives Verständnis der Anforderungen gewährt (Zhang et al., 2004). Dadurch wird mehr Transparenz und Nachvollziehbarkeit gewährleistet, jeder kann aktiv an der Gestaltung partizipieren und die Zusammenarbeit wird verbessert.

Frühe Analyse und Validierung möglich

Prototypen können mit wesentlich geringerem Aufwand hergestellt werden als das geplante Produkt und müssen nicht unbedingt alle Eigenschaften des Zielsystems besitzen. Deshalb können Sachverhalte und Ideen für den Entwurf des UIs bereits in frühen Phasen analysiert und validiert werden (Bäumer et al., 1996), (Elkoutbi et al., 1999), (Myers und Buxton, 1986). Zudem bieten Prototypen die nötige Flexibilität, um schnell an geänderte Vorstellungen oder Anforderungen angepasst werden

zu können (Boehm et al., 1984), (Lin et al., 2000). So wird es möglich, dass sich das Ergebnis des Spezifikationsprozesses immer mehr an eine Entwurfslösung *par excellence* annähert (Fitton et al., 2005).

Qualität der Ergebnisse verbessert sich

Durch die Verwendung von Prototypen sind schnellere Rückkopplungen unter den Beteiligten möglich, wodurch die Iterationszyklen beschleunigt werden und sich die Qualität der Ergebnisse verbessert (Gunaratne et al., 2004), (Memmel et al., 2007a), (Kimmond, 1995), (Ferreira et al., 2007). Constantine (2000) führt an, dass die UIs innovativer werden und besser auf die Bedürfnisse der Benutzer zugeschnitten sind. Zudem wird die Produkteinführungszeit verkürzt, weil Anforderungen schneller geklärt und validiert werden können (Carr, 1996). Darüber hinaus wird die Umsetzung einfacher, da der Prototyp als genaue Vorlage dient. Somit werden Anforderungsdokumente zum Teil überflüssig, weil Erfordernisse direkt über den Prototyp kommuniziert werden können (Cao und Ramesh, 2008). Dies führt sogar so weit, dass die Meinung vertreten wird, dass die Verwendung von Prototypen bestehende *Software-Lifecycles* überflüssig macht (McCracken und Jackson, 1982).

Thinking through prototyping

Wie Klemmer et al. (2006) beschreiben, fördert das Erzeugen von Prototypen überraschende und unerwartete Erkenntnisse, die äußerst wichtig während des Entwurfs sind und welche sich ohne die konkrete Umsetzung nicht aus den abstrakten Vorstellungen ergeben hätten. Die gegenstandslosen Anforderungen werden infolgedessen konkret und erlebbar wodurch weitere Rückschlüsse zugelassen werden. Sie bezeichnen dies als *"Thinking through prototyping"* weil Probleme aufgedeckt werden und Anregungen für neue Konzepte entstehen.

Abstrakt zu Konkret

Prototypen werden verwendet, um Entwürfe und Vorstellungen festzuhalten und die Oberfläche von UIs zu modellieren. Ästhetik, Inhalt und Navigation können dargestellt werden, sodass sich Ideen und Konzepte manifestieren (Bailey et al., 2001), (Zenka und Slavik, 2004), (Walker et al., 2002). Der Prototyp kann demonstriert werden und dient als Basis für weitere Diskussionen. Dadurch entstehen Synergieeffekte, weil die unanschaulichen Anforderungen der Spezifikation sowie die Ideen zur Bewältigung ausdrucksvoll dargestellt und materialisiert werden (Hardtke, 2004), (Klemmer et al., 2006).

> „Prototypes serve as a common language between stakeholders, offering a way for designers to explore design ideas and elicit feedback from stakeholders prior to committing to designs."
>
> (Petrie und Schneider, 2006)

Leicht verständlich

Zudem wird sichergestellt, dass keine Interessengruppe beim Gedankenaustausch und bei der Validierung ausgeschlossen wird, da der Prototyp allen zugänglich gemacht werden kann. Für das Verständnis und für die Abänderung von Prototypen sind in der Regel keine besonderen Voraussetzungen notwenig, weshalb alle am Systementwurf teilhaben können (Myers und Buxton, 1986), (Memmel et al., 2007a).

Wie Klemmer et al. (2006) zeigt, können Prototypen für die UI-Gestaltung in allen Phasen des Entwurfs verwendet werden und lassen sich aufgrund ihrer Bestimmung oder Verwendung in die Kategorien von Floyd (1984) - explorativ, experimentell oder evolutionär - einteilen (siehe Tab. 4.3).

Kategorie	Beschreibung
Explorativer Prototyp	Exploratives Prototyping soll die Problemstellung und die Anforderungen klären sowie verschiedene Lösungen demonstrieren. Es werden Ideen und Konzepte ausgelotet. Dabei wird der Anwendungsbereich sowie die Realisierbarkeit analysiert und kommuniziert. Diese Prototypen geben einen ersten Eindruck und sind weit vom tatsächlichen System entfernt.
Experimenteller Prototyp	Das experimentelle Prototyping findet während des Entwurfs statt, wenn die Anforderungen schon weitestgehend geklärt sind. Funktionalitäten werden abgebildet und Lösungsideen werden auf ihre Tauglichkeit untersucht.
Evolutionärer Prototyp	Evolutionäres Prototyping findet während des ganzen Entwicklungsprozesses statt und beinhaltet die explorative und experimentelle Prototypingphase Die gewonnen Erkenntnisse werden übernommen und der Prototyp wird in einem schrittweise aufbauenden Vorgang inkrementell verfeinert.

Tabelle 4.3: Kategorisierung von Prototypen nach (Floyd, 1984)

Explorative und experimentelle Prototypen

In frühen Phasen des UI Entwurfs werden explorative und experimentelle Prototypen in mehreren Varianten entwickelt (Hardtke, 2004), (Memmel und Reiterer, 2008c). Sie sind meist abstrakter Natur und müssen nicht besonders detailliert sein. Sie sollen Mehrdeutigkeiten, Inkonsistenzen und Unvollständigkeiten in den Anforderungen klären, der Ideenfindung dienen und die Machbarkeit von Systemen zeigen. Daneben zeigen sie, dass die Anforderungen richtig verstanden wurden, und bieten Lösungsmöglichkeiten, die anschließend auf ihre Tauglichkeit untersucht werden können (Sidoran et al., 1995), (Strahonja und Picek, 2005).

Evolutionäre Prototypen

Evolutionäre Prototypen hingegen besitzen keinen Wegwerf-Charakter, sondern begleiten die einzelnen Phasen des Entwicklungsprozesses. Sie werden erzeugt, evaluiert und danach weiter verbessert (Myers und Buxton, 1986). Dies kann so weit gehen, dass am Ende keine großen Unterschiede mehr zum Zielsystem bestehen, und sogar Teile bei der Realisierung übernommen werden können (siehe Abb. 4.5).

Prototypen ergänzen sich mit Spezifikationen

Prototypen können verwendet werden, um die Spezifikationen von UIs zu ergänzen, da sie Anforderungen konkret abbilden und daneben als Referenz bei der späteren Umsetzung dienen können (Boehm et al., 1984) (Sidoran et al., 1995), (Heimdahl und Thompson, 2000). Verständnis- und Verständigungsprobleme verschwinden und es wird eine bessere Partizipation der Interessengruppen erreicht (Carr, 1996), wodurch Zeitaufwand, Kosten und das Risiko von Fehlentwicklungen minimiert werden.

> „Prototypes aid in refining requirements and may be used as a specification for developers."
>
> (Petrie und Schneider, 2006)

Unterschiedliche Erscheinungsformen

Nach der Definition von Houde und Hill (1997) kann jede Repräsentation einer Entwurfsidee als Prototyp bezeichnet werden, ganz egal, welches Medium dabei Verwendung findet. Gerade zu Beginn des Entwurfsprozesses wird oftmals auf Papier und Bleistift zurückgegriffen. Zum einen, weil dies sehr flexibel ist, von jedem durchgeführt werden kann und zum anderen, weil geeignete Werkzeuge fehlen (Newman und Landay, 2000), (Coyette und Vanderdonckt, 2005), (Memmel et al.,

2007d). Das Resultat sind abstrakte *low-fidelity* Prototypen oder *Mockups*. Es kommt dabei weniger auf die detaillierte Definition des Aussehens und des Verhaltens an (Kimmond, 1995), (Hong et al., 2001), (Landay und Myers, 2001), (Phillips und Joe, 2005), sondern vor allem auf die Strukturierung der einzelnen Elemente, die Darstellung von Ideen und die Erzeugung von unterschiedlichen Varianten (Constantine et al., 2003), (Zenka und Slavik, 2004).

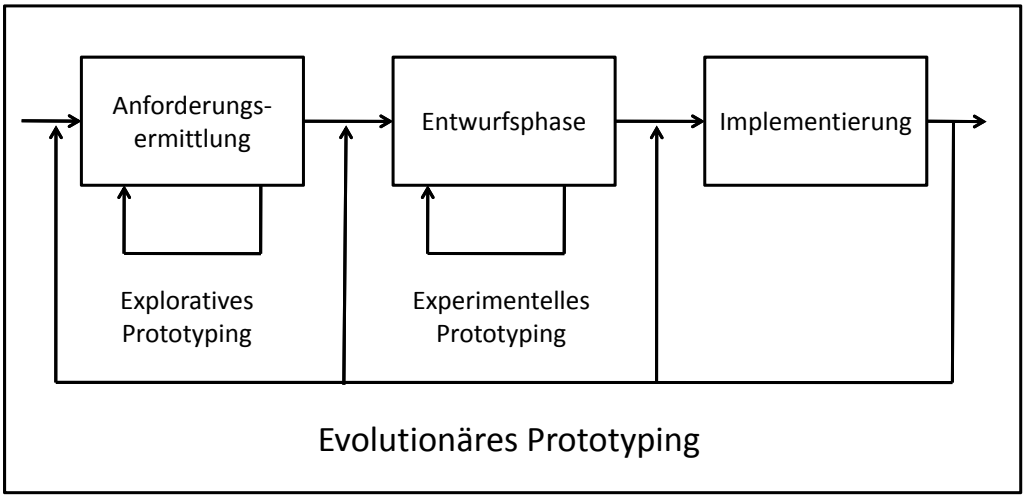


Abbildung 4.5: Verwendung von Prototypen während des Entwicklungsprozsesses

High- und low-fidelity Prototypen

Für die Spezifikation eines UIs reichen abstrakte Darstellungen aber nicht aus. Das Aussehen wird nicht definiert und das Verhalten bleibt vage und ungenau. Deshalb ist es notwendig, exaktere Repräsentationen zu verwenden, nachdem die Abgrenzung des Entwurfs mit *low-fidelity* Prototypen durchgeführt wurde (Memmel et al., 2007c).

High-fidelity Prototypen können sowohl detaillierte Zeichnungen als auch voll funktionsfähige und interaktive Simulation sein. Dadurch können sie vom Detail- und Interaktionsgrad sowie der Gestaltung bis an die spätere Implementierung heranreichen. Bei der Erstellung werden oftmals die gleichen Techniken, Werkzeuge und Methoden verwendet, die auch bei der späteren Herstellung des Endprodukts eingesetzt werden, weshalb ihre Erstellung viel Zeit und Aufwand benötigt (Walker et al., 2002), (Hardtke, 2004), (Lin et al., 2000). Für die Sondierung von Entwurfsentscheidungen in frühen Phasen sind sie deshalb nicht geeignet (Memmel et al., 2007d). Die weiteren Unterschiede zwischen den verschiedenen Ausprägungen sind in Tabelle 4.4 aufgeführt.

Vor- und Nachteile

Wie Landay und Myers (1995) sowie Newman und Landay (2000) aufzeigen, kann deshalb die frühe und schnelle Exploration von Gestaltungsentscheidungen bei der Verwendung von *high-fidelity* Prototypen behindert werden. Im Gegenzug wird dadurch aber die Evaluation, Dokumentation und Verteilung einfacher (Walker et al., 2002). *Low-fidelity* Prototypen hingegen sind schnell und flexibel erstellbar, aber nicht ausreichend geeignet, um Entscheidungen über das Verhalten zu sondieren oder anderen mitzuteilen. Dieser Gegensatz wird von Bailey und Konstan (2003) als *early investment/communication gap* bezeichnet.

Modelle	Erscheinungsbild	Vorteile	Nachteile
Low-Fidelity	Skizzenhaft, geringe Detaillstufe	- Geringe Entwicklungszeit - Niedrige Kosten - Leicht verständlich - Einfach zu erstellen - Evaluationen möglich	- Geringer Nutzen wenn Anforderungen erkannt wurden - Ungenügende Spezifikationsfähigkeit - Keine ansprechende Aufmachung - Weniger geeignet um Verhalten und Interaktionen darzustellen
High-Fidelity	Aussehen und Verhalten entsprechen dem erwünschten UI	- Teilweise oder komplett Funktional - Interaktive „lebende“ Spezifikation - Ansprechende Gestaltung - Argumetationshilfe ggb. Entscheidungsträgern - Kann evaluiert und validiert werden	- Unflexibel bei sich ändernden Anforderungen - hohe Kosten und großer Zeitaufwand - Anforderugnen an die Erstellung sind hoch - Überblendet und verfestigt Entwurfsfehler - Uneffektiv für konzeptionelle Entwürfe - Kann als bereits umgesetztes System missverstanden werden

Tabelle 4.4: Vergleich zwischen Prototypen mit unterschiedlicher Genauigkeit nach Coyette und Vanderdonckt (2005) und Rudd et al. (1996)

Evaluation der Gebrauchstauglichkeit

Für die Bewertung der Gebrauchstauglichkeit eignen sich *low-* und *high-fidelity* Prototypen gleichermaßen. Signifikante Unterschiede in der Anzahl an gefundenen Fehlern sind nicht ersichtlich (Virzi et al., 1996), (Walker et al., 2002). Allerdings zeigt Hong et al. (2001), dass Benutzer bei der Validierung von konkreten Darstellungen davon ausgehen, dass diese fertig und unveränderlich sind, weshalb Änderungsvorschläge eher die Navigation und die Struktur betreffen. Im Gegensatz dazu werden informale Abbildungen als unfertig und änderbar empfunden, wodurch eher Anregungen für die Korrekturen der visuellen Darstellung gemacht werden. Aus Entwicklersicht sieht dies anders aus, wie die Ergebnisse von Landay und Myers (1995) zeigen. Sie führen an, dass detaillierte Abbildungen die Diskussionen über visuelle Gestaltung mehr fördern als abstrakte Entwürfe. Da sich beide Herangehensweisen ergänzen, ist es also notwendig, beide Ausprägungen bei der Spezifikation zu unterstützen.

Look & Feel nur durch interaktive Prototypen

Oftmals wird auf die Aussage verwiesen, dass ein Bild mehr als tausend Worte aussagt, um die Vorteile von Prototypen zu erläutern. Wie Geiger und Müller (1998) zeigen, sind dies aber oftmals nicht die gleichen Worte, weshalb das Verhalten von statischen Prototypen nicht ausreichend ausgedrückt werden kann. Für die Darstellung der Interaktionsmöglichkeiten sind interaktive *high-fidelity* Prototypen deshalb am besten geeignet (Memmel et al., 2007c). Sie können das erwünschte dynamische Verhalten und das Zusammenspiel zwischen UI und Benutzer demonstrieren und spezifizieren (Paternò, 2003). Das *Look & Feel* kann erlebt werden, wodurch sich neue Rückschlüsse bei der Validierung des Verhaltens und der Anforderungen ergeben.

4.2.3 Design Rationale

Traceability

Da Anforderungen und Entwurfsentscheidungen voneinander abhängig sind, tritt ein Problem auf, das in Kapitel 2.4—“Disziplinenübergreifende Zusammenarbeit bei der Anforderungsermittlung” erwähnt wird. Ergeben sich z.B. neue Erkenntnisse aus der Evaluation der Prototypen, dann müssen die Anforderungen überprüft und gegebenenfalls auf den neuesten Stand gebracht werden. Gleiches gilt, auch in umgekehrter Richtung, wenn Konzepte und Modelle aufgrund von korrigierten Anforderungen modifiziert werden müssen. Es ist also wichtig, dass Kausalitäten

nachvollziehbar und verfolgbar sind, wenn erreicht werden soll, dass die sich entwickelnde Spezifikation stringent und verständlich bleibt. Wie Falessi et al. (2006) zeigen, kann dies durch die Dokumentation mithilfe von *Design Rationale* erreicht werden.

> „Design rationale, in its simplest, is the explicit listing of decisions made during a design process and the reasons why those decisions were made."
>
> (Jarczyk et al., 1992)

Argumente sowie Alternativen

Design Rationale ist eine strukturierte Auflistung aller Gründe, die zu Entscheidungen bei der Umsetzung geführt haben. Zudem werden betrachtete Alternativen aufgeführt, sowie Argumente die für oder gegen diese Art der Umsetzung sprechen (Carroll, 1991), (Lee, 1997). Dadurch wird eine *Traceability* und Transparenz hergestellt und es ist möglich, den Zusammenhang von Ursache und Wirkung bzw. Anforderung und Umsetzung festzustellen.

Monk et al. (1995) identifizieren die folgenden Gründe, die für die Verwendung von *Design Rationale* sprechen:

1. Ideen und Konzepte können verdeutlicht werden, zudem wird klar, dass Entscheidungen nicht auf willkürlichen Entschlüssen basieren.
2. Es wird eine strukturierte Methodik zur Entscheidungsfindung geliefert.
3. Nachdem Anforderungen revidiert wurden, können getroffene Entscheidungen erneut beurteilt und bewertet werden. Die daraus folgenden Konsequenzen können abgeschätzt werden und es wird möglich festzustellen, ob Teile des Systems geändert oder neu entwickelt werden müssen.
4. *Design Rationale* können wiederverwendet werden, wodurch auch dazugehörende Entwürfe und Designs erneut genutzt werden können.

Entwicklungsvorgang und Entscheidungsfindung werden beschleunigt

Wie Bratthall et al. (2000) in einer Studie zeigen, können quantitative und qualitative Verbesserungen erzielt werden, wenn ein *Design Rationale* zur Verfügung steht. Die Geschwindigkeit, die zur Durchführung von Änderungen benötigt wird, kann signifikant gesteigert werden. Daneben wird der Entwicklungsvorgang beschleunigt und es kann eine höhere Fehlerfreiheit erzielt werden. In einer weiteren Studie von Falessi et al. (2006) zeigte sich, dass die Entscheidungsfindung für den Einzelnen, aber auch für das Team signifikant verbessert wurde. Darüber hinaus steigert sich die Kommunikation und Kollaboration zwischen den Beteiligten, wie Lee (1997) sowie Jarczyk et al. (1992) aufzeigen.

Design Rationale als Wissensbasis

Zudem können Gründe für einzelne Entscheidungen bewahrt werden, sodass auch im Nachhinein nachvollzogen werden kann, warum Beschlüsse in dieser Form getroffen wurden. Dies ist durch die Verwendung einer herkömmlichen Spezifikation nicht möglich (Monk et al., 1995), weil dort nur die Anforderungen dokumentiert sind, aber nicht die Entscheidungsfindung. In diesem Fall dient das *Design Rationale* als Archiv, in welchem das Wissen und die gemachten Erfahrungen gespeichert sind. Im Übrigen sind Erfahrungswerte dann nicht an einzelne Mitarbeiter gebunden, weshalb Know-how sowie Sachkenntnis erhalten bleiben. Dies unterstützt

Lernprozesse und spiegelt sich auch in einer schnelleren Einarbeitungszeit von neuen Teammitgliedern wieder (Lee, 1997), (Borchers, 2001).

Besonders geeignet für iterative Prozesse

Zu guter Letzt profitieren vor allem iterative und evolutionäre Vorgehensweisen, weil die Dokumentation ausführlicher wird und sich die Gründe für die zuletzt getroffenen Entscheidungen besser nachvollziehen lassen (Carroll, 1991), (Bratthall et al., 2000). Außerdem ist eine Übersicht über mögliche Alternativen vorhanden, was gerade für zukünftige Entwicklungen von Vorteil ist.

Monk et al. (1995) schlagen deshalb vor, dass neben einer Spezifikation immer auch ein *Design Rationale* erstellt und mitgeliefert werden sollte. Dadurch kann die Komplexität des Spezifikationsprozesses verringert werden, was sich in niedrigeren Kosten und einer schnelleren Entwicklung niederschlägt.

4.3 Zusammenfassung

Die Zusammenarbeit von verschiedenen Interessengruppen macht die Spezifikation von Anforderungen zu einer kritischen Phase. Treten Kommunikationsprobleme auf, so kann es passieren, dass Anforderungen falsch interpretiert werden und alle späteren Entscheidungen negativ beeinflusst werden.

Bridging the Gap

Kollaboration und Kooperation sind bei der Erstellung von interdisziplinären Spezifikationen essenziell. In diesem Kapitel wurden Techniken und Konzepte vorgestellt, welche die Verständigung untereinander in besonderem Maße unterstützen. Sie machen es möglich, dass sich Personen, mit unterschiedlichem Wissens- und Kenntnisstand, aktiv an der Spezifikation beteiligen können.

Für das bessere Verständnis müssen Anforderungen dokumentiert und spezifiziert werden. Dies geschieht meist in textueller Form und durch die Ergänzung mit Skizzen, Grafiken oder anderen Artefakten. Textbasierte Dokumente bieten zahlreiche Vorteile, sind allerdings auch mehrdeutig interpretierbar. Zudem entsteht ein Mix an Formaten, da kein einheitliches Werkzeug verwendet wird und deshalb persönliche Interessen bei der Auswahl überwiegen.

Die Möglichkeiten und Techniken des *Agile Modeling* machen die schwer verständlichen und aufwendigen, formalen Beschreibungen leichter zugänglich und durch die reduzierte Modellpalette können alle Akteure partizipieren. Der Spezifikationsprozess wird so leichtgewichtig und einfacher durchzuführen, weshalb Entscheidungen früher erzielt werden können.

Da semiformale Beschreibungen alleine nicht ausreichen, um den Entwurf anderen mitzuteilen, werden zusätzlich Prototypen verwendet. Neben der Spezifikation unterstützen Prototypen auch das Design und die Evaluation, da verschiedene Varianten während des Entwicklungsprozesses verwendet und verglichen werden können. Dadurch sind neue Erkenntnisse und Interpretationen möglich. Zusätzlich kann die Spezifikation schon während des Entwurfs getestet und validiert werden. Interaktive Prototypen erlauben zusätzlich eine genaue Darstellung des *Look & Feel*; die Spezifikation wird erlebbar und das Verhalten kann nachvollzogen werden. Mit der Verwendung von *Design Rationale* werden Ursache und Wirkung, genauer gesagt, die Verbindung zwischen Anforderungen und resultierende Entwurfsentscheidun-

gen dokumentiert. Damit wird die Problematik von sich ändernden Anforderungen gelöst und zusätzlich kommt es zu einer Verbesserung von Kommunikation und Zusammenarbeit zwischen den Beteiligten.

Mit den vorgestellten Techniken ist es möglich, die Kluft zwischen den Interessengruppen zu überbrücken. Missverständnisse bei der Erstellung der Spezifikation können vermieden werden, weil die verwendeten Modellierungsformen leicht verständlich und einfach anzuwenden sind. Der Entwurfsprozess wird effizienter und effektiver, weshalb sich die Kosten verringern und die Qualität der Spezifikation erhöht wird.

Kapitel 5

Vorgehensweise und geeignete Modelle für die interdisziplinäre Spezifikation

In diesem Kapitel wird das Vorgehensmodell, die einzelnen Spezifizierungsphasen sowie die verwendeten Modelle für eine interdisziplinär durchgeführte Spezifikation vorgestellt. Durch die Verwendung der beschreibenen Techniken (siehe Kapitel 4.2—"Techniken zur Realisierung einer interdisziplinären Spezifikation") lässt sich eine Modellpalette zusammenstellen, die für Beteiligte aus allen Disziplinen verständlich und anwendbar ist. Dadurch sind interdisziplinäre Teams in der Lage, eine funktionale Benutzerschnittstelle mit hoher Gebrauchstauglichkeit zu entwerfen, die dem Anwender eine effektive und effiziente Aufgabenerfüllung ermöglicht.

5.1 Vorgehensweise während der Spezifizierung

Benutzerorientierte Vorgehensweise

Da der Entwicklungsprozess für interaktive Systeme (ISO/IEC, 1999) (siehe Kapitel 2.2.1—"User-Centered Design") sehr universell und allgemeingültig ist, kann er als Vorgehensweise für die Spezifikation von interaktiven Systemen angewandt werden (siehe Abb. 5.1). Dazu ist es lediglich nötig, die einzelnen Aktivitäten sowie die zugrunde liegenden Prinzipien der vier Entwicklungsstufen anzupassen. Dadurch kann sichergestellt werden, dass das zu entwickelnde System den Erwartungen der Benutzer entspricht, benutzbar ist und die späteren Anwender ihre Aufgaben effizient und effektiv ausführen können.

Analysephase

Begonnen wird mit der Analysephase, bei welcher der Nutzungskontext untersucht wird und grundlegende Erkenntnisse über Benutzer sowie deren Aufgaben gewonnen werden. Diese Phase entspricht der bereits erwähnten Anforderungsanalyse in Kapitel 2.4—"Disziplinenübergreifende Zusammenarbeit bei der Anforderungsermittlung".

Konzeptionsphase

Die erhobenen Daten fließen anschließend in die nächste Stufe ein, bei der konzeptionelle Entwürfe gemacht werden, um Lösungsansätze zu finden. Eine nähere Analyse zu Benutzern und deren Aufgaben wird durchgeführt, was zu ersten Roh-

entwürfen der Oberfläche führt. Dabei geht es in erster Linie darum, Ideen und Konzepte zu entwerfen, aus denen später verschiedene Variationen des UIs entstehen.

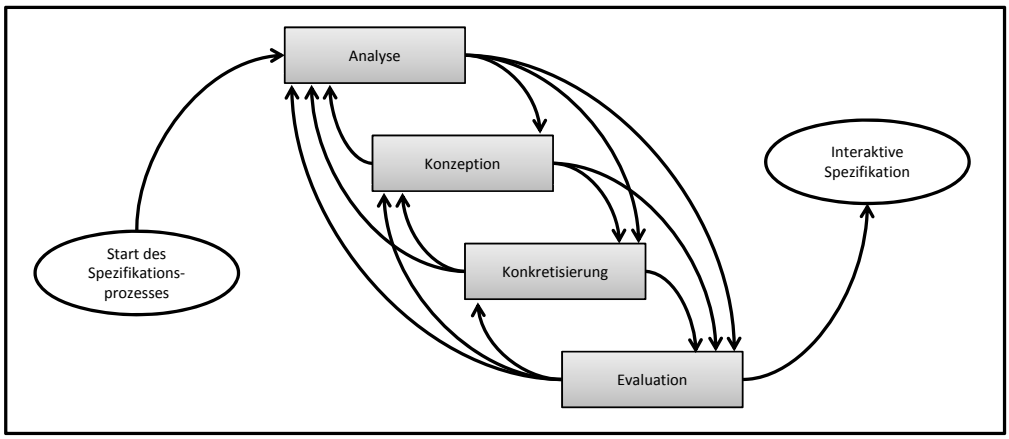


Abbildung 5.1: Vollständig iterativer Spezifikationsprozess in Anlehnung an (ISO/IEC, 1999)

Konkretisierungsphase

Anschließend werden die abstrakten Entwürfe konkretisiert und durch detailliertere Darstellungen ersetzt. Am Ende steht ein interaktiver Prototyp, mit dem es möglich ist, das *Look & Feel* des UIs darzustellen. Durch die Verwendung eines leichtgewichtigen Prozesses, ist es möglich, dass schnelle Iterationen durchgeführt werden, wodurch Rückmeldungen sehr früh erzielt werden können. Da die Reihenfolge der Phasen nicht unbedingt eingehalten werden muss, kann jederzeit zwischen den einzelnen Stufen gewechselt werden, weshalb der Spezifikationsprozess vollständig iterativ ist. Fehlerhafte Annahmen oder Probleme können sofort behoben werden, ohne dass es nötig ist, die Iteration zu Ende zu führen.

Evaluationen

In mehreren Durchgängen kann die Spezifikation verfeinert werden und durch die wiederholt durchgeführten Evaluationen ist gewährleistet, dass mögliche Probleme identifiziert und gelöst werden. Dadurch ist es möglich, das Risiko von Fehlentwicklungen zu vermeiden, wodurch Nachbesserungen und die damit verbundenen hohen Kosten verhindert werden.

5.2 Konsensmodelle und Modellierungsphasen

Konsensmodelle zur Vermeidung von Verständigungsproblemen

Für die Phasen der Konzeption und der Konkretisierung innerhalb der gerade vorgestellten Vorgehensweise werden Modelle benötigt, mit welchen die Verständigungsprobleme zwischen den einzelnen Disziplinen (siehe Kapitel 2—"Interessengruppen und Vorgehensweisen") überbrückt werden können. Gemäß den Erkenntnissen, welche von Bailey et al. (2001) gewonnen wurden, gibt es in diesem Prozess keinen Hauptverantwortlichen, da die Spezifikation gemeinschaftlich erstellt wird und alle Akteure wichtig sind. Modelle müssen gefunden werden, welche einen gemeinsamen Konsens erlauben, sodass alle Beteiligten in der Lage sind, ihre Anregungen, Vorstellungen und Beanstandungen mitzuteilen. Geeignete formale und informale Modellierungsarten werden dabei mit den Prinzipien des *Agile Modeling* verknüpft und dienen zusammen mit abstrakten und detaillierten

Prototypen als interaktive, erlebbare Spezifikation (siehe Kapitel 4.2—"Techniken zur Realisierung einer interdisziplinären Spezifikation").

Identifikation adäquater Modelle

Es wurden Modelle identifiziert, welche ähnliche Zeichen und Symbole verwenden und deshalb nach den Prinzipien des *Agile Modeling* durch den Einsatz einer einzigen Notation ausgedrückt werden können. Weil dies nicht überall möglich ist, kann es durchaus sein, dass manche Akteure mit Modellen arbeiten müssen, die ihnen nicht vertraut sind. Diese Problematik wird ebenfalls durch die Möglichkeiten des *Agile Modeling* entschärft, da Modelle sehr viel einfacher zu verstehen und zu benutzen sind, wenn ihre agile Version Verwendung findet (Memmel und Reiterer, 2008b). Zudem war wichtig, dass Anknüpfungspunkte zwischen den Modellen hergestellt werden kann, sodass es möglich ist, Beziehungen herzustellen.

Überschaubare Modellpalette

Außerdem wurde bei der Auswahl geeigneter Modelle darauf geachtet, dass die Empfehlungen von Constantine et al. (2003) Berücksichtigung finden, und die Modellpalette nur einige wenige Modelle mit möglichst geringer Komplexität beinhaltet. Schwer verständliche, komplexe Modelle werden nicht einbezogen. Darüber hinaus wird Entwurf von Implementierung getrennt, weshalb Modelle herausgefiltert werden, deren Nutzen sich in erster Linie erst während der Umsetzung entfaltet (Brooks, 1995). Wichtig ist *"Was"* umgesetzt werden soll und nicht *"Wie"* dies geschieht (Barbosa et al., 2004).

Zugrundeliegene Philosophien

Basierend auf den Konzepten *Usage-centered Design* (Constantine und Lockwood, 1999a), *Participatory Design* (Chin et al., 1997), (Carroll, 2000b), *Agile Modeling* (Ambler und Jeffries, 2002) sowie *Multi-fidelity* bzw. *Mixed-fidelity Prototyping* (Coyette et al., 2007a), (Petrie und Schneider, 2006) wurde eine Einteilung der Modelle in die in Abbildung 5.2 dargestellten Phasen vorgenommen. Diese Phasen werden mehrmals iteriert und liefern Ergebnisse, die von abstrakten textbasierten Beschreibungen zu detaillierten Entwürfen reichen. Begonnen wird mit der Analyse des Problembereiches, wobei unter anderem erste Anforderungen an Benutzer und Aufgaben ermittelt werden. Diese Anforderungen werden dann in den nächsten Phasen konkretisiert und durch die Modellierung der funktionalen Aspekte ergänzt. Dadurch wird es möglich, die Aufgaben aus Benutzer- und Systemsicht darzustellen. Am Ende werden detaillierte Darstellungen der Oberfläche entworfen, welche zur Simulation von Interaktivität geeignet sind. Über mehrere Durchläufe können die Anforderungen immer weiter verfeinert werden und schließlich als Spezifikation des UIs dienen.

Richtlinien für die Auswahl von Modellen

Die im Folgenden beschriebenen Modelle sind das Resultat der Analyse. Aufgrund des Umfangs der Arbeit können allerdings nicht alle untersuchten Modelle aufgeführt werden, weshalb sich die folgenden Kapitel nur mit den Modellen beschäftigt, die als sogenannte Konsensmodelle verwendet werden können. Flexibilität, Generalität, Kompatibilität, Dialogfähigkeit und Allgemeingültigkeit stehen im Vordergrund, wohingegen Präzision, Striktheit und richtige Syntax eine geringere Bedeutung beigemessen wird. Das Resultat ist eine allgemein verständliche Modellpalette, welche während des Spezifikationsprozesses von interdisziplinär zusammengesetzten Gruppen verwendet werden kann und den gemeinsamen Nenner bildet. Zur Veranschulichung der Konsensmodelle und ihrer Anküpfungspunkte werden Modellierungsbeispiele aus dem Kapitel 6—"Interaktive visuelle Spezifikation" verwendet.

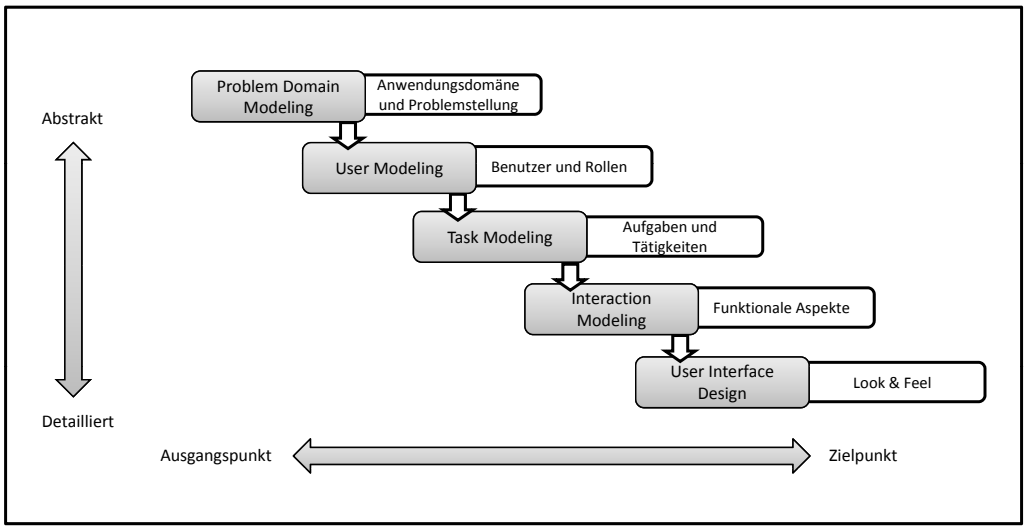


Abbildung 5.2: Phasen der Modellierung und entsprechende Aufgaben

5.2.1 Problem Domain Modeling

Informale Analyse von Problemstellungen

Wie Heinilä et al. (2005) sowie Memmel und Reiterer (2008b) beschreiben, sind *Scenarios* ein gut geeignetes interdisziplinäres Mittel für die Analyse von Problemstellungen. *Scenarios* finden in vielen Vorgehensweisen Verwendung (Paternò, 2001), (Aoyama, 2005) und werden meist zu Beginn der Modellierung verwendet. Sutcliffe (2003) und auch Markopoulos und Marijnissen (2000) stellen fest, dass der Begriff *Scenario* in den unterschiedlichsten Zusammenhängen verwendet wird, weshalb es keine allgemeingültig akzeptierte Definition und abweichende Interpretationen gibt. Deshalb wird die Bedeutung auf die folgende Begriffsbestimmung des *Scenario-based Design* eingeschränkt:

> „Scenarios are stories [...] about people and their activities [...] In scenario-based design, descriptions of how people accomplish tasks are a primary working design representation [...] Scenarios highlight goals suggested by the appearance and behavior of the system, what people try to do with the system, what procedures are adopted, not adopted, carried out successfully or erroneously, and what interpretations people make of what happens to them."
>
> (Carroll, 2000a)

Technik aus dem Participatory Design

Scenarios sind eine weitverbreitete Technik aus dem Bereich des PD (siehe Kapitel 2.2.2—"Participatory Design"), bei der die Bedürfnisse und Belange der späteren Benutzer im Mittelpunkt stehen (Carroll und Rosson, 1990), (Chin et al., 1997). Sie beschreiben in erzählerischer Form Arbeitsabläufe, Aufgaben und Interaktionen des Benutzers mit dem System und werden meist durch kurze natürlichsprachige Geschichten ausgedrückt (Constantine und Lockwood, 1999a), (Carroll, 2000a). Durch die Darstellung von konkreten Sachverhalten können die Ziele des Benutzers, seine Handlungen und der Nutzungskontext anschaulich dargestellt werden. Zudem gewährleistet der informale Charakter, dass Ideen und Konzepte

in verständlicher Weise kommuniziert werden können. Das Auffinden und Evaluieren von Lösungsmöglichkeiten wird einfacher, weil die erzählerische Form Verständnisprobleme unter den Beteiligten beseitigt (Rosson und Carroll, 2002), (Barbosa et al., 2004), (Lim und Sato, 2006), (Anggreeni, 2008). Carroll (2000a) führt die folgenden fünf Gründe an, welche für die Verwendung von *Scenarios* sprechen:

- Scenarios stärken die Reflexion bei kreativen Tätigkeiten, weshalb sie die Erzeugung von Ideen und deren Bewertung unterstützen.
- Scenarios sind gleichzeitig konkret und flexibel, wodurch sie Entwickler zur Bewältigung von mehrdeutigen und veränderlichen Situationen befähigen.
- Scenarios ermöglichen mehrere Sichtweisen mit unterschiedlichem Detaillierungsgrad auf Interaktionen und ihre Auswirkungen.
- Scenarios lassen sich abstrahieren und kategorisieren.
- Scenarios unterstützen die arbeitsorientierte Kommunikation unter allen Beteiligten, wodurch die Entwicklung zugänglicher wird.

Ähnliche Ansätze sind in allen Disziplinen vorhanden

Natürlichsprachige Beschreibungen für frühe Entwurfsphasen werden in allen Disziplinen verwendet. So werden *Activity*, *Information* oder *Interaction Scenarios* bei HCI dazu verwendet, um den Kontext der Entwicklung, Benutzer, Rollen, Interaktionen oder Aufgaben abzustecken. *Usage Scenarios* (siehe Abb. 5.3) sind bei SE bekannt und beschreiben ebenfalls Interaktionen zwischen Benutzer und System (Jarke, 1999). Daneben werden sie auch zur Spezifikation und Validierung von Anforderungen benutzt (Sutcliffe, 2003), (Araujo et al., 2004). Beim BE gibt es durch die Business Vision ebenfalls eine verwandte Notation, welche zur Beschreibung von Zielsetzungen verwendet wird und Annahmen über Benutzer bzw. Kunden, deren Absicht und Interessen beinhaltet (Castela et al., 2001), (Memmel und Reiterer, 2008b).

Emily Shawn is looking for a shirt.
Emily sees a nice blue one.
Emily puts the blue shirt in her shopping cart.
Emily successfully logs into the member area using her personal account information.
Emily pays the bill of 25 € for the shirt with her credit card.
Emily logs off.

Abbildung 5.3: Ein Usage Scenario zur Beschreibung von Interaktionen während eines Kaufvorgangs in einem Onlineshop

Verbesserung der Zusammenarbeit

Scenarios verbessern die Kommunikation zwischen den Interessengruppen und machen den Entwurfsprozess zugänglicher (Lim und Sato, 2006), (Obendorf und Finck, 2008). Sie können zur Beschreibung der Anwendungsdomäne und der Problematik verwendet werden, aber auch zur Beschreibung von Benutzern, Interaktionen und Aufgaben verwendet werden. Sie sind deshalb ein gut geeigneter Einstieg in die interdisziplinäre Modellierung und können als gemeinsamer Nenner dienen. Durch die informale Form, ihren Dokumentationscharakter, ihre unkomplizierte Anwendung sowie die leichte Verständlichkeit ist es darüber hinaus ebenfalls möglich, *Scenarios* als *Design Rationale* zu verwenden (Sutcliffe, 2003), (Carroll, 2000a) oder sie bei der Evaluation einzusetzen (Kwon et al., 2007).

5.2.2 User Modeling

Benutzer als zentrale Rolle

Da Benutzer eine zentrale Rolle bei der Entwicklung von Softwaresystemen spielen (siehe Kapitel 2.2—"Mensch-Computer-Interaktion"), ist es schon beim Entwurf nötig, darauf zu achten, dass Bezüge zu den späteren Anwendern hergestellt werden. Es ist deshalb wichtig, zu wissen, wer die zukünftigen Bediener des Systems sind und welche Bedürfnisse und Erwartungen sie haben. Durch die Verwendung von *User Models* können die verschiedenen Personen beschrieben werden, die mit dem System interagieren (Kobsa, 1995), (Sousa und Furtado, 2003). Daneben ist es möglich ihre Charakteristika, Eigenschaften, Rollen und Interessen aufzuführen (Wilson et al., 1993), (Bouillon et al., 2002). Dadurch wird die Erstellung von personalisierten Oberflächen möglich, die individuell auf die Bedürfnisse der späteren Anwender (aber auch Gruppen von Anwendern) ausgelegt sind (Schlungbaum, 1997), (Lozano et al., 2002). Das Resultat ist eine gesteigerte Gebrauchstauglichkeit des Systems, wodurch sich Aufgaben effektiver und effizienter ausführen lassen.

> „User modeling will ensure that user interactions match user needs and expectations, users' existing knowledge and experience, user goals and tasks, users' physical attributes, cultural factors, and users' attitudes to the system."
>
> (Adikari et al., 2006)

User Roles zur Kategorisierung von Benutzern

Eine weit verbreitete Technik, die zur Modellierung der Benutzer eingesetzt werden kann, sind *User Roles* (Constantine und Lockwood, 1999a). Ursprünglich wurden sie im *Usage-Centered Design* (siehe Kapitel 3.1.6—"Usage-Centered Software Engineering") zur Generalisierung von Benutzern verwendet. Benutzer, welche in gleicher oder ähnlicher Weise mit dem System interagieren, werden gruppiert und so weit vereinfacht, dass sie auf abstraktere Rollen reduziert werden können (Constantine, 1995). Dabei ist ein Benutzer nicht auf eine einzelne Rolle beschränkt, sondern kann durchaus mehrere Rollen gegenüber dem System innehaben. Im Gegenzug ist es natürlich auch möglich, dass Rollen geteilt werden.

> „The characteristics of the role, the relationship to the system, has a more immediate and direct relevance for interaction design than do characteristics of the person playing the role. "
>
> (Constantine, 2006a)

Vermittlung von spezifischen Anforderungen

User Roles repräsentieren stereotypische Urbilder der späteren Nutzer und können zur Darstellung von spezifischen Interessen und Erwartungen dieser Nutzergruppe an das System verwendet werden. So wird es möglich, gemeinsame Charakteristika wie etwa Kenntnisse, Verhaltensweisen und Verantwortlichkeiten zu identifizieren und auszudrücken (Nunes, 2001), (Hill und Bartek, 2007).

Darstellung durch Role Maps

Zur Darstellung von *User Roles* können *Role Maps* (siehe Abb. 5.4 (1)) verwendet werden, welche von der Darstellung mit den von der UML bekannten *Use Case* Diagrammen übereinstimmen (Nunes, 2001). Diese *User Role Map* ist eine einfache Möglichkeit, die einzelnen Rollen zu strukturieren, wodurch ein Überblick über die Unterschiede sowie die Verbindungen untereinander erreicht werden kann (Constantine und Lockwood, 1999a).

Personas

Zur besseren Charakterisierung von besonders wichtigen *User Roles* kann eine Verfeinerung durch *Personas* vorgenommen werden (Cohn, 2004), (Beyer und Holtzblatt, 1997), (Schlungbaum, 1997) (siehe Abb. 5.4 (2)). Personas sind hypothetische Repräsentanten einer Rolle und wurden erstmals von Cooper (2004) eingeführt. Basierend auf den Daten aus der Anforderungsanalyse, welche z.B. durch Interviews oder Beobachtungen gewonnen wurden, werden imaginäre Benutzer erschaffen. Jede *Persona* wird mit Namen, Bild, Bedürfnissen, Zielen und Aufgaben individualisiert und in detaillierter Form dargestellt (Blomquist und Arvola, 2002).

> „Identifying user roles is a great leap forward, but for some of the more important user roles, it might be worth going one step further and creating a persona for the role. A persona is an imaginary representation of a user role [...with, d.V. ...] a name, a face, and enough relevant details to make them seem real to the project members."
>
> (Cohn, 2004)

Imaginäre Benutzer

Cooper (2004) lehnt den Begriff des „Benutzers" ab, da er nicht spezifisch genug ist. Anstatt sich auf eine Liste mit Anforderungen und einen allgemeingültig definierten Benutzer zu beziehen, ist es seiner Ansicht nach besser, konkrete Individuuen als Gruppenrepräsentanten zu erzeugen, auch wenn diese zuvor selbst erstellt wurden. Da sich Entwickler stärker mit den selbst erstellten Benutzern identifizieren können, wird der Wechsel in die Sicht- und Denkweise der Anwender leichter. Constantine (2006b) drückt dies mit der kurzen und prägnanten Äußerung aus: *"Personas are fun."*

> „The persona description includes the typical tasks and problems of the user, identifies the assistance he receives from his co-workers or collaborators (who, when, why), and the tools he uses."
>
> (Hill und Bartek, 2007)

User Roles und Personas ergänzen sich

Durch die Verwendung von abstrakten Benutzergruppen im Zusammenspiel mit konkreten Benutzern lassen sich Anforderungen an das System anschaulich darstellen. *User Roles* beschreiben die Verbindungen zwischen Benutzern und dem System, während *Personas* sehr detailliert auf den einzelnen Anwender eingehen (Constantine, 2006b) (siehe Abb. 5.4 (3)). Wie Hill und Bartek (2007) darstellen, sind sowohl *User Roles* als auch *Personas* wichtige Techniken, welche sich ergänzen und bei der Bestimmung der Zielgruppe für das spätere Produkt sehr hilfreich sind. Durch ihre Verwendung wird der Interpretationsspielraum geringer und die Ziele und Bedürfnisse der späteren Benutzer können leichter nachvollzogen und kommuniziert werden. Wie Constantine (2006b), Hanna (2005), Memmel und Reiterer (2008b) sowie Memmel et al. (2008a) zeigen, gibt es ähnliche Notationen bei allen Disziplinen, weshalb *User Roles* und die informalen *Personas* zur interdisziplinären Modellierung von Benutzern und ihren Bedürfnissen geeignet sind, da sie von allen leicht nachvollzogen werden können.

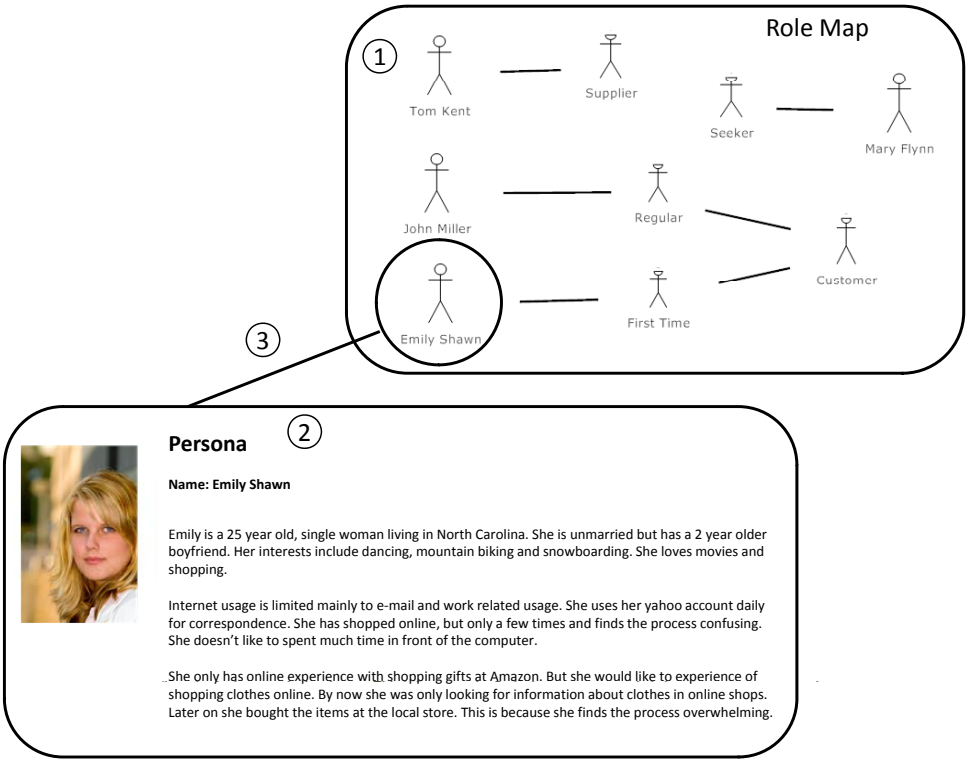


Abbildung 5.4: Zusammenhang zwischen Role Maps und Persona

5.2.3 Task Modeling

Kenntnisse der Aufgaben wichtig

Beim Entwurf von Oberflächen ist neben dem genauen Verständnis der Nutzer genauso wichtig zu wissen, welche Aufgaben mithilfe des UIs bzw. des Systems erledigen werden sollen (Paternò, 2001), (O'Neill und Johnson, 2004), (Constantine, 2006b), (Bowen und Reeves, 2007). Durch die Verwendung von *Task Models* können die Tätigkeiten und Aufgaben genau definiert werden, welche vom Benutzer ausgeübt werden müssen, um ein konkretes Ziel zu erreichen (Lozano et al., 2002). *Task Models* sind eine weitverbreitete Technik und sehr gut zur Ergänzung von *User Mo-*

dels geeignet (Constantine, 1995), (Bouillon et al., 2002), (Sousa und Furtado, 2005).

> „The Task Model defines the ordered set of activities and actions the user has to perform to achieve a concrete purpose or goal."
>
> (Lozano et al., 2002)

Essential Use Cases zur Modellierung von Aufgaben aus Benutzersicht

Zur Modellierung der Aufgaben aus Benutzersicht kann das Konzept der *Essential Use Cases* aus dem Usage-Centered Design (siehe Kapitel 3.1.6—"Usage-Centered Software Engineering") Verwendung finden (Constantine und Lockwood, 1999a). Die grundlegende Idee dahinter ist, einen Anwendungsfall zwischen Benutzer und System aus den vorhandenen *User Roles* abzuleiten (Constantine, 1995), (Nunes, 2001). Wie Constantine (2006b) darstellt, werden diese *Use Cases* durch Generalisierung und Vereinfachung so weit abstrahiert, dass die Interaktion zwischen Benutzer und System auf das Notwendigste reduziert werden kann. Dabei stehen nicht die Aktionen und Reaktionen zwischen Benutzer und System im Vordergrund, sondern das Vorhaben des Anwenders aus Sicht der Rolle und die entsprechenden Verantwortlichkeiten des Systems ihm gegenüber (Constantine und Lockwood, 1999b). Dadurch wird keine Aussage über die technische Realisierung vorweggenommen, weshalb die Möglichkeiten bei der Umsetzung nicht im Voraus beschränkt werden und kreativere Lösungen realisierbar sind. Zudem ist es möglich einfachere und leichter verständliche UIs zu erstellen – im Idealfall benötigt eine gut entworfene Oberfläche nur die im *Essential Use Cases* festgehaltenen Möglichkeiten der Interaktion (Constantine, 1995).

Intentionen und Verantwortlichkeiten

> [An essential use case, d.V.] „ is one expressed in abstract, simplified, and generalized form independent of explicit or implied assumptions about technology or implementation. Instead of user actions and system responses, essential use cases represent user intentions and system responsibilities."
>
> (Constantine, 2006b)

Informale und leicht verständliche Darstellungsform

Essential Use Cases – auch *Task Cases* (Constantine et al., 2003) oder *Business Use Cases* (Ambler, 2006) – können auf die von Rumbaugh et al. (1998) vorgestellten *Use Cases* zurückgeführt werden und werden in tabellarischer Form festgehalten (siehe Abb. 5.5 (1)). Durch ihre informale, textuelle Form und ihre einfache Struktur sind sie selbsterklärend. Dadurch ist es einfacher, die Absichten sowie die Intentionen des Benutzers in seiner Rolle nachzuvollziehen, wodurch die Oberfläche genau auf den Anwender zugeschnitten werden kann.

> „Essential use cases are particularly effective as a medium for communicating with users about their work and the system requirements to support that work users do not need to learn a new language, an obscure notation, or a specialized set of modeling conventions to understand essential use case narratives."
>
> (Constantine, 1995)

Use Case Diagramme zur Modellierung von Aufgaben aus Systemsicht

Zur Beschreibung der Interaktionen zwischen Benutzer und System ist es ebenso wichtig, die systeminternen Vorgänge zu beschreiben. Eine weitverbreitete Darstellungsform für diesen Zweck ist das in der UML verwendete *Use Case* Diagramm (Rumbaugh et al., 1998), (Ambler, 2006), (Jacobson, 2004), (da Silva und Paton, 2003), (Oestereich et al., 2003), (Holt, 2005), das verwandt mit dem bereits vorgestellten *Scenario* ist. Ein *Use Case* ist aber eher generell gültig, wohingegen ein *Scenario* konkrete Sachverhalte zu Benutzern und ihren Aufgaben ausdrücken kann.

> „The use case diagram is useful for task modeling although there is no clear view on the differences between a use case and a scenario. One definition could be that a use case describes a specific „path“ through a task tree under specified conditions. A scenario can then be defined as a more general description that also sets the context for a use case.“
>
> (van Welie, 2001)

Aktionen und Reaktionen

Bei dieser Form des *Use Cases* wird weniger Wert auf die Darstellung von Intentionen und Verantwortlichkeiten gelegt sondern vielmehr auf konkrete Aktionen und Reaktionen. *Use Case* Diagramme werden deshalb dazu verwendet, Aktions- und Reaktionssequenzen zu modellieren, welche für die Interaktion des Systems mit externen Instanzen vorhanden sein müssen (Rumbaugh et al., 1998). Diese Instanzen oder Akteure können dabei sowohl Benutzer als auch andere Systeme sein. Dadurch ist es möglich, das System genau abzugrenzen, funktionale Anforderungen und Schnittstellen zu definieren sowie alle Instanzen zu identifizieren, welche mit dem System interagieren (Williams, 2003), (Phillips und Joe, 2005), (Kholkar et al., 2005), (Almendros-Jimenez und Iribarne, 2005). Miller (2006) identifiziert die folgenden drei Verwendungsarten für *Use Case* Diagramme:

- Durch die Verwendung von Use Cases lassen sich neue Anforderungen und notwendige Funktionalitäten ermitteln, weil eine Analyse vorgenommen wird und sich dabei Ideen herausbilden.
- Durch ihre einfache Darstellung sind *Use Case* Diagramme eine gute Möglichkeit zur Verbesserung der Kommunikation.
- Durch die Erzeugung von *Use Case* Diagrammen lassen sich Testfälle für die spätere Verifikation des umgesetzten Systems erstellen.

Einfache Darstellung

Wie van Welie (2001) zeigt, ist die Darstellung eines *Use Case* Diagrammes (siehe Abb. 5.5 (2)) eher informal und auf einen reduzierten Symbolsatz beschränkt – Akteure, Use Cases sowie Verbindungslinien bzw. -pfeile um die Beziehungen zwischen den einzelnen *Use Case* und den Akteuren darzustellen.

Task Maps zur Strukturierung von Essential Use Cases

Constantine und Lockwood (1999a) stellen dar, dass *Task Maps* (bzw. *Use Case Maps*) zur Strukturierung von *Essential Use Cases* verwendet werden können (siehe Abb. 5.5 (3)). Da diese *Task Maps* durch Zeichen und Symbole ausgedrückt werden, welche in etwa der UML Notation von *Use Case* Diagrammen entsprechen, sind Anknüpfungspunkte vorhanden. Hinzu kommt, dass es ebenfalls Ansätze gibt, bei

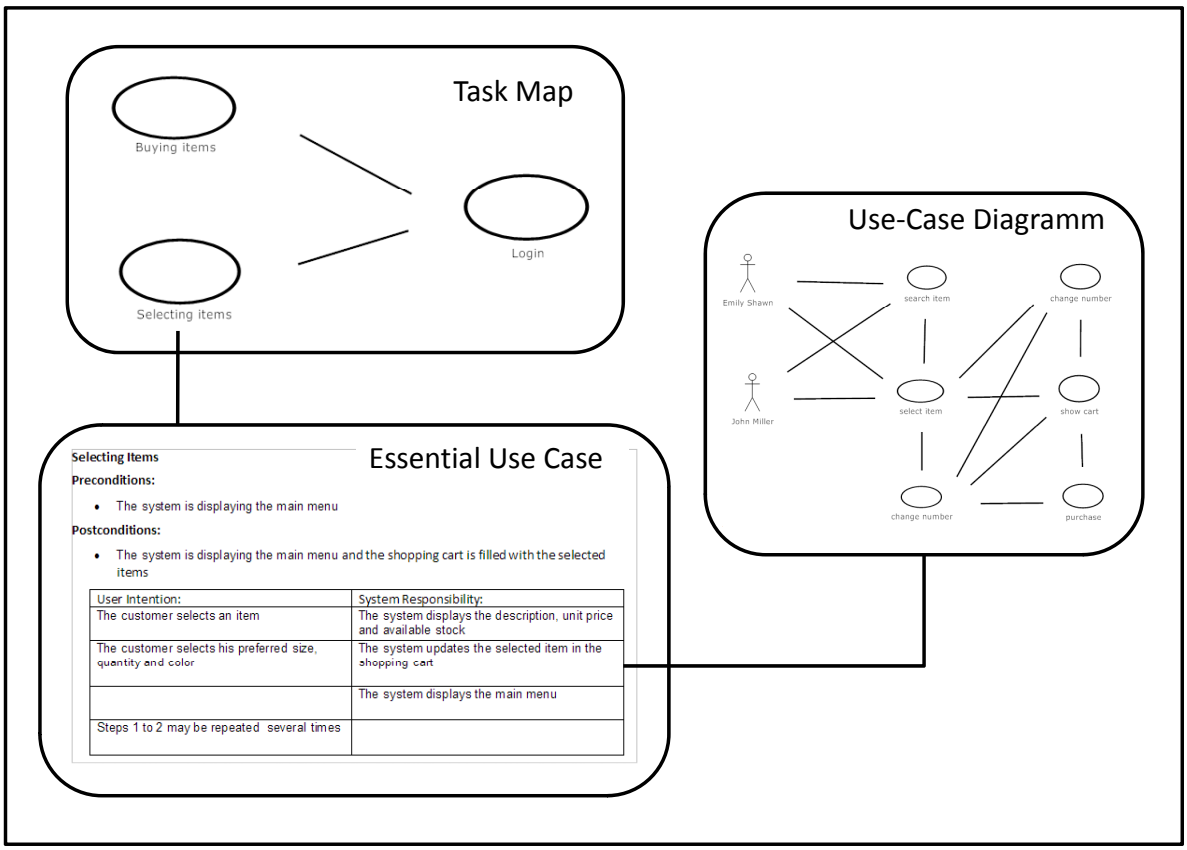


Abbildung 5.5: Zusammenhänge zwischen Task Map, Essential Use Case und Use Case Diagramm

welchen *Use Case* Diagramme erfolgreich zur Geschäftsprozessmodellierung eingesetzt werden (Oestereich et al., 2003), (Holt, 2005). Diese Adaption der *Use Case* Diagramme für BE wird durch die Tatsache begünstigt, dass BPs abstrakte Darstellungen von Business Tasks sind (Artim et al., 1998), (Sousa et al., 2008a) welche durchaus durch die *Use Case* Notation der UML dargestellt werden können.

> „When use cases are combined with [...essential use cases, d.V. ...] the result is a process that smoothly connects the design of user interface architecture back to the essential purposes of a system and the work it supports."
>
> (Constantine, 1995)

Essential Use Cases und Use Case Diagramme als Konsensmodelle

Essential Use Cases und *Use Case* Diagramme können als Konsensmodelle innerhalb des *Task Modeling* verwendet werden, weil sie in allen Disziplinen bekannt sind. Zwar liegt der Fokus auf verschiedenen unterschiedlichen Aspekten, wie etwa Funktionalität (SE), Prozesse und Arbeitsabläufe (BE) sowie Aufgaben des Benutzers (HCI), grundsätzlich gibt es aber zwischen den verschiedenen Variationen der Aufgabenmodellierung Überschneidungen und Beziehungen (siehe Abb. 5.5 (4)), wodurch ein gemeinsames Verständnis geschaffen werden kann (Memmel und Reiterer, 2008a).

5.2.4 Interaction Modeling

Spezifikation des dynamischen Verhaltens

Für die weitere Verfeinerung der im *Task Modeling* getroffenen Enschlüsse werden Ausdrucksweisen benötigt, mit denen sich das dynamische Verhalten des Systems detaillierter abbilden lässt. Die Funktionsweise sowie die interne Logik, welche für die Interaktion des Systems mit externen Akteuren benötigt wird, muss genauer dargestellt werden, sodass die Spezifikation nicht Gefahr läuft, lediglich als *"Black Box"* zu fungieren.

> „[...] resulting effects of certain tasks may influence the procedures for other tasks (possibly with other roles involved). Therefore, we will also need to understand task flow and data flow over time as well as the relation between several concurrent flows. "
>
> (van der Veer und van Welie, 2000)

Flow Chart und Activity Diagramme

Flow Charts sind eine Möglichkeit die Interaktionen und Abläufe zwischen Benutzer und System genauer zu beschreiben (Dix, 1995a) und eignen sich durch ihre informale Struktur gut, um Vorgänge gegenüber Benutzern darzustellen (Cook und Bailey, 2005). Zur besseren Darstellung von dynamischen Verhalten innerhalb eines Vorgangs können *Activity* Diagramme, eine Sonderform des *Flow Chart* Diagramms, verwendet werden (Xia, 2005), (Miller, 2006) (siehe Abb. 5.6). Neben der Darstellung von Abhängigkeiten, Abläufen und Ereignissen kann das *Activity* Diagramm auch durchaus während der Anforderungsanalyse benutzt werden (Nunes und Cunha, 2000b), (Phillips und Kemp, 2002), (Xia, 2005). Entscheidungen, die zu optionalen Vorgängen führen, lassen sich ebenso ausdrücken wie parallel ablaufende Geschehnisse (van der Veer und van Welie, 2000).

> „Activity diagrams capture sequencing information about the task quite clearly and succinctly, allowing for the specification of unordered or concurrent activities."
>
> (Markopoulos und Marijnissen, 2000)

Disziplinenübergreifendes Anwendungsgebiet

Activity Diagramme werden in einer Vielzahl von Vorgehensweisen zur Spezifikation von UIs eingesetzt (siehe Kapitel 3.1—"Brückenschläge zwischen den Disziplinen"), und finden selbst beim BE Verwendung. So können sie eingesetzt werden, um die Logik sowie die Abfolgen von Tätigkeiten oder Aktionen innerhalb eines Geschäftsprozesses zu erläutern (Ambler und Jeffries, 2002), (Holt, 2005).

Sequence Diagramme

Wie Ambler (2006) darstellt, sind *Sequence* Diagramme das weitverbreitetste UML-Diagramm zur Modellierung von dynamischen Sachverhalten, logischen Abläufen und zur Darstellungen von Aktionen und Reaktionen, welche bei der Interaktion zwischen Benutzer und System auftreten. Durch ihre Verwendung ist es möglich, logische Vorgänge innerhalb von *Use Cases* genau zu strukturieren, weshalb Verhalten dokumentiert und validiert werden können.

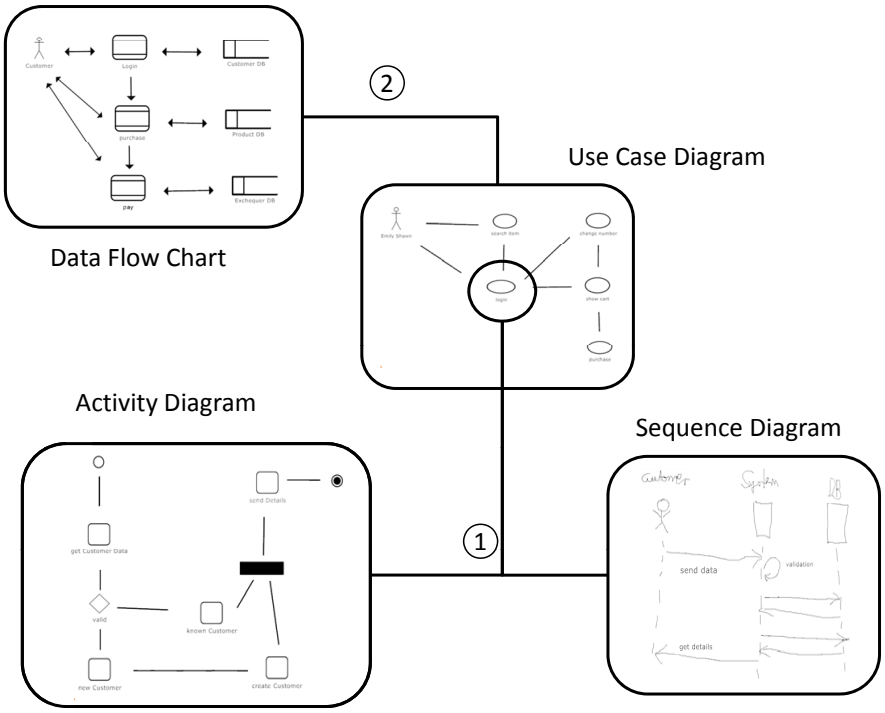


Abbildung 5.6: Zusammenhänge zwischen Activity Diagramm, Sequence Diagramm, Data Flow Chart und Use Cases

Zeitliche Strukturierung von Abläufen

Durch *Sequence* Diagramme können Abläufe zeitlich genau strukturiert werden, was besonders praktikabel für die Veranschaulichung des Informationsaustauschs zwischen System und involvierten Personen bzw. Rollen ist, wie van der Veer und van Welie (2000), van Welie (2001) sowie Berenbach (2004) zeigen. Dadurch ist es möglich, zeitliche Abhängigkeiten innerhalb von Tätigkeiten zu erkennen, weshalb interne Abläufe des Systems eindeutig festgelegt werden können.

> „[The Sequence Diagram, d.V.] realizes a behavioural aspect of the overall model and is used to model high-level behaviour. The sequence diagram is an excellent diagram for tying different views of the system together and forms the basis of the process validation of the process metamodel."
>
> (Holt, 2005)

Data Flow Diagramme

Zur Modellierung des Informationsflusses reicht eine sequenzielle Darstellung nicht aus, weshalb *Data Flow* Diagramme, eine weitere Abwandlung des *Flow Charts*, Verwendung finden. Datenströme innerhalb eines *Task Models* (1) lassen sich darstellen, wodurch die Herkunft, die Verarbeitung sowie die Weiterleitung von Daten innerhalb von Arbeitsabläufen anschaulich beschrieben werden kann (Ambler, 2002).

Weitverbreiteter Ansatz

Dass *Sequence* Diagramme und *Activity* Diagramme zur näheren Bestimmung von *Task Models* (2) während der Spezifikation von UIs eingesetzt werden können, zeigt neben Tick (2005) auch Trætteberg (2002a), van der Veer und van Welie (2000) sowie Markopoulos und Marijnissen (2000) (vgl. auch Kapitel 3.1—"Brückenschläge zwischen den Disziplinen"). Holt (2005) verwendet zur Geschäftsprozessmodellierung

ebenfalls beide Diagrammarten, weshalb von einem breiten Anwendungsgebiet und einem gemeinsamen Verständnis der Disziplinen für diese Notationsformen ausgegangen werden kann.

Data Flow, Activity und Sequence Diagramme als Konsensmodelle

Für die Darstellung von *Flow Charts* werden nur drei grundlegende Symbole benötigt und da *Activity*, *Sequence* sowie *Data Flow* Diagramme ebenfalls nur aus einem leicht verständlichen Symbol- und Zeichensatz zusammengesetzt sind, ist es möglich, dass *Flow Charts*, *Activity* und *Sequence* Diagramme als Konsensmodelle eingesetzt werden können. Sie dienen während des Interaction Modeling zur näheren Beschreibung von den durch die *Task Models* definierten Vorgängen.

Storyboards

Für den Übergang von Modell- auf die konkrete Darstellungsebene, bei welcher von der funktionalen Sichtweise in die eigentliche Darstellungsschicht gewechselt wird, empfiehlt sich die Verwendung von *Storyboards*. Sie werden zur Veranschaulichung von dynamischem Verhalten verwendet und ermöglichen dadurch die Verknüpfung von Verhaltensmodellen mit den Komponenten oder *Widgets* der Oberfläche (Markopoulos und Marijnissen, 2000). Details können vernachlässigt werden, weshalb die Gefahr gering ist, sich in Einzelheiten zu verlieren. Wichtiger ist es, die Interaktionsmöglichkeiten zwischen Benutzer und UI zu veranschaulichen, die zur Lösung, der im *Task Modeling* beschreibenen Aufgaben führen (Ambler, 2006).

Daneben kann ein Überblick über den späteren Aufbau des UIs hergestellt werden, da sich die Anordnung von Dialogfenstern darstellen lässt. Zudem wird sichtbar, wie sich die Zustände des UIs verändern, während der Benutzer auf die gewünschte Funktionalität zugreift (Lozano et al., 2002).

> „Even for very simple scenarios, the modelling of a part of the UI presentation is essential. At this stage we do not need a detailed model of the UI presentation, but only to know what kind of components compose the UI, how many components there are, and how they may be grouped. We also need to know which operations these UI elements should have. Therefore, we need an abstract presentation model."
>
> (da Silva und Paton, 2000b)

Storyboards sind für die UI-Entwicklung unverzichtbar

In einer Studie von Bailey et al. (2001), bei welcher professionelle Entwickler von Multimedia-Applikationen befragt wurden, zeigte sich, dass *Storyboards* als unerlässlich für den erfolgreichen Entwurf eines interaktiven Systems angesehen werden und von nahezu jedem Befragten während des Entwicklungsprozesses eingesetzt werden.

Einfache Umsetzung

Zur Realisierung von *Storyboards* können z.B. einfache Skizzen oder Grafiken verwendet werden. Im Grunde genommen handelt es sich bei einem *Storyboard* ebenfalls um ein *Flow Chart* Diagramm. Jede Grafik zeigt das UI in einem bestimmten Zustand und mithilfe von Verbindungen können Zustandswechsel und sequenzielle Änderungen des UI dargestellt werden. So kann spezifiziert werden, was beim Auslösen eines *Widgets* geschieht, wie das UI aufgebaut ist und welche Navigationsoptionen vorhanden sind (Heinilä et al., 2005). Anmerkungen an den Verbindungen erlauben die Zuordnung zu Interaktionen des Benutzers (Zenka und Slavik, 2004), wodurch interaktives Verhalten simuliert werden kann. Zudem können sie bereits in

frühen Phasen eingesetzt werden, um das Verhalten des UIs zu evaluieren (Heinilä et al., 2005).

Storyboards können einfach erstellt werden und sind trotzdem sehr ausdrucksstark, weshalb sie kommunikative Prozesse bei der Verifizierung von Anforderungen und die weitere Ideenfindung sehr gut unterstützen. Trotz der gegebenen Abstraktheit kann nachvollzogen werden, wie das UI funktioniert, welche Schritte überarbeitet werden sollten und welche sinnvoll sind. Änderungen können durch geringen Aufwand in Echtzeit vorgenommen werden, ohne dass sich die Entwickler Gedanken über die spätere Implementierung machen müssen (Landay und Myers, 2001).

> „By avoiding an early commitment and raising user's expectations about the interface solution that will be adopted, both designers and users will be open to explore alternative solutions at later stages."
>
> (Bailey et al., 2001)

5.2.5 User Interface Modeling

Prototypen zur Modellierung von Struktur, Gestaltung und Verhalten des UIs

Da interaktives Verhalten mit *Storyboards* (siehe Abb. 5.7 (1)) nur rudimentär verwirklicht werden kann (Heinilä et al., 2005), ist es besonders in späteren Phasen wichtig eine detailliertere Darstellungsform für die Spezifikation zu benutzen. Für die Kommunikation des späteren *Look & Feels* ist es deshalb wichtig, dass Darstellungsformen verwendet werden, die den Aufbau, die Funktionalität und die Gestaltung des UI gut wiedergeben können. Die Verwendung von Prototypen mit unterschiedlicher Genauigkeit ist deshalb ein gut geeignetes Mittel für die Phase des *User Interface Modeling* (siehe Kapitel 4.2.2—"Modellierung mit Prototypen"). Begonnen wird mit einem abstrakten *low-fidelity* Prototypen (2), wodurch gewährleistet wird, dass keine zu frühe Festlegung auf ein besonderes Aussehen erfolgt und kreative Lösungen weiterhin möglich sind (Constantine et al., 2003). Durch iterative Verbesserungen und regelmäßige Evaluationen kann dieser so weit verfeinert werden, dass sich Ideen verfestigen und sich eine Umsetzungsmöglichkeit abzeichnet.

Durch die Verwendung von *high-fidelity* Prototypen (3) wird die Benutzerschnittstelle im Idealfall erlebbar, wodurch Interaktionsverhalten, Ästhethik und Strukturierung eindeutig nachvollzogen werden können. Das *Look & Feel* des UIs kann genau festgelegt werden, da aber noch keine Aussage über die spätere Realisierung getroffen worden ist, bleibt die Umsetzung nicht auf eine bestimmte Technologie beschränkt (Paternò, 2001).

> „Prototypes serve as a common language between stakeholders, offering a way for designers to explore design ideas and elicit feedback from stakeholders prior to committing to designs. Prototypes aid in refining requirements and may be used as a specification for developers."
>
> (Petrie und Schneider, 2006)

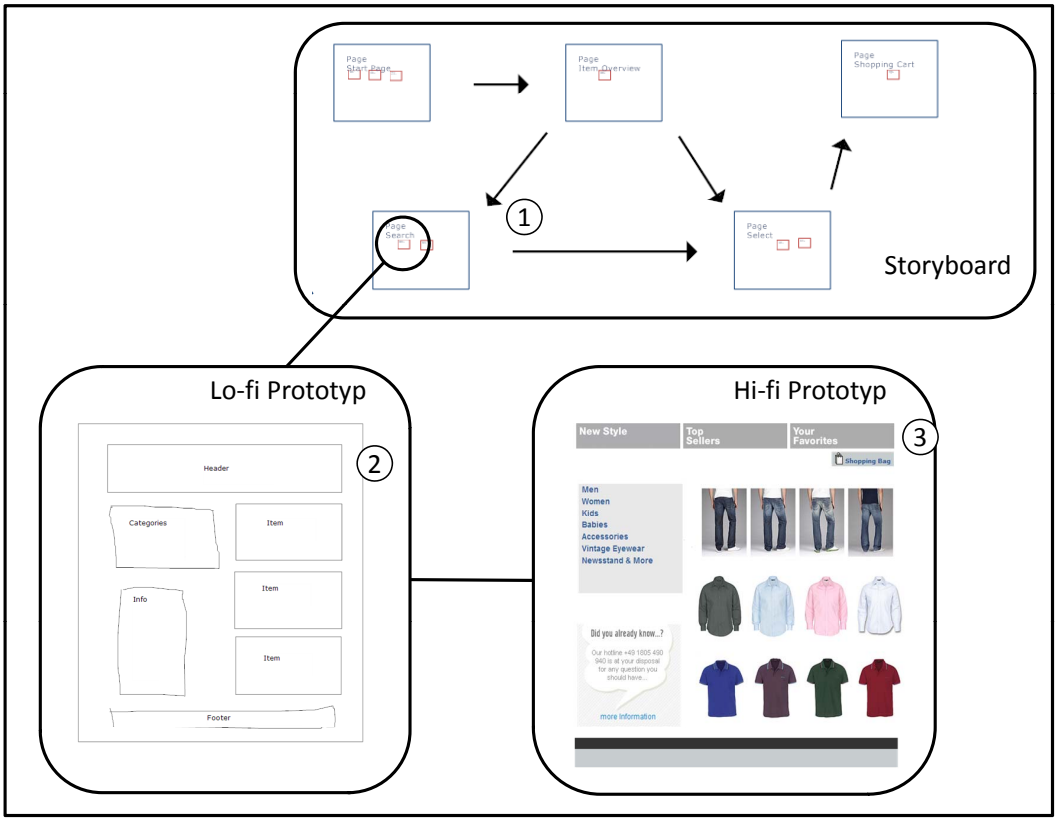


Abbildung 5.7: Zusammenhang von Storyboard und Prototypen mit unterschiedlicher Genauigkeit

Begrenzte Iterationsmöglichkeiten durch herkömmliche Ansätze

Zur besseren Unterstützung von Prototypen mit unterschiedlicher *Fidelity* wird das Prinzip des *Mixed/Multi-fidelity* Prototypen verwendet (Petrie und Schneider, 2006), (Coyette et al., 2007a), (Memmel et al., 2008c). Dabei werden verschiedene Detailstufen in einem Prototyp vereint, wodurch dem Entwicklerteam die Möglichkeit gegeben wird, den Schwerpunkt beim Entwurf auf einzelne Sachverhalte zu legen, wohingegen andere Aspekte zurückgestellt werden.

> „We usually observe that a UI prototype only involves one representation type, i.e. one fidelity level. But due to the variety of stakeholders' input, several fidelities could be imagined together, thus leading to the concept of mixed-fidelity, where several different fidelities are mixed in the same UI design."
>
> (Coyette et al., 2007b)

Nahtloser Übergang zwischen unterschiedlichen Fidelities

Um einen Bruch beim Übergang zwischen Prototypen mit unterschiedlicher *Fidelity* zu vermeiden, ist es besser, die Elemente direkt zu verfeinern, wodurch sich nach und nach aus *low-fidelity* ein *high-fidelity* Prototyp entwickelt. Die Iterationen können dadurch öfters ausgeführt werden und der Entwicklungsverlauf einer Komponente lässt sich leichter nachvollziehen. Nach Petrie und Schneider (2006) ergeben sich folgende Vorteile durch die Verwendung eines *Mixed/Multi-fidelity* Konzepts:

- Komponenten mit unterschiedlichen Genauigkeiten können gleichzeitig in einem einzigen Prototyp verwendet werden.
- Um Konzepte zu verfeinern oder zu überdenken ist es möglich zwischen den Genauigkeiten zu wechseln.
- Domänenspezifische Daten und Funktionalitäten lassen sich zusammenführen.
- Neuartige Interaktionstechnicken können ausfindig gemacht werden.
- Das Vergleichen von mehreren Entwurfsvarianten wird möglich.
- Der Entwurfsprozess wird protokolliert.

Kombination der Vorzüge und Verringerung des early investment/ communication gaps

Dieser *Mixed/Multi-fidelity* Ansatz vereinigt die Vorzüge von *low-fidelity* und *high-fidelity* Prototypen und verbessert die Iterationshäufigkeit, weshalb davon ausgegangen werden kann, dass innovativere Entwürfe entstehen. Zudem wird das von Bailey und Konstan (2003) angesprochene Problem des *early investment/communication gap* umgangen. Sie verweisen darauf, dass *low-fidelity* Prototypen nicht ausreichen, um das Verhalten des UIs zu kommunizieren, wohingegen *high-fidelity* Prototypen nicht für die schnelle Erzeugung von verschiedenen Entwürfen geeignet sind, weil ihre Erstellung zu viel Zeit und Aufwand beansprucht. Durch die Verwendung von *Mixed/Multi-fidelity* Prototypen kann dieses Problem umgangen werden, wodurch das *Look & Feel* des UIs mit akzeptablem Aufwand dargestellt werden kann.

Gemeinsammer Nenner

Durch die Verwendung der vorgestellten Konsensmodelle lassen sich Anwendungsdomäne, Benutzer und Aufgaben modellieren. Darüber hinaus ist es möglich, die funktionalen Anforderungen von Vorgängen zu spezifizieren und Ablauf sowie strukturellen Aufbau des UIs darzustellen, was schließlich in der Erstellung von Prototypen endet. Durch mehrere Iterationen kann dadurch ein UI von den abstrakten Anforderungen über detaillierte Abbildungen bis hin zu interaktiven Simulationen spezifiziert werden. Die vorgestellen Konsensmodelle bilden den gemeinsamen Nenner für interdisziplinäre Teams bei der Spezifikation von UIs, weil für jede Phase geeignete Modelle bereitgestellt werden, die von allen verstanden und anwendbar sind.

5.3 Zusammenfassung

Rahmenmodell für die interaktive Spezifikation

Der gemeinsame Nenner für den interdisziplinären Spezifikationsprozess basiert auf den im vorherigen Kapitel vorgestellten Konsensmodellen. Diese können während der Konzeptions- und Konkretisierungsphase (siehe Kapitel: 5.1—"Vorgehensweise während der Spezifizierung") eingesetzt werden, um die Anforderungen der Benutzer festzuhalten. Das Rahmenwerk dieses Ansatzes ist in Abbildung 5.8 zu sehen. Auf der vertikalen Achse sind die Modelle von textbasierten Darstellungen über semiformale Modelle hin zu detaillierten ausführbaren Prototypen geordnet. Die horizontale Achse symbolisiert den Filtervorgang, welcher durchgeführt wurde, um von den adäquat einsetzbaren Modellen zu den allgemeingültigen Notationen zu gelangen.

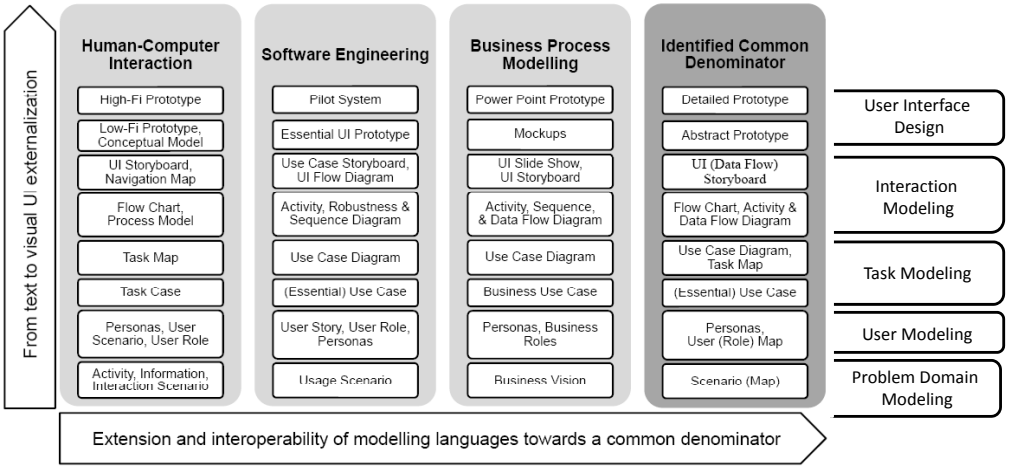


Abbildung 5.8: Der Aufbau der einzelnen Modellierungsphasen und die verwendbaren Konsensmodelle (Memmel und Reiterer, 2008b)

Ablauf der Spezifikation

Begonnen wird mit *Scenarios*, um die Zielsetzung des Entwicklungsprozesses festzulegen (siehe Abb. 5.9). Ferner können erste Aufgaben und mögliche Rollen der späteren Benutzer definiert werden. Benutzer können durch die Verwendung von *Personas*, *User Roles* und *Role Maps* bestimmt werden. Zur besseren Veranschaulichung von Aufgaben aus Benutzersicht werden *Essential Use Cases* verwendet. Funktionale Anforderungen an die Aufgaben können hingegen besser durch *Use Case* Diagramme bzw. eine *Task Map* vorgegeben werden.

> „The application of use case models and task models helps UI designers and system analyst to work together during the definition of requirements, making the validation of functional and usability requirements more agile."
>
> (Sousa und Furtado, 2005)

Von informalen über semiformale zu detaillierten Beschreibungen

Als Schnittstelle von Benutzer- und Aufgabenanalyse zur detaillierten Spezifikation der Oberfläche werden die eher formaleren Notationen aus der UML verwendet und mit Prinzipien des *Agile Modeling* vereinfacht. Das Verhalten des Systems und andere interne Prozesse, wie etwa Datenflüsse oder die Abfolge von Vorgängen kann durch die Verwendung von *Flow Charts*, *Activity* und *Sequence* Diagrammen analysiert und festgehalten werden. Durch die Verwendung von *Storyboards* und Prototypen lassen sich Ideen zum Entwurf realisieren und in für alle *Stakeholder* verständlicher Form festhalten (siehe Kapitel 4.1.4—"Informale Ausdrucksweisen"). Darüber hinaus sind frühe abstrakte Entwürfe zur Lösung von grundsätzlichen Fragen genauso möglich wie detaillierte Präsentationen.

Design Rationale zur besseren Nachverfolgbarkeit von Anforderungen

Sind Designentscheidungen so weit gereift, dass sie spezifiziert werden können, so kann ein interaktiver *high-fidelity* Prototyp erstellt werden, der das *Look & Feel* des UIs verkörpert und dadurch die Anforderungen der Benutzer kommuniziert (siehe Kapitel 4.2.2—"Modellierung mit Prototypen"). Gleichzeitig dienen die erstellten Artefakte als *Design Rationale* (siehe Kapitel 4.2.3—"Design Rationale"), da

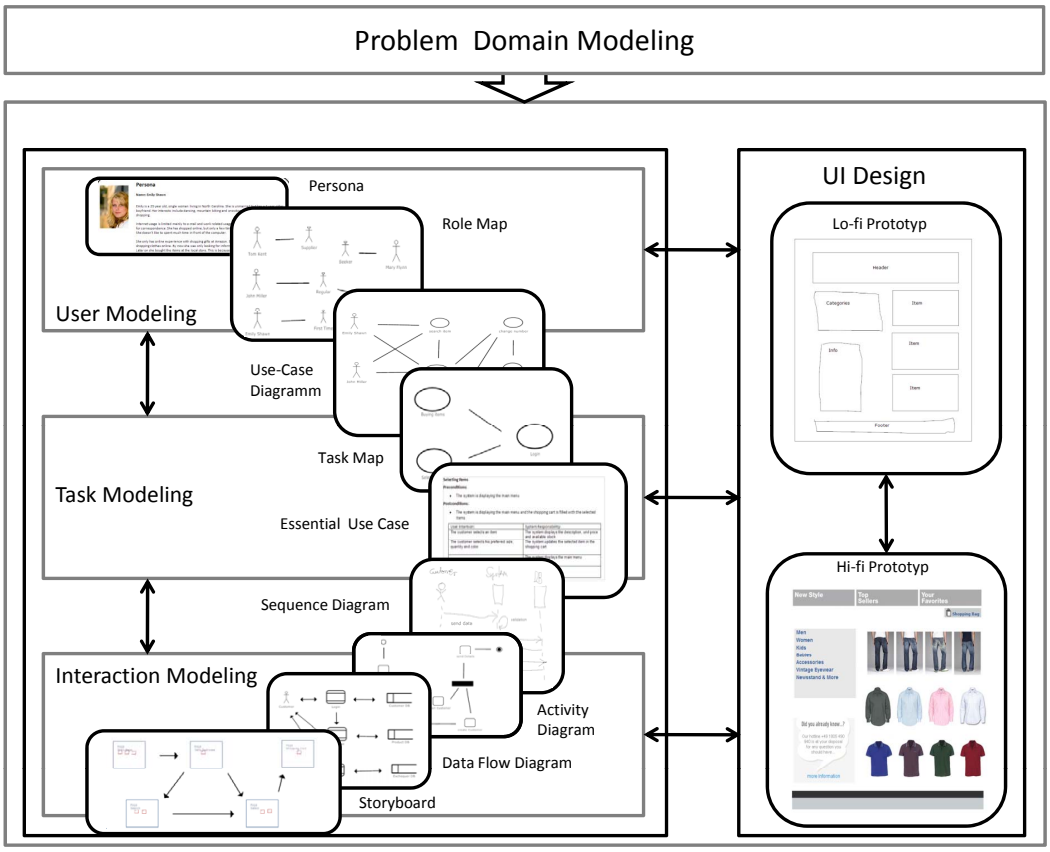


Abbildung 5.9: Schematischer Modellierungsablauf, welcher mehrmals iteriert wird und vom Problem Domain Modeling bis zum UI Design reicht (vlg. Memmel und Reiterer (2008b))

alle Objekte erhalten und nicht ersetzt werden. Treten Fragen oder Kontroversen zur Umsetzung auf, so lassen sich die Entwürfe durch die vorhandenen Modelle legitimieren. Durch erhöhte Kommunikation und eine vereinfachte (agile) Notation (siehe Kapitel 4.2.1—"Agile Modeling") können die Nachteile von formalen Modellen (siehe Kapitel 4.1.3—"Formale Ausdrucksweisen") ausgeglichen werden. Gleichzeitig kommen die Vorteile von informalen Beschreibungen aber zum Tragen (siehe Kapitel 4.1.4—"Informale Ausdrucksweisen") und die Möglichkeit zur kooperativen Zusammenarbeit ist für die *Stakeholder* aus allen Disziplinen gegeben.

Interaktive Spezifikation zur Kommunikation der Anforderungen

Das Resultat ist ein strukturierter Ansatz zur Erstellung einer interaktiven Spezifikation, der besser zur Kommunikation der Anforderungen geeignet ist, als bisherige textbasierte Dokumente (siehe Kapitel 4.1.1—"Textbasierte Spezifikationen") und keine konkreten Angaben zur technischen Umsetzung beinhaltet („Was" und nicht „Wie") (siehe Kapitel 2.4—"Disziplinenübergreifende Zusammenarbeit bei der Anforderungsermittlung"). Da alle *Stakeholder* partizipieren können, ist es möglich, funktionale UIs mit hoher Gebrauchstauglichkeit zu entwickeln, welche es den Benutzern ermöglichen, ihre Aufgaben effektiv und effizient auszuführen (siehe Kapitel 2.2—"Mensch-Computer-Interaktion"). Der in diesem Kapitel vorgestellte Ansatz für eine interdisziplinäre Spezifikation verwendet einfache, verständliche Modelle, welche den Kommunikationsprozess unterstützen, weshalb die folgende Forderung erfüllt werden kann:

„We need to elaborate a more structured representation with the specific purpose of addressing communication among team members, i.e., to support their decision-making processes and to serve as a reference material to which every team member may refer during design and development. It is important that this representation comprise a minimal set of elements that are required to create a shared understanding of the solution being elaborated, and to motivate reflection of each professional in his own field.“

(Barbosa et al., 2004)

Kapitel 6

Interaktive visuelle Spezifikation

Im folgenden Kapitel wird das Werkzeug *INSPECTOR* (*Interdisciplinary Specification Tool*) vorgestellt. Nach einer Einführung in die technischen Hintergrunde wird gezeigt, wie das konzipierte Rahmenwerk und die enhaltenen Modelle darin abgebildet werden (siehe Kapitel 5—"Vorgehensweise und geeignete Modelle für die interdisziplinäre Spezifikation"). Werkzeug und Konsensmodelle bilden dann zusammen eine interaktive visuelle Spezifikation für interdisziplinäre Gruppen. Anschließend wird die Anwendung der Konsensmodelle an einem Beispiel gezeigt[1], bei dem es aber in erster Linie darum geht zu zeigen, wie das innovative Konzept von *INSPECTOR* den Wechsel zwischen den Modellierungsphasen unterstützt und wie die Relationen zwischen verbundenen oder zusammengehörenden Modellen hergestellt werden können.

6.1 INSPECTOR - Interdisipinary Specification Tool

Inspector

Zur Unterstützung der Vorgehensweise und basierend auf der Untersuchung von Anforderungen für Spezifikationswerkzeuge (König, 2008) wurde das *Interdisipinary Specification Tool (INSPECTOR)*[2] entwickelt. *INSPECTOR* beinhaltet die in Kapitel 5—"Vorgehensweise und geeignete Modelle für die interdisziplinäre Spezifikation" vorgestellten Konsensmodelle und stellt zur besseren Unterstützung von kreativen Vorgängen eine elektronische Zeichenfläche (engl. *Whiteboard*) zur Verfügung (Memmel et al., 2008a). Objekte auf diesem *Whiteboard* lassen sich durch ein Zeigegerät mithilfe von *Point & Click*-Operationen auswählen und verschieben. Neben dem Erzeugen von Text und dem skizzenhaften Zeichnen ist das Einbinden von Dokumenten[3], Grafiken sowie Multimediainhalten (Macromedia Flash) möglich. *INSPECTOR* basiert auf .NET und ist in C# geschrieben. Zur Realisierung der Skalierungsfunktionalität wurde das ZUI Framework Piccolo (Bederson et al., 2004) verwendet. Für Informationen zu technischen Details und weiteren Informationen zur Implementierung wird auf die Arbeit von Geyer (2008) verwiesen.

[1] Auf der beiliegenden CD ist ein Video zu finden, das den Modellierungsvorgang ausführlich darstellt.

[2] http://hci.uni-konstanz.de/inspector

[3] Unterstützt werden Dokumente von Textverarbeitungssoftware (MS Word, Open Office Writer), Präsentationssoftware (MS PowerPoint, Open Office Impress) sowie das Adobe Portable Document Format

Zoomable User Interface

Da die Anzahl an darstellbaren Artefakten durch die Größe des *Whiteboards* begrenzt ist, wurde *INSPECTOR* mit einer skalierbaren grafischen Benutzeroberfläche (engl. *Zoomable User Interface*) (ZUI) ausgestattet. Dadurch kann eine Zeichenfläche dargestellt werden, die größer als der sichtbare Bereich des Bildschirms ist. Als grundlegende Interaktionstechniken werden die für ein ZUI geläufigen Techniken des *Panning & Zooming* unterstützt (siehe Abb. 6.1).

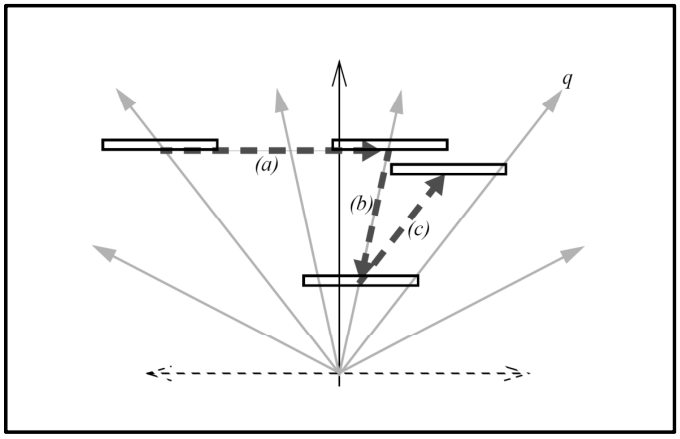


Abbildung 6.1: Panning (a) verschiebt den sichtbaren Bildschirminhalt auf der Abszissen- und Ordinatenachse, während Zooming (b) (c) einen Wechsel der Raumachse möglich macht (Furnas und Bederson, 1995)

Navigationsmöglichkeiten und Zoomtechniken

Zur Verbesserung der Verständlichkeit und zur Erhöhung der *Traceability* unterstützt *INSPECTOR* geometrisches Zooming zur Vergrößerung von Objekten und semantisches Zooming zur Anpassung des Informationsgehalts. Beim Auslösen eines Zoomvorgangs wird die aktuelle Skalierungsstufe gewechselt und zusätzliche Informationen abhängig vom aktuellen Zoomfaktor ein- oder ausgeblendet. Dabei kann die Skalierung der Oberfläche mithilfe des Mausrades manuell ausgeführt werden oder durch einen Doppelklick auf ein Objekt automatisch erfolgen. Bei diesem automatischen Zoom wird das gewünschte Artefakt zentriert und dann in einem animierten Vorgang vergrößert. Zielgerichtetes Zoomen ermöglicht das Erreichen von entfernten Artefakten. So wird es z.B. möglich, einem Hyperlink zu verknüpften Objekten zu folgen, welche im Moment nicht sichtbar und weiter entfernt sind. Ein Sprung zu anderen Elementen kann darüber hinaus auch über ein integriertes Navigationsmenü oder mittels einer einblendbaren Overview-Komponente durchgeführt werden. Die Verringerung der Skalierung kann ebenfalls manuell erfolgen oder durch einen Doppelklick auf freie Flächen im *Whiteboard*.

Hierarchischer Modellierungsraum

Durch die Verwendung eines ZUI lassen sich die einzelnen Modellierungsphasen (siehe Kapitel 5.2—"Konsensmodelle und Modellierungsphasen") hierarchisch abbilden (siehe Abb. 6.1). Dabei wird der Inhalt der Modellpalette bei jeder Zoomstufe angepasst, sodass nur aktuell verwendbare Konsensmodelle angezeigt werden. Dadurch ist es möglich den Anwender anzuleiten und ihm den Modellierungsraum und die verfügbaren Modellierungsoptionen auf optisch ansprechende Weise darzustellen, ähnlich wie es bei dem Werkzeug DAMASK (Lin und Landay, 2003) der Fall ist.

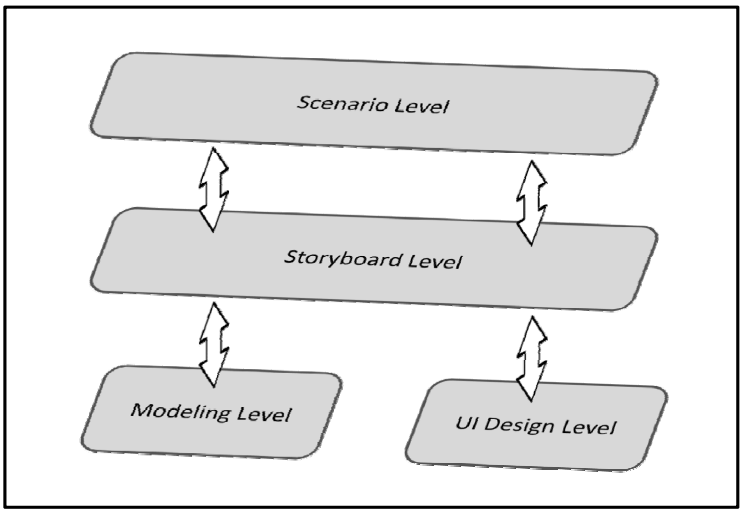


Abbildung 6.2: Die vier Skalierungsebenen in INSPECTOR (siehe Anhang A)

Scenario Level

Die erste Modellierungsebene ist der *Scenario Level*. In ihm lassen sich mithilfe von *Scenarios* erste Anwendungsfälle formulieren und Ziele für die Entwicklung in allgemeingültiger und verständlicher Form abstecken. Zusammengehörende *Scenarios* können auf dem *Whiteboard* gruppiert werden und lassen sich nach ihrer Wichtigkeit ordnen. Dokumente mit Anforderungen und zusätzlichen Informationen, wie z.B. Styleguides, lassen sich einbinden und mit *Scenarios* verknüpfen.

Storyboard Level

Das *Storyboard Level* kann erreicht werden, wenn in ein *Scenario* gezoomt wird. Dort können *Pages* angelegt werden und miteinander verbunden werden, um die Gesamtstruktur des UIs und den Dialogfluß zu bestimmen. Dadurch ist es möglich gedankliche Entwürfe von Navigationsmöglichkeiten zu visualisieren ohne zu sehr ins Detail gehen zu müssen. Zusätzlich können Modelle angelegt werden, welche die Anforderungen spezifizieren. Der *Storyboard Level* dient deshalb als Schnittstelle zwischen Modellen und UI-Entwürfen und stellt eine Relation zwischen den beiden Ebenen her.

Modeling Level

Um auf das *Modeling Level* zu gelangen ist ein Zoom in einen Modellbereich nötig. Dadurch wird es möglich auf die Konsensmodelle des *User*, *Task*, und *Interaction Modeling* zuzugreifen. Benutzer und Aufgaben sowie funktionale Anforderungen lassen sich festhalten und im Detail ausarbeiten. Durch die Verknüpfung von Modellen können enge Beziehungen zwischen Benutzern, System und Aufgaben hergestellt werden, was zu einer besseren Verständlichkeit und zu einer höheren Nachvollziehbarkeit führt.

UI Design Level

Entsprechend der soeben beschriebenen Vorgehensweise ist es möglich vom *Storyboard Level* durch einen Zoomvorgang in den *UI Design Level* zu gelangen. Hier ist es dem Anwender möglich, sowohl abstrakte Entwürfe als auch detaillierte Prototypen mit konkreten *Widgets* zu erstellen. Durch Verlinkungen zwischen den einzelnen *Designs* des Konzeptentwurfes lässt sich Navigationsverhalten und Interaktion simulieren. Zudem können Modelle aus dem *Modeling Level* mit *Designs* auf dem *UI Design Level* verknüpft werden, wodurch sich Entscheidungen auf der Darstellungsschicht durch Modelle rechtfertigen lassen.

Rückmeldungen und Anmerkungen

Rückmeldungen und Ergebnisse von Evaluationen sowie Diskussionen können über *Sticky Notes* direkt auf dem *Whiteboard* festgehalten werden. Dadurch können Anmerkungen mit direktem Bezug zur Ursache schnell und einfach vorgenommen werden. Diese Notizen lassen sich durch einen Doppelklick vergrößeren, wodurch sie editierbar werden und geben durch ihre farbliche Kennzeichnung Auskunft über die Art und Schwere eines identifizierten Problems. Zur Verfügung stehen die Zustände "kritisch" (rot), "berücksichtigen" (gelb) und "in Ordnung" (grün). Aufgrund der besseren Übersicht ist es darüber hinaus möglich, eine Liste mit allen Notizen in tabellarischer Form einzublenden. Weitere Informationen zu Art und Eigenschaft dieser Feedback-Komponente sind durch König (2008) veröffentlicht worden.

Vorteile für die Spezifikation

Durch die Verwendung eines Zoom-Konzepts wird die Spezifikation im Unterschied zu textbasierten Spezifikationsdokumenten erlebbar und interaktiv. Dadurch wird es möglich, schnell zwischen der konzeptionellen Modellierungsebene und der gestalterischen Entwurfsebene zu wechseln.

Die Unterstützung von *Copy & Paste* und die Möglichkeit zum Erstellen von Templates erleichtert das schnelle Erzeugen von Elementen. Durch die Größe des Modellierungsraumes lassen sich Artefakte in einfacher Weise deponieren, bleiben aber durch Verknüpfungen dennoch auffindbar und leicht zugänglich. Zudem lassen sich Verbindungen zwischen Modellen auf unterschiedlichen Ebenen unkompliziert erstellen, wodurch Anforderungen nachverfolgbar sind und resultierende Entscheidungen nachvollziehbar werden.

Unterstützung von verteilten Teams

Für den Austausch von Daten ist es möglich, die interaktive Spezifikation in ein XML-Format zu speichern. Da Spezifikation und die XML-Datei beide hierarchisch aufgebaut sind, ist es einfach, getrennt entwickelte Teile zusammenzuführen. Dadurch wird es möglich, dass verteilte Teams an einer Spezifikation arbeiten können, weil Ergebnisse kombiniert werden können. Daneben ist der Export von Oberflächen in XAML oder UsiXML möglich, weswegen spezialisierte Werkzeuge, wie etwa *Interfacebuilder*, für die Weiterbearbeitung eingesetzt werden können. An einem Beispiel wird der Vorgang der Modellierung und das Zusammenwirken von Zoomkonzept und Konsensmodellen im nächsten Kapitel genauer erklärt.

6.2 Interaktive Modellierung

Aufbau der Oberfläche

Abbildung 6.3 zeigt die erste Phase einer Modellierung mit *INSPECTOR*. Zu sehen ist das *Whiteboard* (1) mit einer *Scenario Map*, auf der bereits erste *Scenarios* in eine Relation gebracht wurden, um den Problemraum abzustecken. Rechts im Bild ist die Modellpalette zu sehen, deren Inhalt sich automatisch an die aktuellen Skalierungsebene anpasst (2). Zur Modellierung des Anwendungsbereiches *Scenario Map* können *Scenario Shapes* und *Info Shapes* verwendet werden. *Info Shapes* sind auf jeder Ebene vorhanden und können zum Anlegen von textbasierten Informationen und zur Ablage von Dokumenten verwendet werden. Der Pfeil symbolisiert den Vorgang des Einfügens von Modellen auf das *Whiteboard* per *Drag & Drop*. Eingefügte Objekte sind nicht fixiert, sondern lassen sich auf dem *Whiteboard* verschieben oder in ihrer Form und Gestaltung verändern. In der Werkzeugliste finden sich Hilfsmittel, die auf jeder Ebene verwendet werden können (3). Durch die Auswahl der geeigneten Option lassen sich z.B. Verbindungen zwischen Objekten herstellen. Da-

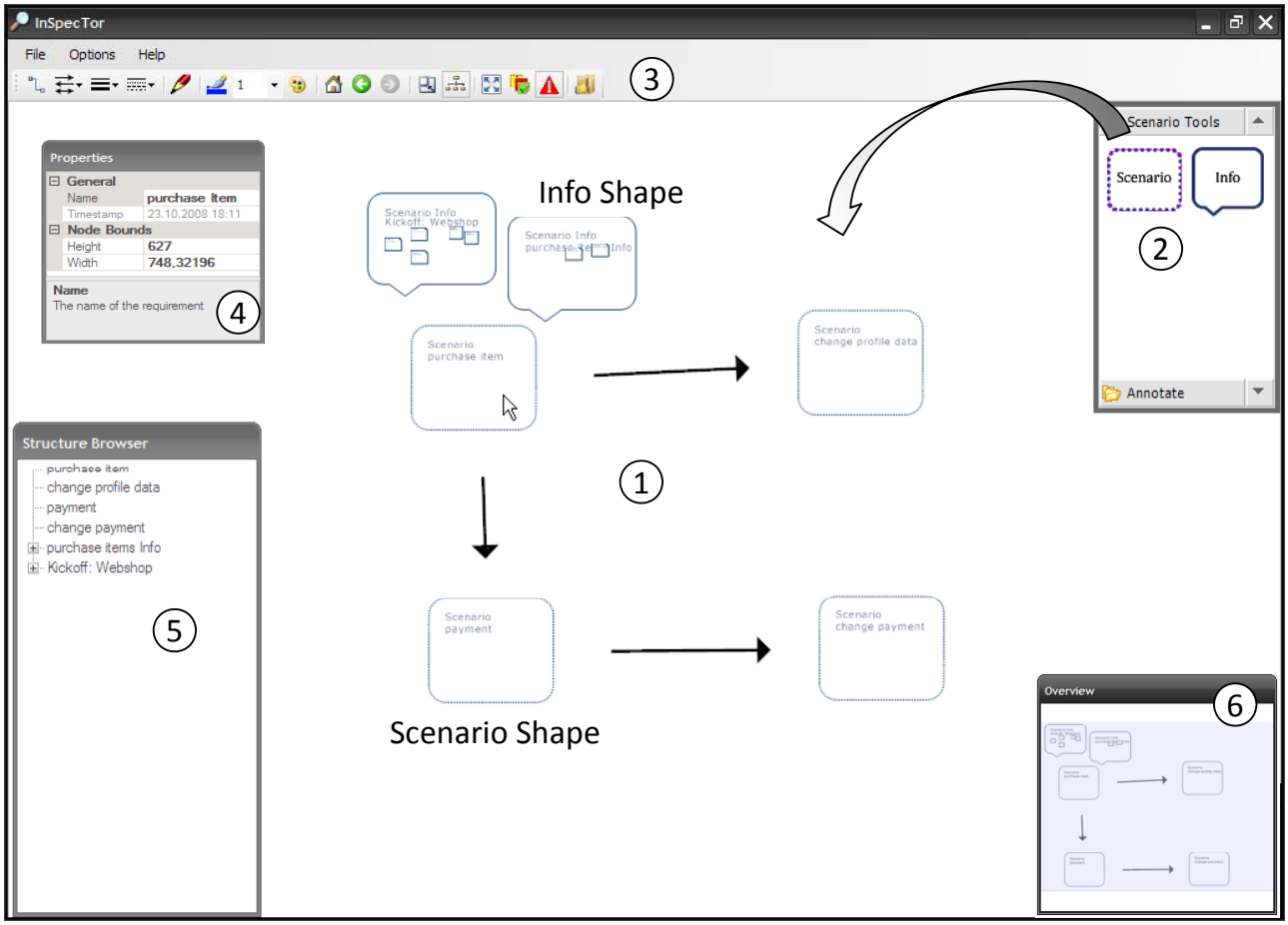


Abbildung 6.3: Die Scenario Map als Startpunkt der Modellierung (siehe Anhang A)

neben ist es möglich, den Mauszeiger in einen Stift umzuschalten, um skizzenhafte Entwürfe durchführen zu können. Zudem gibt es zur Verbesserung der Gebrauchstauglichkeit eine *History*-Funktion und einen *Home Button*, um von jeder Ebene schnell und unkompliziert in die Scenario Map gelangen zu können. Rechts im Bild befindet sich das *Properties*-Fenster (4), welches nähere Informationen zum gerade ausgewählten Objekt enthält und dem Benutzer erlaubt, bestimmte Eigenschaften zu ändern. Zur besseren Übersicht gibt es den *Structure Browser* (5), der alle Elemente in einer Baumstruktur anzeigt, und das *Overview*-Fenster rechts unten (6). Beide können ebenfalls zur Navigation verwendet werden. Über einen Doppelklick ist es möglich, einen automatischen Zoom zum ausgewählten Objekt auszulösen.

Vorgang der Modellierung

Am Beispiel eines Onlineshops für Bekleidung werden die nächsten Schritte bei der Modellierung des UIs erklärt. Die durch *Drag & Drop* eingefügten *Info Shapes* eignen sich zur Definition von Zielen für die Modellierung und können zusätzliche Informationen wie *Styleguides* oder bereits erhobene Anforderungen für die spätere Umsetzung aufnehmen. Erste Entwürfe in Form von Skizzen oder Muster können angelegt werden und dienen zur Klärung der Anwendungsdomäne sowie zur Definition von *Scenarios*. Für dieses Beispiel wurden die folgenden vier Anwendungsfälle für einen Benutzer identifiziert – *"purchase item"*, *"change profile data"*, *"payment"* und *"change payment"*. Durch Verbindungslinien können die *Scenarios* zu einer *Scenario Map* verbunden und gruppiert werden.

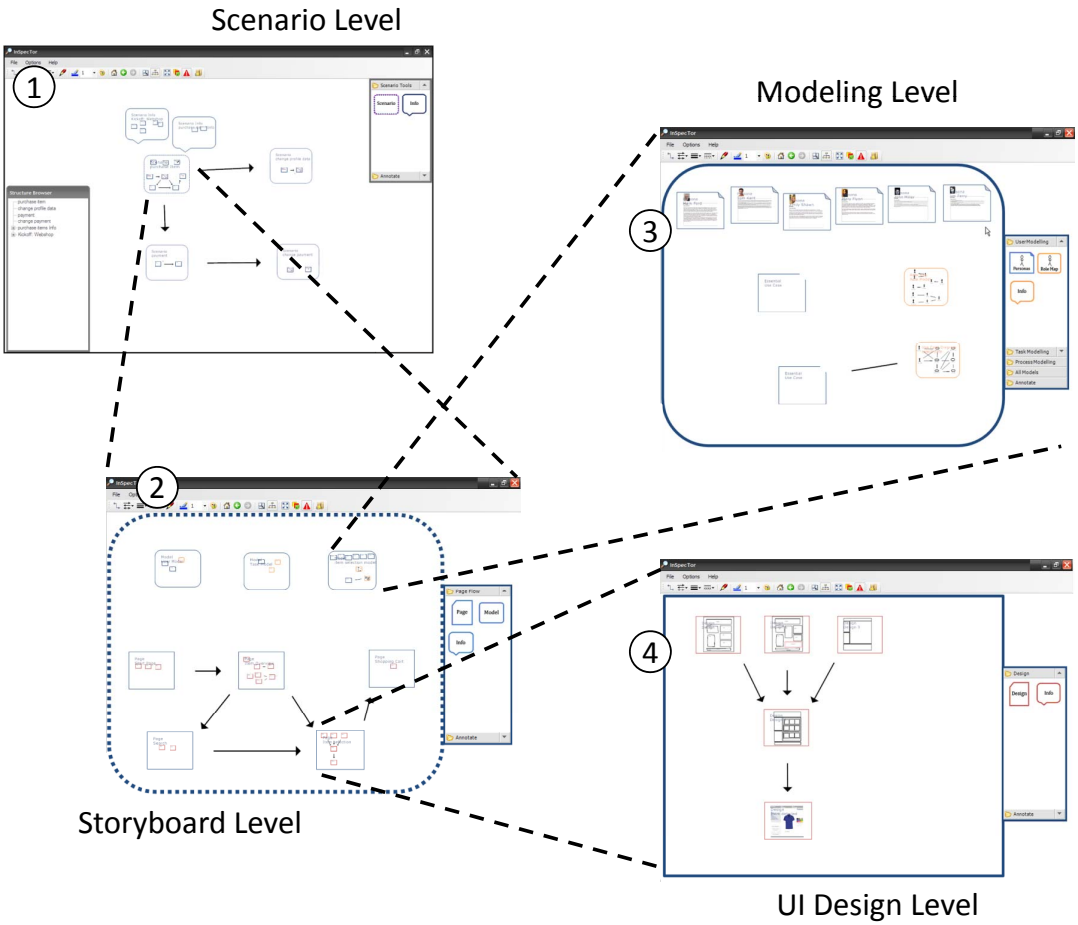


Abbildung 6.4: Die vier Skalierungsebenen dargestellt im Werkzeug (siehe Anhang A)

Hierarchiewechsel durch zoomen

Weitere Schritte lassen sich vornehmen, nachdem ein Zoom durch einen Doppelklick auf das *Scenario "purchase item"* ausgelöst wird. Dadurch wird das *Scenario Level* (1) verlassen und der *Storyboard Level* (2) erreicht, was in Abbildung 6.4 zu sehen ist. Das ausgewählte Scenario wird vergrößert und zentriert, gleichzeitig wird ein semantischer Zoom ausgelöst und Details kommen zum Vorschein. Die Modellpalette befindet sich nun direkt am Scenario und enthält neben dem obligatorischen *Info Shape* auch *Model* und *Page* Elemente. Der *Storyboard Level* stellt den Ablauf von Zuständen innerhalb eines Scenarios mithilfe einer *Storyboard Map* genauer dar. In ihr werden *Models* und *Pages* eingefügt und durch Verbindungen lässt sich die Sequenz einer Interaktionsabfolge mithilfe von unterschiedlichen Pfeil- oder Linienarten darstellen. *Pages* dienen zur eigentlichen Erläuterung von Zuständen und werden durch *Models* ergänzt, welche die definierten Konsensmodelle zur Modellierung von Benutzern, Aufgaben und Interaktionen bereitstellen (siehe Kapitel 5—"Vorgehensweise und geeignete Modelle für die interdisziplinäre Spezifikation"). Die Zustände innerhalb des *Scenarios "purchase item"* sind beim vorgestellten Beispiel des Onlineshops *"start page"*, *"item overview"*, *"search item"*, *"item selection"* und *"shopping cart"*. Als *Models* werden *"user model"*, *"task model"* und *"interaction model"* erstellt. Durch einen Zoom in *Models* oder *Pages* können die Ebenen *Modeling Level* (3) und *UI Design Level* (4) erreicht werden.

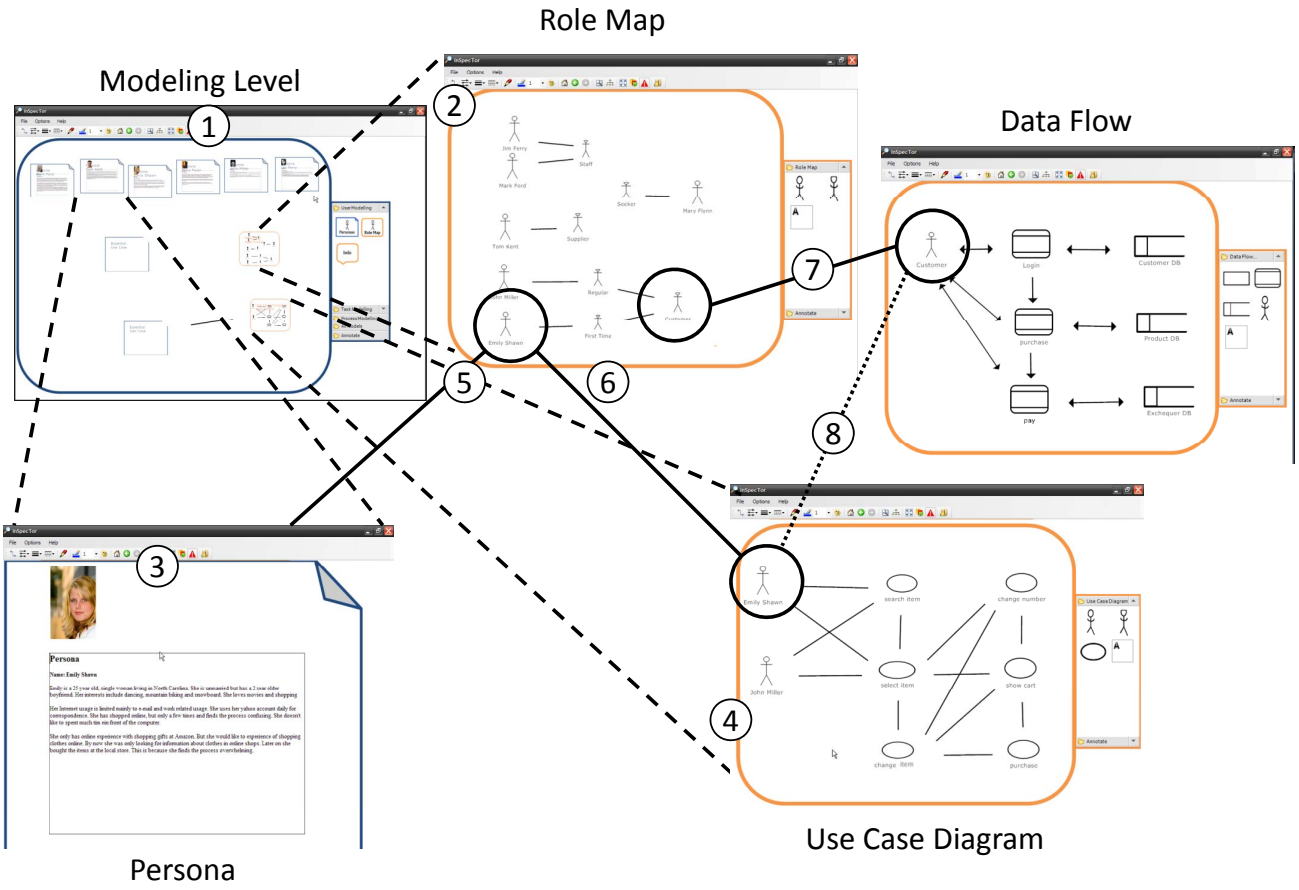


Abbildung 6.5: Zusammenhänge zwischen Modellen lassen sich in INSPECTOR herstellen (siehe Anhang A)

Anknüpfungspunkte der verwendeten Modelle

Wird das *Modeling Level* in dem Modellierungsbeispiel durch einen Zoom vom *Storyboard Level* in das *"user model"* betreten, so stehen *Role Maps* und *Essential Use Cases* zur Definition von Benutzern zur Verfügung. In Abbildung 6.4 können dann, wie hier bereits geschehen *Personas* und *Role Maps* angelegt werden. Durch einen erneuten Zoom in eine *Role Map* (2) lassen sich anschließend Rollen anlegen. Für den Onlineshop sind dies z.B. die Rollen *"Customer"*, *"Supplier"*, *"Staff"* und *"Information Seeker"*. Rollen lassen sich ähnlich wie objektorientierte Klassen ableiten und vererben ihre Eigenschaften, was durch Verbindungslinien zu Unterrollen symbolisiert werden kann. So kann ein *Customer* z.B. in die Unterrollen *"First Time Customer"* und *"Regular Customer"* aufgeteilt werden. Dadurch ist es möglich, Gruppen von Benutzern zu erstellen, sodass sich Merkmale verallgemeinern lassen. Spezielle Rollen können durch Personas genauer spezifizieren werden (siehe Kapitel 5.2.2—"User Modeling"). Für die Rolle des *"First Time Customer"* wurde deshalb die bereits aus dem vorherigen Kapitel bekannte *Persona "Emily Shawn "* angelegt (3). Da sie noch nie in einem Onlineshop eingekauft hat, ist es besonders wichtig ihre Anforderungen und Bedüfnisse genauer zu ermitteln, um die Hürden für Erstkunden niedrig zu halten. Da ein *Model* auf dem *Modeling Level* (1) nicht auf eine spezielle Modellierungsart beschränkt ist, können die Konsensmodelle aus dem *Task* und *Interaction Modeling* verwendet werden, weshalb *Use Case* Diagramme eingefügt und näher definiert werden können. Über die *Persona "Emily Shawn"* kann ebenfalls ein Bezug zu einem *Actor* hergestellt werden, um systeminterne Vorgänge bei der Aufgabenana-

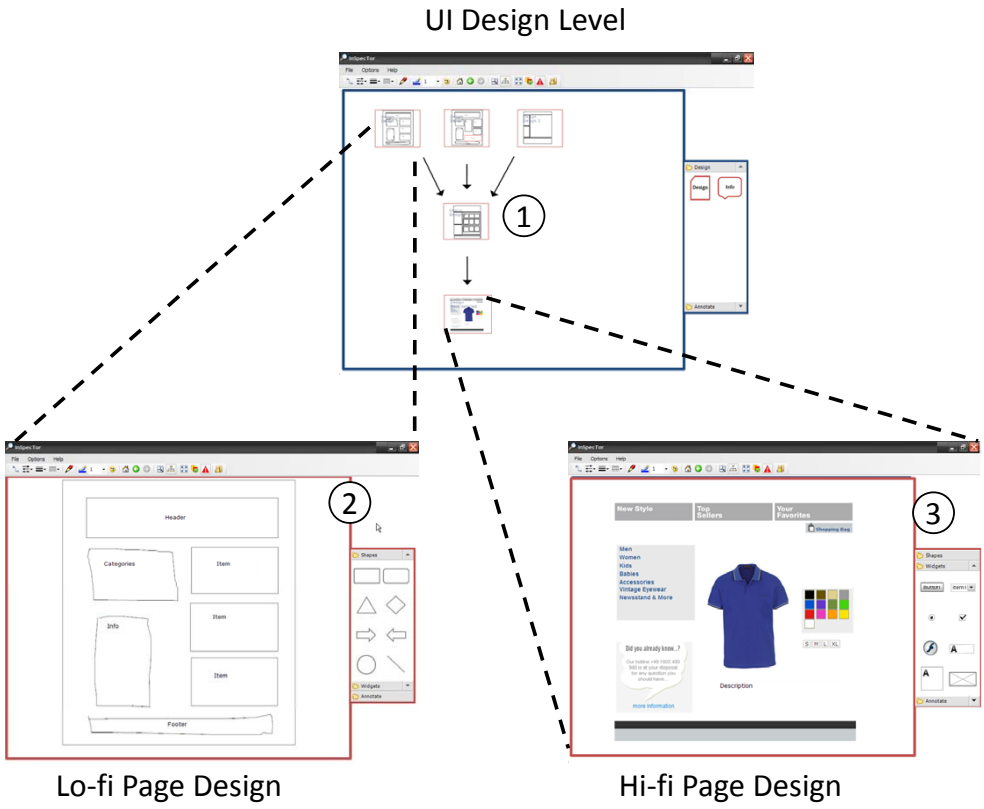


Abbildung 6.6: Prototypen mit unterschiedlicher Fidelity verbunden über das UI Design Level (siehe Anhang A)

lyse des *Scenarios "purchase item"* darstellen zu können. Dadurch ist es möglich die in diesem *Use Case* Diagramm vorhanden *Use Cases "search item"*, *"select item"*, *"buy item"*, etc. in Bezug zu einem Benutzer zu stellen. Neben der bereits in Kapitel 5.2.3—"Task Modeling" vorgestellten Verbindung von *Task Map*, *Essential Use Case* und *Use Case* Diagramm sind also auch die Verbindungen von *Persona*, *Role Map* und *Use Case* Diagramm möglich (5) (6). Gleichzeitg lassen sich Verbindungen zu anderen *Models* herstellen, die sich in einem anderen *Scenario* oder auf einem anderen *Modeling Level* befinden, wie die Verbindung zwischen der Rolle *"Customer"* in der *Role Map* zu dem im *Data Flow Diagram* definierten *Actor* darstellt. Dieser *Actor* steht in Kontakt zu systeminternen Datenbanken und empfängt oder sendet Daten während des Kaufvorgangs. Durch die Verbindung von *"Customer"* in der *Role Map* und dem *Actor* im *Data Flow* Diagramm wird gleichzeitig eine Verbindung von *Persona*, *Role* und *Actor* im *Use Case* Diagramm (8) hergestellt.

Modellierung von Oberflächen

Nachdem die Modellierung etwas vorangetrieben wurde, können erste Entwürfe für das *Scenario "purchase item"* angelegt werden. Dazu wird das *Modeling Level* über einen Doppelklick auf das *Whiteboard* verlassen und das *Storyboard Level* erreicht. Von hier aus kann in *Pages* gezoomt werden, um die Oberfläche mit Bezug auf die erzeugten Modelle zu erstellen. Abbildung 6.6 zeigt das *UI Design Level* mit mehreren Varianten eines Prototyps (1), welche alle zur Detailansicht eines Kleidungsstücks im *Scenario "purchase item"* gedacht sind. Durch iterative Verfeinerung und dem Zusammenführen von Ideen können mehrere Konzepte umgesetzt und vielversprechende Entwurfsentscheidungen in einen Prototyp mit höherer Genauigkeit übernommen werden. Zudem ist gewährleistet, dass Design Rationale erhal-

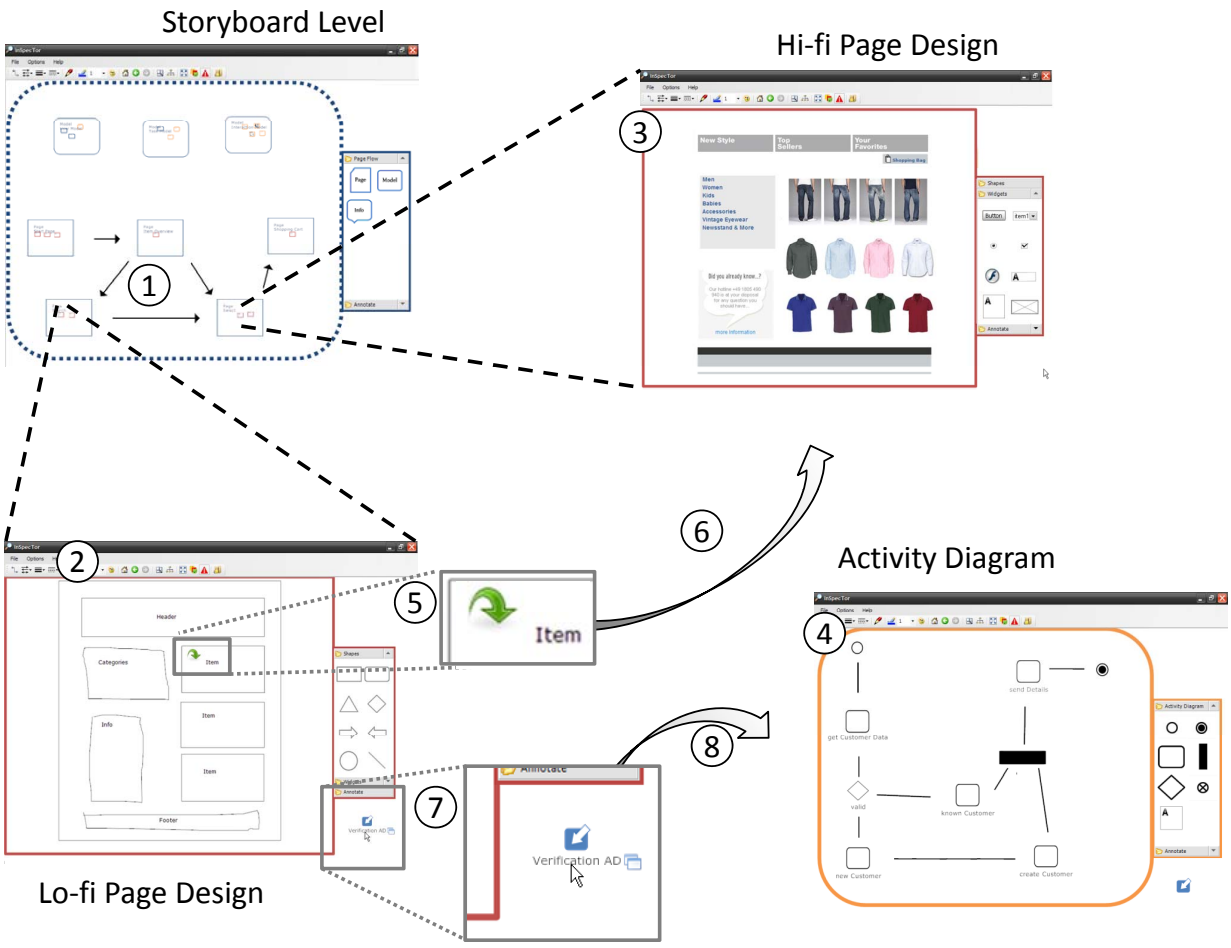


Abbildung 6.7: Unterschiedliche Verknüpfungsmöglichkeiten von Modellen um Interaktivität zu simulieren und eine bessere Traceability herzustellen (siehe Anhang A)

ten bleibt und Entscheidungen nachvollziehbar bleiben. Abstrakte Prototypen (2) können gezeichnet werden, durch ihren *Mixed/Multi-fidelity* Charakter (siehe Kapitel 5.2.5—"User Interface Modeling") ist es aber auch möglich Text, *Shapes*, Bilder, Platzhalter oder die aus den *GUI-Buildern* bekannten Oberflächenkomponenten wie etwa *Buttons* und Textfelder einzufügen. Zur Darstellung von interaktiven Inhalten und Animationen kann Macromedia Flash eingebunden werden. Durch das Einfügen dieser *Widgets* können detailliertere Prototypen (3) für den Onlineshop entworfen werden. Zur Arbeitserleichterung beim Entwerfen von weiteren Oberflächen können *Templates* aus einzelnen Elementen erstellt und verwendet werden. Diese Funktion ermöglicht es, neben der Verwendung von *Copy & Paste*, dass sich bereits getroffene Entscheidungen mehrmals vervielfältigen lassen, sodass die Erzeugung von Komponenten oder Oberflächen einfach und schnell abläuft.

Interaktives Verhalten und Traceability

Da der Modellierungsraum eine räumliche Trennung von Objekten ermöglicht ist es notwendig, Zusammenhänge zwischen Elementen herstellen zu können. In Abbildung 6.7 sind auf dem *Storyboard Level* (1) bereits Verbindungen zwischen Zuständen hergestellt, aber es ist nicht definiert, durch welche Aktion ein Zustandswechsel ausgelöst wird. Zur besseren Veranschaulichung kann ein Link auf eine Komponente innerhalb eines Entwurfs (2) erstellt werden. Diese Links werden mithilfe von *Drag & Drop* eingefügt. Dazu lässt sich das Zielobjekt aus der Baumansicht des bereits erwähnten *Structure Browser* auf ein Objekt im *Whiteboard* ziehen.

Auf die gleiche Weise lassen sich Referenzen anlegen, um die Verbindung zwischen abhängigen Modellen herstellen zu können. Das Resultat ist ein Symbol (5) (7), das die Verbindung zu einem anderen Zustand darstellt und beim Anklicken einen automatischen Zoom (6) (8) auf das Zielobjekt (3) (4) auslöst. Dabei werden Links an ein Element angeheftet, während Referenzen unten rechts angezeigt werden. Beim Erzeugen von Referenzen wird ebenfalls am Zielobjekt eine Referenz erstellt, sodass die Verbindung auf beiden Seiten dargestellt wird. Werden die Symbole für Link oder Referenz mit der Maus berührt, so wird darunter eine Liste mit den verbundenen Artefakten sichtbar. Durch einen Doppelklick auf einen Eintrag in der Liste kann ein Zoomvorgang ausgelöst werden. Dieser erfolgt bei einem Link sofort, um einen Zustandswechsel darstellen zu können. Wird eine Referenz angeklickt, so findet hingegen ein animierter Zoom statt, mit welchem es dem Benutzer möglich ist, einen besseren Überblick über die Modellierungslandschaft zu gewinnen. Am Beispiel des Onlinshops ist es deshalb möglich, durch einen Mausklick auf einen Link zwischen einzelnen Oberflächen zu wechseln, wodurch sich die Interaktivität des UIs simulieren und erleben lässt. Zudem können Modelle wie z.B. *Activity* Diagramme mit *Pages* verknüpft werden, um die Traceability zwischen Anforderungen und Oberflächenentwurf herzustellen.

Kapitel 7

Zusammenfassung

Im Rahmen dieser Arbeit wurde eine Methode vorgestellt, welche es Akteuren aus den Disziplinen Software Engineering, Mensch-Computer Interaktion und Business Engineering ermöglicht, die Anforderungen für Benutzerschnittstellen gemeinsam zu modellieren und zu spezifizieren. Dabei wurden Ziele aufgestellt, die im Folgenden noch einmal rückblickend dargestellt werden.

Interdisziplinärer Ansatz

Da interaktive Systeme nicht mehr nur Expertenwerkzeuge sind, wird an die Benutzerschnittstelle hohe Anforderungen gestellt. Benutzer müssen in der Lage sein, ihre Tätigkeiten in effizienter und effektiver Weise zu erledigen. Deshalb kommt es darauf an, dass die Software neben der erforderlichen Funktionalität auch gebrauchstauglich ist. Da die Aufgaben der Benutzer in einem betriebswirtschaftlichen Zusammenhang stehen, ist es darüber hinaus notwendig ökonomische Planungen und Geschäftsprozesse zu berücksichtigen. Dies kann nur bewerkstelligt werden, wenn es Akteuren aus allen Fachrichtungen möglich ist, die Anforderungen in gemeinsam verständlicher Form zu spezifizieren (siehe Kapitel 2—"Interessengruppen und Vorgehensweisen") .

Agile Herangehensweise

Informationstechnologien, Organisationen, Wettbewerbssituationen und das Umfeld der Unternehmen sind einem schnellen Wandel unterworfen. Deshalb können schwergewichtige und stark formalisierte Herangehensweisen durch ihren hohen Detail- und Dokumentationsgrad nicht mehr den erhofften Nutzen liefern (siehe Kapitel 2.1.4—"Agile Methoden"). Flexible und agile Vorgehensweisen basieren hingegen auf leichtgewichtigen Prozessen, welche sehr gut an die aktuellen Bedürfnisse angepasst werden können. Zudem wird bei der Modellierung mehr Wert auf Kommunikation und Kooperation gelegt, weshalb diese Herangehensweisen verständlicher für interdisziplinäre Gruppen sind (siehe Kapitel 4.2.1—"Agile Modeling").

Kreativität

Modelle, die einen eher unfertigen und rauhen Charakter haben, werden bei Entwurfsprozessen eher Änderungen unterzogen als detaillierte Entwürfe. Deshalb ist es gerade bei kreativen Vorgängen wichtig, dass informale Darstellungen verwendet werden. Sie wirken stimulierend und weniger einschränkend und schaffen dadurch Freiraum für Innovation. Zudem unterstützen sie durch ihren anschaulichen Charakter die Entscheidungsfindung und verbessern die Kommunikation bei kooperativ durchgeführten Modellierungsvorgängen (siehe Kapitel 4.1.4—"Informale Ausdrucksweisen").

Traceability

Da sich neue Erkenntnisse, Einsichten und Bedürfnisse erst während des Entwicklungsprozesses ergeben, sind Anforderungen einem Wandel unterzogen, weshalb sie sich jederzeit entwickeln und ändern können. Um Inkonsistenzen und Widersprüchlichkeiten zu identifizieren, müssen Abhängigkeiten erkennbar und verfolgbar sein. Dadurch kann gewährleistet werden, dass die Spezifikation eindeutig und aussagekräftig bleibt (siehe Kapitel 2.4—"Disziplinenübergreifende Zusammenarbeit bei der Anforderungsermittlung"). Zudem lassen sich Fragen oder Kontroversen zur Umsetzung durch die zugehörigen Anforderungen klären und Designentscheidungen legitimieren (siehe Kapitel 4—"Entwurfs- und Spezifikationsmöglichkeiten").

Interaktive Spezifiaktion

Eine wichtige Eigenschaft von Spezifikationen ist es, den Entwurf nachvollziehbar darzustellen. Spezifikationsdokumente sind unübersichtlich, mehrdeutig und können falsch interpretiert werden. Formale Darstellungen sind zwar eindeutig und präzise, allerdings schwer verständlich und nur für Experten geeignet. Zur Veranschaulichung des *Look & Feels* sind interaktive Spezifikationen daher besser geeignet als abstrakte Beschreibungen. Bei Unklarheiten ist es möglich, die Spezifikation auszuführen, wodurch Darstellung und Verhalten des erwünschten UIs erlebt werden kann. Diese Form der Spezifikation demonstriert das erwünschte Verhalten und verdeutlicht das Zusammenspiel zwischen UI und Benutzer, weshalb die Spezifikation verständlicher wird (siehe Kapitel 4—"Entwurfs- und Spezifikationsmöglichkeiten").

Vorhandene Ansätze nicht geeignet

Wie in Kapitel 3—"Ansätze zur Modellierung von Benutzerschnittstellen" gezeigt wurde, sind vorhandene Ansätze, die sowohl die funktionale als auch die benutzerorientierte Modellierung der Anforderungen ermöglichen, nicht geeignet für die Erfüllung der gestellten Ziele. Deshalb wurde eine neue Vorgehensweise entwickelt, mit welcher die fächerübergreifende Modellierung und Spezifizierung möglich ist (siehe Kapitel 5—"Vorgehensweise und geeignete Modelle für die interdisziplinäre Spezifikation").

Leichtgewichtiger, mehrmals iterierter Prozess

Dabei ist ein leichtgewichtiger Prozess entstanden, der aus fünf Abschnitten besteht, welche mehrmals iteriert werden, sodass aus den abstrakten textbasierten Beschreibungen detaillierte Entwürfe entstehen können, welche die Anforderungen schlußendlich spezifizieren (siehe Kapitel 5—"Vorgehensweise und geeignete Modelle für die interdisziplinäre Spezifikation").

Problem Domain Modeling

Begonnen wird mit dem *Problem Domain Modeling* um die Anwendungsdomäne und den Problembereich zu analysieren sowie erste Anforderungen an Benutzer und Aufgaben ermitteln zu können. Dabei wurden als allgemein verständliches Konsensmodell *Scenarios* identifiziert.

User Modeling

In der folgenden Phase des *User Modeling* werden Bezüge zu Anwendern hergestellt, um die Gebrauchstauglichkeit der Oberfläche für die spätere Nutzung zu erhöhen. Dabei wird eine Generalisierung durch abstrakte *User Roles* vorgenommen. Besonders wichtige *User Roles* lassen sich durch konkrete *Personas* verdeutlichen, um ein genaueres Verständnis zu schaffen.

Task Modeling

Aus den ermittelten Benutzern können Aufgaben abgeleitet werden, die in der Phase des *Task Modeling* genauer analysiert werden. Dabei wird sowohl die Systemsicht durch *Use Case* Diagramme als auch die Benutzersicht mithilfe von *Essential Use Cases* bzw. *Task Maps* abgebildet. So wird es möglich sowohl Intentionen und Ver-

antwortlichkeiten als auch Aktionen und Reaktionen innerhalb von Aufgaben und Tätigkeiten darzustellen.

Interaction Modeling

Während des *Interaction Modelings* werden die funktionalen Aspekte der ermittelten Aufgaben konkretisiert. Zur Darstellung der Systemsicht werden *Flow Charts*, *Activity* und *Sequence* Diagramme verwendet. Abläufe, Zustandswechsel und zeitliche Reihenfolgen lassen sich dadurch festlegen. *Storyboards* helfen bei der Strukturierung der eigentlichen Oberfläche und unterstützen den Übergang von der Modellierung in die eigentliche Gestaltung der Oberfläche.

User Interface Modeling

In der Phase des *User Interface Modeling* werden die vorhandenen modellierten Anforderungen durch die Verwendung von *Mixed/Multi-fidelity* Prototypen greifbar gemacht und münden in einer interaktiven Darstellung. Dadurch kann das *Look & Feel* der Oberfläche genau und nachvollziehbar festgelegt werden. Über mehrere Iterationsstufen können die Anforderungen und die Gestaltung der Oberfläche verfeinert werden und schließlich als interaktive Spezifikation des UIs dienen.

Verbesserung von Kommunikation und Kooperation

Für jede der fünf Ebenen wurden Modelle mit geringer Komplexität identifiziert, die in einem ähnlichen Hintergrund in allen Disziplinen vorhanden sind. Dabei war wichtig, dass sie Anknüpfungspunkte zwischen den ermittelten Hierarchien herstellen, sodass die Rückverfolgbarkeit gewährleistet werden kann. Auf die identifizierten Modelle wurden die Prinzipien des *Agile Modeling* angewandt, wodurch Kommunikation und Kooperation erhöht und die Formalität der Modelle gesenkt wird. Dadurch ist es möglich, dass auch Notationen als Konsensmodelle dienen können, die nicht in allen Disziplinen vorkommen. Durch die Verwendung von *Design Rationale* sind Anforderungen und resultierende Entwurfsentscheidungen dokumentiert, wodurch Ursache und Wirkung zugeordnet werden können. Sich ändernde Anforderungen können angepasst werden, und da das Verständnis erhöht wird, verbessern sich Kommunikation und Zusammenarbeit unter den Beteiligten aus den verschiedenen Disziplinen.

Testumgebung zur Modellierung und Spezifikation

In einem weiteren Schritt wurde das Werkzeug *INSPECTOR* zum Testen der Vorgehensweise erstellt (siehe Kapitel 6—"Interaktive visuelle Spezifikation"). Da ein Zoomable User Interface verwendet wurde, lässt sich die Hierarchie der Modellierungsebenen auf vier Skalierungsstufen abbilden. Jede Skalierungsstufe enthält spezifische Elemente aus der Modellpalette, sodass der Benutzer bei der Modellierung angeleitet werden kann. Dabei dient das *Scenario Level* als Einstiegspunkt und für die erste Analyse der Anwendungsdomäne. Durch einen Zoomvorgang in ein *Scenario* kann der *Storyboard Level* zur Strukturierung des UIs und zur Definition des Dialogflusses verwendet werden. Über das *Storyboad Level* können das *Modeling Level* und das *UI Design Level* erreicht werden. Das Storyboard Level dient deshalb als Schnittstelle zwischen Modellen und UI-Entwürfen und stellt eine Relation zwischen der Modellierung von *User*, *Task* und *Interaction Modeling* zur Gestaltung der Oberfläche im *User Interface Modeling* her. Zum Wechsel der Skalierungsebenen werden dabei verschieden Zoom-Techniken verwendet. So wird es möglich über *Zoom-Links* Relationen zwischen einzelnen Artefakten herzustellen und die Anknüpfungspunkte zwischen den Modellen herauszuheben. Die Traceability und Transparenz zwischen Abhängigkeiten kann dadurch hergestellt werden. Zudem ist es möglich, interaktives Verhalten festzulegen. *Widgets* der entworfenen Oberfläche werden mit Links versehen und beim Auslösen findet ein Wechsel des Entwurfs durch einen Zoomvorgang auf ein anderes Artefakt statt.

Ausblick

Erste Evaluationsergebnisse

Im Laufe des Projektes wurden der Modellierungsansatz und das Werkzeug bereits durch Interviews mit Experten der HCI und Tagebuchstudien evaluiert (Memmel und Reiterer, 2008c), (Geyer, 2008). Gegenstand der Befragungen war, herauszufinden, wie die Benutzbarkeit des Werkzeuges verbessert werden kann. Dabei wurde auch schon ein erster Bezug auf die Praxistauglichkeit der vorgeschlagenen Modelle hergestellt. Es wurde ersichtlich, dass die HCI-Experten eine Kombination von Modellen mit multi-fidelity Darstellungen als Verbesserung vorhandener Arbeitsabläufe ansehen (ø 4,8 Punkte, Skala 1-5 Punkte, n=6). Im Moment ist eine weitere Studie in Arbeit (n=8), bei der die verwendeten Modelle stärker berücksichtigt werden und bei der geklärt wird, ob die aktuelle Auswahl an Konsensmodellen ausreichen. Es gibt darüber hinaus aber noch offene Punkte, für die Klärungsbedarf besteht

Kritik am Modellumfang

Momentan werden die frühen Phasen der Anforderungsanalyse bei der vorgestellten Methode nicht unterstützt. Die genauen Hintergründe und die aktuelle Bedürfnislage, die zur Entwicklung des Systems führen, können im Moment noch nicht ausreichend abgebildet werden. Dazu gehört neben der Abgrenzung der Aufgabenstellung auch die genauere Erfassung der Projektziele sowie die Prüfung von Durchführbarkeit, Aufwand, Nutzen und Risiken. Vielversprechend wäre es, wenn diese frühen Phasen der Anforderungsanalyse, ebenfalls in den Modellierungsansatz mit einbezogen werden könnten. In einer weiteren Forschung müsste gezeigt werden, in wie fern dieser Aspekt in die aufgeführte Vorgehensweise integriert werden kann, sodass für alle Akteure eine fundierte Kenntnis der aktuellen Situation und dadurch auch ein besseres Verständnis der Aufgabenstellung hergestellt

Kritik an der Methode

werden kann. Geklärt ist zudem noch nicht, ob die Konzepte des *Agile Modeling* auf die Spezifikation von Anforderungen so angewendet werden können, dass alle Anforderungen verständlich dokumentiert sind. Die Verwendung dieser agilen Prinzipien verbessert zwar die Kommunikation unter den Beteiligten und fördert die Ideenfindung bei gleichzeitiger Vereinfachung der Modellierung, allerdings ist nicht genau klar, ob die resultierenden Modelle auch für die Spezifikation aussagekräftig genug sind. Da sich die Formalität von Modellen, die durch die Prinzipien des *Agile Modeling* erzeugt werden, immer nach dem Hintergrund der Modellierung richtet, können Modelle, die zum Verstehen einer Problemstellung notwendig sind, informaler dargestellt werden als dies bei Modellen der Fall ist, die zu Dokumentations- oder Spezifikationszwecken verwendet werden. Wie hoch der Grad an Formalität sein muss, sodass die Anforderungen vollständig, konsistent und nachvollziehbar wiedergegeben werden können, ist deshalb nicht klar. Zudem sind Modelle mit geringer Formalität, welche mit den Prinzipien des *Agile Modeling* erstellt wurden, nur für Akteure verständlich, die in die Modellierung und Spezifikation involviert waren. Gerade bei großen Projekten mit verteilten Teams, bei welchen unter Umständen externe Dienstleister einbezogen werden, kann die Spezifikation deshalb nicht aussagekräftig sein, weil die Anforderungen an Kommunikation und Kooperation nicht im geforderten Maß umgesetzt werden können. Daher müssten die vorgestellten Methoden zur Spezifikation an einem realen Projekt getestet werden, um zu klären ob die Prinzipien des *Agile Modeling* nicht im Widerspruch zu den Zielsetzungen einer Spezifikation stehen. Dabei müssten Experten aus den verschiedenen Disziplinen involviert sein, weshalb die Klärung dieser Fragestellung sehr aufwendig zu realisieren wäre.

Anhang A

Anhang

CD-ROM

Dieser Arbeit liegt eine CD-ROM bei, welche eine digitale Version der vorliegenden Arbeit sowie Quellcode zu INSPECTOR und ein Video zum vorgestellten Beispiels enthält (siehe Kapitel 6.2—"Interaktive Modellierung").

Für weitere Informationen zu INSPECTOR wird auf die Webseite des Projektes [1] verwiesen, auf welcher neben weiteren Videos auch Publikationen veröffentlicht sind.

[1] http://hci.uni-konstanz.de/inspector

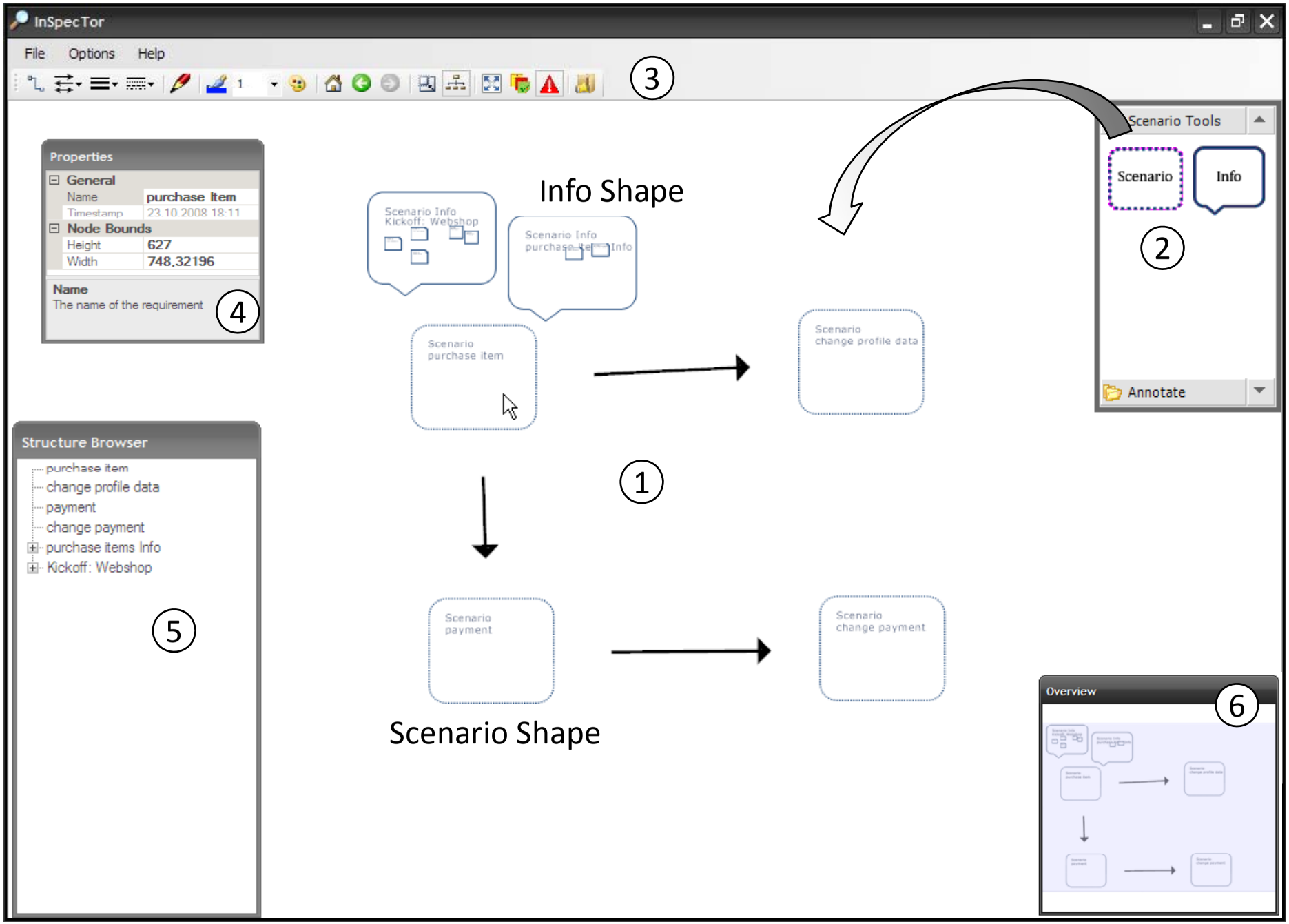


Abbildung A.1: Die Scenario Map als Startpunkt der Modellierung

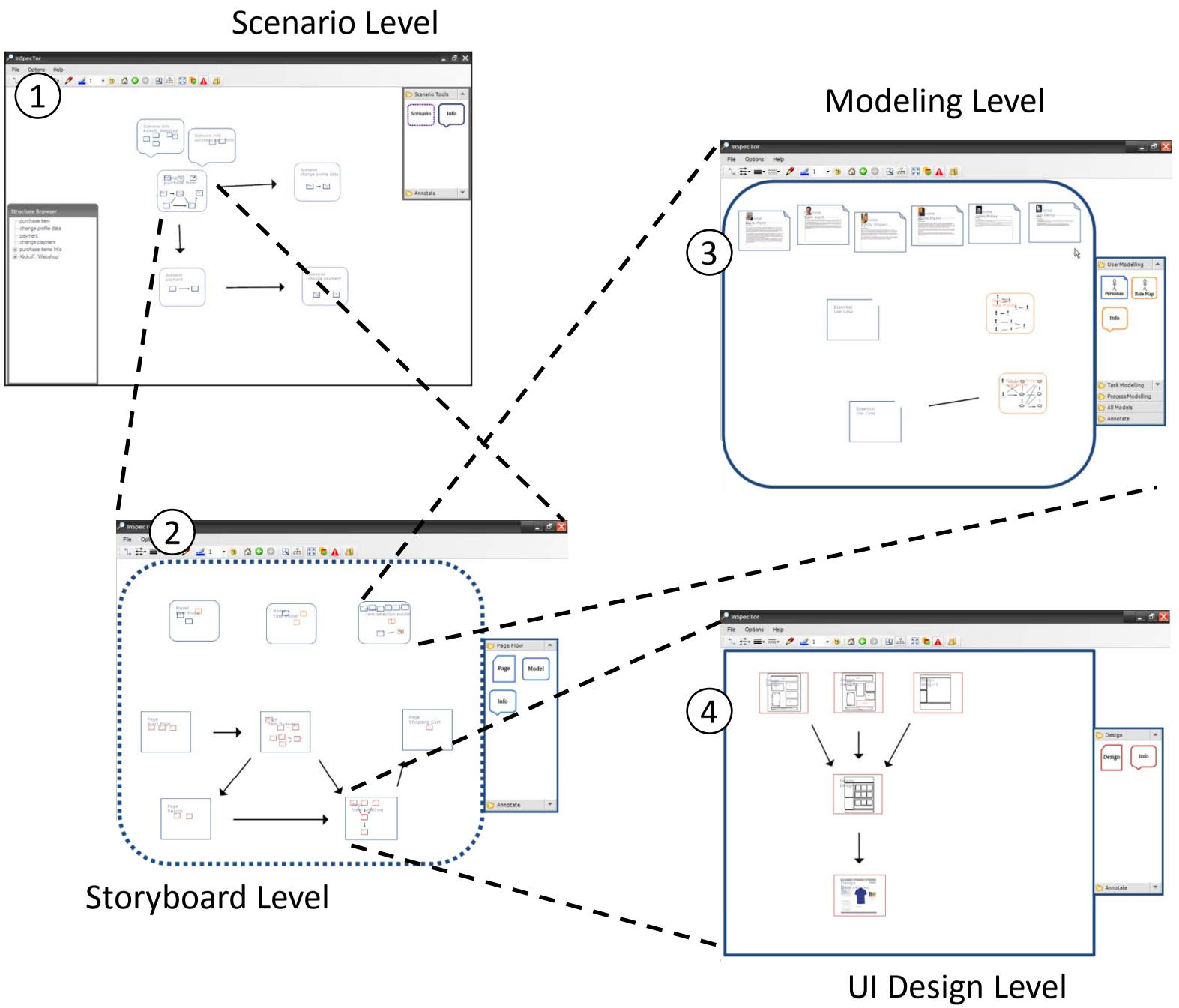


Abbildung A.2: Die vier Skalierungsebenen in INSPECTOR

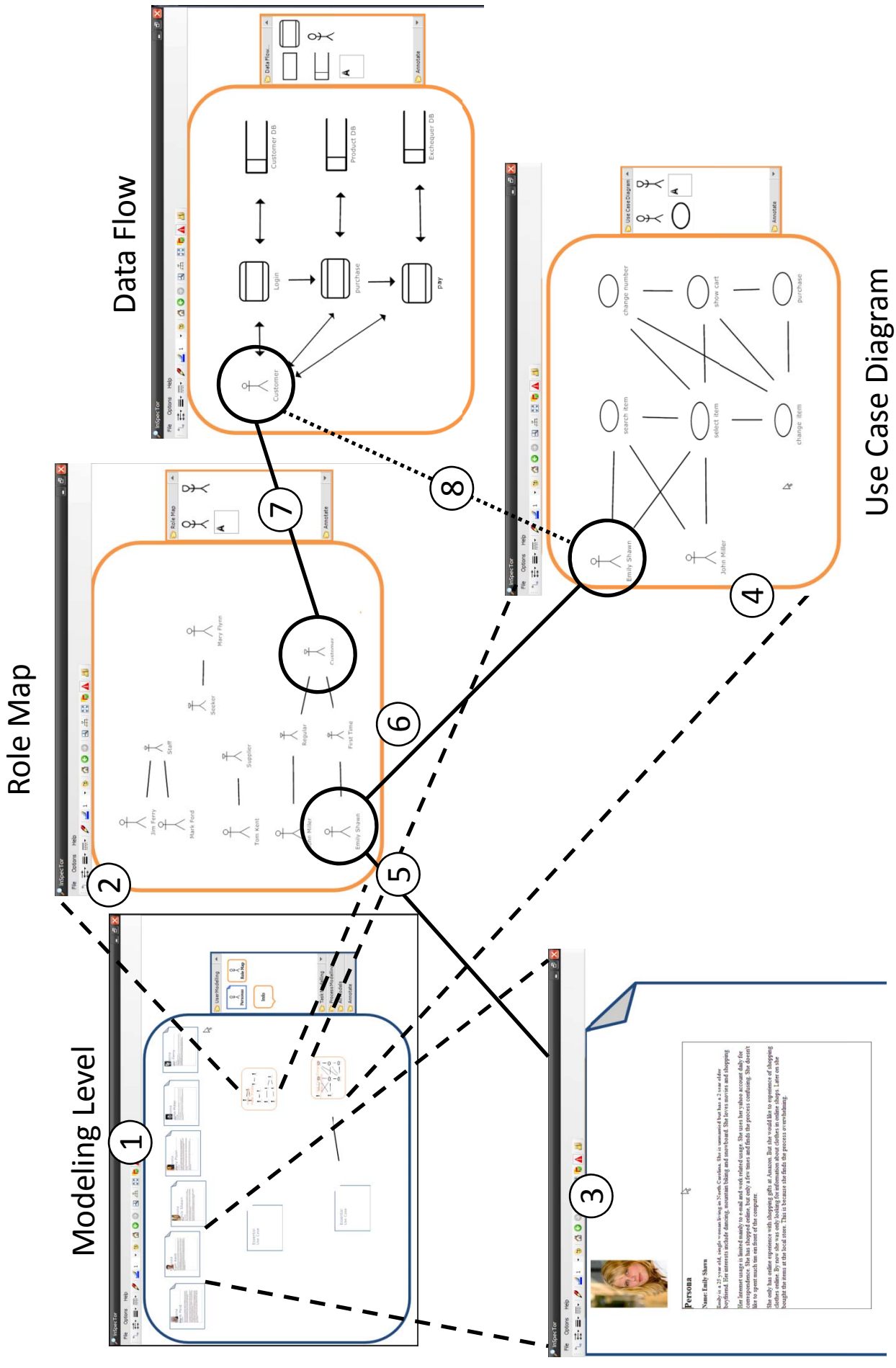


Abbildung A.3: Zusammenhänge zwischen Modellen lassen sich in INSPECTOR herstellen

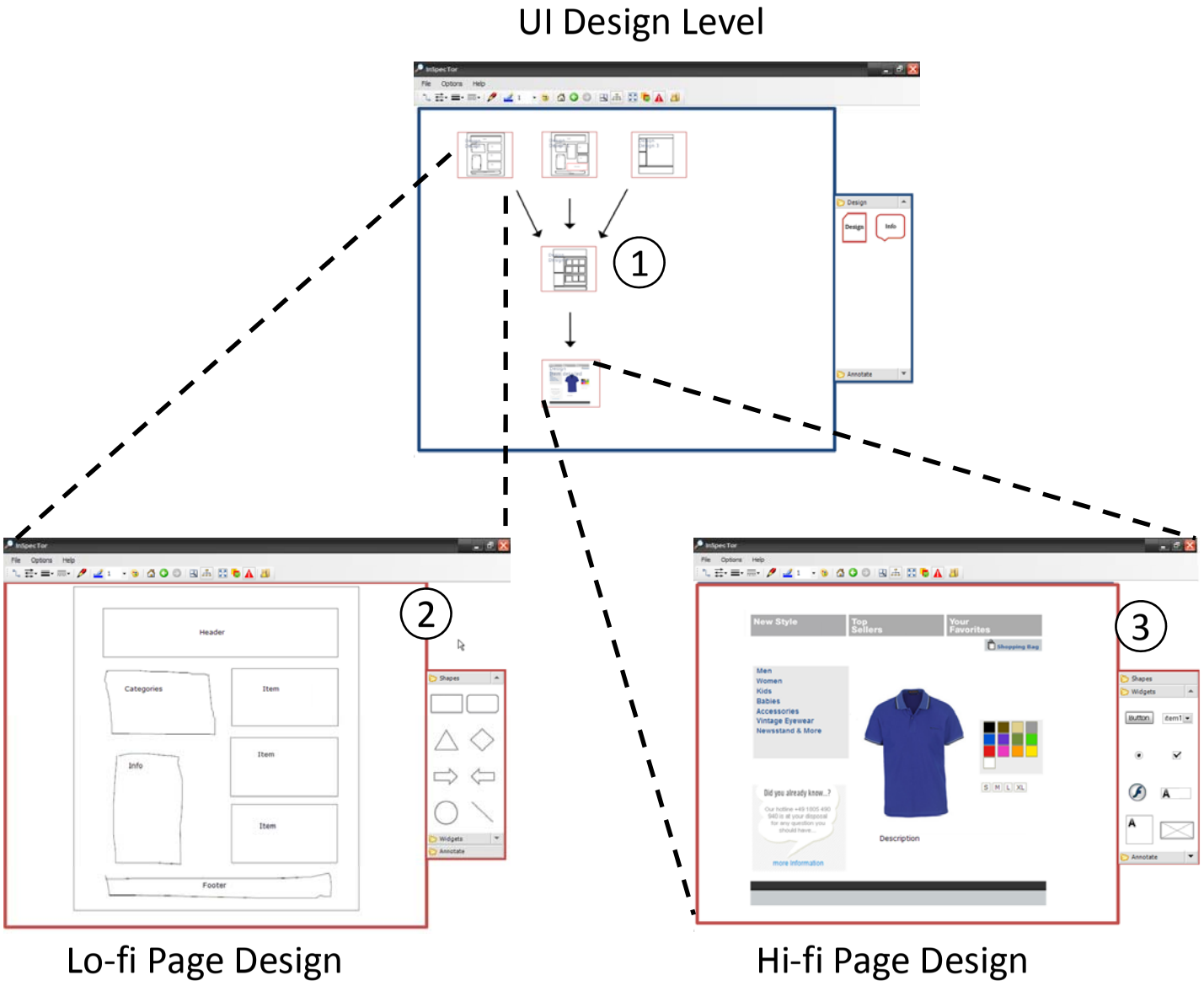


Abbildung A.4: Prototypen mit unterschiedlicher Fidelity verbunden über das UI Design Level

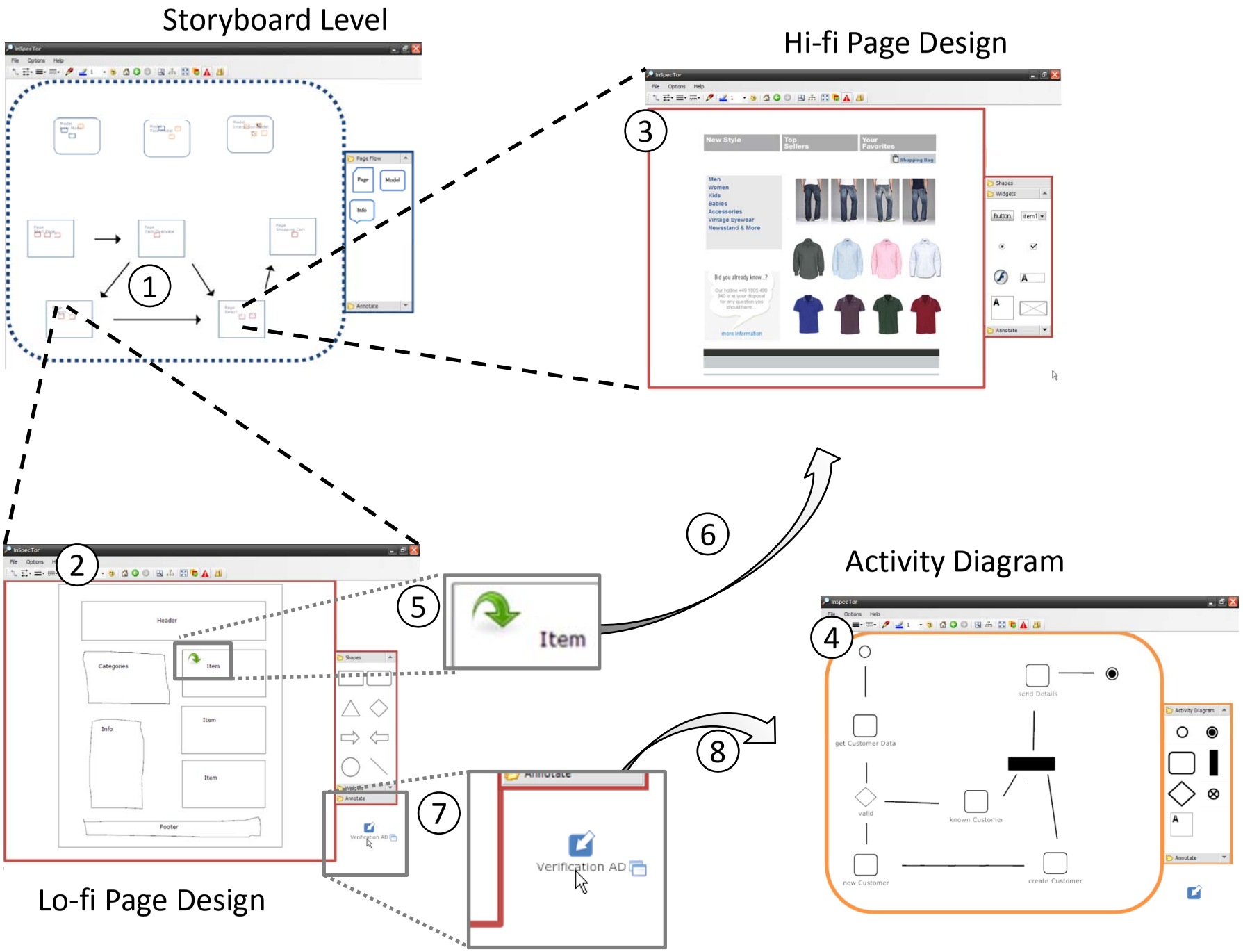


Abbildung A.5: Unterschiedliche Verknüpfungsmöglichkeiten von Modellen um Interaktivität zu simulieren und eine bessere Traceability herzustellen

Literaturverzeichnis

[ACM SIGCHI 1992] ACM SIGCHI: Curricula for human-computer interaction / ACM. New York, NY, USA : ACM, 1992. – Forschungsbericht

[Adikari et al. 2006] ADIKARI, S. ; MCDONALD, C. ; COLLINGS, P.: A design science approach to an HCI research project. In: *OZCHI '06: Proceedings of the 20th conference of the computer-human interaction special interest group (CHISIG) of Australia on Computer-human interaction: design: activities, artefacts and environments*. New York, NY, USA : ACM, 2006, S. 429–432

[Alben 1996] ALBEN, L.: Quality of experience: defining the criteria for effective interaction design. In: *Interactions* 3 (1996), Nr. 3, S. 11–15

[Almendros-Jimenez und Iribarne 2005] ALMENDROS-JIMENEZ, J. ; IRIBARNE, L.: Designing GUI components from UML use cases. In: *12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems, 2005. ECBS '05.* (2005), April, S. 210–217

[Ambler 2002] AMBLER, S.: Lessons in agility from Internet-based development. In: *Software, IEEE* 19 (2002), Nr. 2, S. 66–73

[Ambler 2003] AMBLER, S.: Agile model driven development is good enough. In: *Software, IEEE* 20 (2003), Nr. 5, S. 71–73

[Ambler 2006] AMBLER, S.: *Agile Models Distilled: Potential Artifacts for Agile Modeling*. 2006. – URL `http://www.agilemodeling.com/artifacts/`. – (Zugriff: 15 Okt 2008)

[Ambler und Jeffries 2002] AMBLER, S. ; JEFFRIES, R.: *Agile Modeling: Effective practices for Extreme Programming and the Unified Process*. Wiley, March 2002

[Anggreeni 2008] ANGGREENI, I.: Scenario-Based Product Design: Scenario generation tool as a framework of use / University of Twente Enschede. 2008. – Forschungsbericht

[Aoyama 2005] AOYAMA, Mikio: Persona-and-Scenario Based Requirements Engineering for Software Embedded in Digital Consumer Products. In: *RE '05: Proceedings of the 13th IEEE International Conference on Requirements Engineering*. Washington, DC, USA : IEEE Computer Society, 2005, S. 85–94

[Araujo et al. 2004] ARAUJO, J. ; WHITTLE, J. ; KIM, D.: Modeling and Composing Scenario-Based Requirements with Aspects. In: *RE '04: Proceedings of the Requirements Engineering Conference, 12th IEEE International*. Washington, DC, USA : IEEE Computer Society, 2004, S. 58–67

[Artim et al. 1998] ARTIM, J. ; HARMELEN, M. ; BUTLER, K. ; GULLIKSEN, J. ; HENDERSON, A. ; KOVACEVIC, S. ; LU, S. ; OVERMYER, S. ; REAUX, R. ; ROBERTS, D. ; TARBY, J.-C. ; LINDEN, K. V.: Incorporating work, process and task analysis into commercial and industrial object-oriented systems development. In: *SIGCHI Bull.* 30 (1998), Nr. 4, S. 33–36

[Aversano et al. 2005] AVERSANO, L. ; BODHUIN, T. ; TORTORELLA, M.: Assessment and impact analysis for aligning business processes and software systems. In: *SAC '05: Proceedings of the 2005 ACM symposium on applied computing*. New York, NY, USA : ACM, 2005, S. 1338–1343

[Bailey und Konstan 2003] BAILEY, B. ; KONSTAN, J.: Are informal tools better?: Comparing DEMAIS, pencil and paper, and authorware for early multimedia design. In: *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM, 2003, S. 313–320

[Bailey et al. 2001] BAILEY, B. ; KONSTAN, J. ; CARLIS, J.: Supporting Multimedia Designers: Towards More Effective Design Tools. In: *In Proc. Multimedia Modeling: Modeling Mutlimedia Information and Systems*, 2001, S. 267–286

[Barbosa et al. 2004] BARBOSA, S. ; DE PAULA, M. ; LUCENA, C.: Adopting a Communication-Centered Design Approach to Support Interdisciplinary Design Teams. In: *ICSE 2004 Workshop Bridging the Gaps II: Bridging the Gaps Between Software Engineering and Human-Computer Interaction*. Edinburgh, Scotland., 2004

[Bass und John 2001] BASS, L. ; JOHN, B. E.: Supporting usability through software architecture. In: *Computer* 34 (2001), Nr. 10, S. 113–115

[Bäumer et al. 1996] BÄUMER, D. ; BISCHOFBERGER, W. ; LICHTER, H. ; ZÜLLIGHOVEN, H.: User interface prototyping - concepts, tools, and experience. In: *ICSE '96: Proceedings of the 18th international conference on Software engineering*. Washington, DC, USA : IEEE Computer Society, 1996, S. 532–541

[Beck und Andres 2004] BECK, K. ; ANDRES, C.: *Extreme Programming Explained: Embrace Change (2nd Edition)*. Addison-Wesley Professional, 2004

[Bederson et al. 2004] BEDERSON, Benjamin B. ; GROSJEAN, Jesse ; MEYER, Jon: Toolkit Design for Interactive Structured Graphics. In: *IEEE Trans. Softw. Eng.* 30 (2004), August, Nr. 8, S. 535–546

[Berenbach 2004] BERENBACH, B.: Comparison of UML and text based requirements engineering. In: *OOPSLA '04: Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications*. New York, NY, USA : ACM, 2004, S. 247–252

[Bevan 1999] BEVAN, N.: Quality in use: Meeting user needs for quality. In: *J. Syst. Softw.* 49 (1999), Nr. 1, S. 89–96

[Beyer und Holtzblatt 1997] BEYER, H. ; HOLTZBLATT, K.: *Contextual Design: ACustomer-Centered Approach to Systems Designs (Morgan Kaufmann Series in Interactive Technologies)*. Morgan Kaufmann, September 1997

[Bias und Mayhew 2005] BIAS, R. ; MAYHEW, D.: *Cost-Justifying Usability*. 2. Edition. Oxford : Elsevier LTD, 2005

[Blomkvist 2005] BLOMKVIST, S.: Towards a Model for Bridging Agile Development and User-Centered Design. In: SEFFAH, A. (Hrsg.) ; GULLIKSEN, J. (Hrsg.) ; DESMARAIS, M. C. (Hrsg.): *Human-centered software engineering - integrating usability in the development process*. Springer, 2005, S. 219 – 244

[Blomquist und Arvola 2002] BLOMQUIST, Å. ; ARVOLA, M.: Personas in action: ethnography in an interaction design team. In: *NordiCHI '02: Proceedings of the second Nordic conference on Human-computer interaction*. New York, NY, USA : ACM, 2002, S. 197–200

[Bock 2007] BOCK, C.: Model-Driven HMI Development: Can Meta-CASE Tools do the Job? In: *HICSS '07: Proceedings of the 40th Annual Hawaii International Conference on System Sciences*. Washington, DC, USA : IEEE Computer Society, 2007

[Boehm 1988] BOEHM, B.: A Spiral Model of Software Development and Enhancement. In: *Computer* 21 (1988), Nr. 5, S. 61–72

[Boehm 2006] BOEHM, B.: A view of 20th and 21st century software engineering. In: *ICSE '06: Proceedings of the 28th international conference on Software engineering*. New York, NY, USA : ACM, 2006, S. 12–29

[Boehm et al. 1984] BOEHM, B. ; GRAY, T. ; SEEWALDT, T.: Prototyping versus specifying: a multiproject experiment. In: *IEEE Trans. Softw. Eng.* 10 (1984), Nr. 3, S. 290–302

[Booch et al. 2005] BOOCH, G. ; RUMBAUGH, J. ; JACOBSON, I.: *Unified Modeling Language User Guide, The (2nd Edition) (The Addison-Wesley Object Technology Series)*. Addison-Wesley Professional, May 2005

[Borchers 2001] BORCHERS, J.: *A Pattern Approach to Interaction Design*. John Wiley & Sons, May 2001

[Bouillon et al. 2002] BOUILLON, L. ; VANDERDONCKT, J. ; EISENSTEIN, J.: Model-Based Approaches to Reengineering Web Pages. In: *TAMODIA '02: Proceedings of the First International Workshop on Task Models and Diagrams for User Interface Design*, INFOREC Publishing House Bucharest, 2002, S. 86–95

[Bowen 2005] BOWEN, J.: *Formal specification of user interface design guidelines*, University of Waikato, Master Thesis, 2005

[Bowen et al. 1995] BOWEN, J. ; HINCHEY, M. ; EINSTEIN, A.: Ten Commandments of Formal Methods. In: *IEEE Computer* 28 (1995), S. 56–63

[Bowen und Reeves 2007] BOWEN, J. ; REEVES, S.: Formal models for user interface design artefacts. In: *Innovations in Systems and Software Engineering* 4 (2007), Nr. 2, S. 125 – 141

[Bratthall et al. 2000] BRATTHALL, L. ; JOHANSSON, E. ; REGNELL, B.: Is a Design Rationale Vital when Predicting Change Impact? A Controlled Experiment on Software Architecture Evolution. In: *PROFES '00: Proceedings of the Second International Conference on Product Focused Software Process Improvement*. London, UK : Springer-Verlag, 2000, S. 126–139

[Brooks 1987] BROOKS, F.: No Silver Bullet Essence and Accidents of Software Engineering. In: *Computer* 20 (1987), Nr. 4, S. 10–19

[Brooks 1995] BROOKS, F.: *The mythical man-month (anniversary ed.)*. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 1995

[Brown et al. 2008] BROWN, Judith ; LINDGAARD, Gitte ; BIDDLE, Robert: Stories, Sketches, and Lists: Developers and Interaction Designers Interacting Through Artefacts. In: *agile* 0 (2008), S. 39–50

[Buxton 2003] BUXTON, W.: Performance by Design: The Role of Design in Software Product Development. In: *Proceedings of the Second International Conference on Usage-Centered Design*. Portsmouth, NH, 2003, S. 1–15

[Campos und Nunes 2007] CAMPOS, P. ; NUNES, N. J.: Practitioner Tools and Workstyles for User-Interface Design. In: *Software, IEEE* 24 (2007), Nr. 1, S. 73–80

[Cao und Ramesh 2007] CAO, L. ; RAMESH, B.: Agile software development: Ad Hoc practices or sound principles? In: *IT Professional* 9 (2007), Nr. 2, S. 41–47

[Cao und Ramesh 2008] CAO, L. ; RAMESH, B.: Agile Requirements Engineering Practices: An Empirical Study. In: *IEEE Softw.* 25 (2008), Nr. 1, S. 60–67

[Carr 1996] CARR, D. A.: Toward more understandable user interface specifications. In: BODART, F. (Hrsg.) ; VANDERDONCKT, J. (Hrsg.): *Design, Specification and Verification of Interactive Systems '96*. Wien : Springer-Verlag, 1996, S. 141–161

[Carrington et al. 1990] CARRINGTON, D. ; DUKE, D. ; DUKE, R. ; KING, P. ; ROSE, G. ; SMITH, G.: Object-Z: An Object-Oriented Extension to Z. In: *FORTE '89: Proceedings of the IFIP TC/WG6.1 Second International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols*. Amsterdam, The Netherlands, The Netherlands : North-Holland Publishing Co., 1990, S. 281–296

[Carroll 1991] CARROLL, J.: *Designing Interaction: Psychology at the Human-Computer Interface (Cambridge Series on Human-Computer Interaction)*. Cambridge University Press, June 1991

[Carroll 2000a] CARROLL, J.: Five Reasons for Scenario-Based Design. In: *Interacting with Computers* 13 (2000), Nr. 1, S. 43–60

[Carroll 2000b] CARROLL, J.: *Making Use: Scenario-Based Design of Human-Computer Interactions*. Cambridge, MA, USA : MIT Press, 2000

[Carroll und Rosson 1985] CARROLL, J. ; ROSSON, M.: Usability specifications as a tool in iterative development. In: HARTSON, H. (Hrsg.): *Advances in human-computer interaction*. Norwood, NJ : Ablex Publishing Corporation, 1985, S. 1–28

[Carroll und Rosson 1990] CARROLL, J. ; ROSSON, M.: Human computer interaction scenarios as a design representation. In: *In Proceedings of HICSS-23: Hawaii International Conference on System Sciences.* Los Alamitos, Ca. : IEEE Computer Society Press, 1990, S. 555–561

[Castela et al. 2001] CASTELA, N. ; TRIBOLET, J. ; SILVA, A. ; GUERRA, A.: Business Process Modeling with UML. In: *ICEIS (2)*, 2001, S. 679–685

[Chervany und Lending 1998] CHERVANY, N. ; LENDING, D.: CASE tools: understanding the reasons for non-use. In: *SIGCPR Comput. Pers.* 19 (1998), Nr. 2, S. 13–26

[Chin et al. 1997] CHIN, G. ; ROSSON, M. ; CARROLL, J.: Participatory analysis: shared development of requirements from scenarios. In: *CHI '97: Proceedings of the SIGCHI conference on Human factors in computing systems* (1997), S. 162–169

[Cohn 2004] COHN, M.: *User Stories Applied: For Agile Software Development*. Redwood City, CA, USA : Addison Wesley Longman Publishing Co., Inc., 2004

[Constantine 1995] CONSTANTINE, L.: Essential modeling: Use cases for user interfaces. In: *Interactions* 2 (1995), Nr. 2, S. 34–46

[Constantine 2000] CONSTANTINE, L.: Rapid Abstract Prototyping. In: AMBLER, S. (Hrsg.) ; CONSTANTINE, L. (Hrsg.): *The Unified Process Elaboration Phase: Best Practices in Implementing the UP.* CMP Books, 2000

[Constantine 2001] CONSTANTINE, L.: Process Agility and Software Usability: Toward Lightweight Usage-Centered Design / Constantine & Lockwood, Ltd. 2001 (110). – Report

[Constantine 2003] CONSTANTINE, L.: Canonical Abstract Prototypes for Abstract Visual and Interaction Design. In: *Interactive Systems. Design, Specification, and Verification* (2003), S. 1–15

[Constantine 2006a] CONSTANTINE, L.: Activity Modeling: Toward a Pragmatic Integration of Activity Theory with Usage-Centered Design / Laboratory for Usage-centered Software Engineering - LabUSE Technical Paper. Madeira, Portugal, 2006. – Forschungsbericht

[Constantine 2006b] CONSTANTINE, L.: Users, Roles, and Personas. In: PRUITT, J. (Hrsg.) ; ADLIN, T. (Hrsg.): *The Persona Lifecycle: Keeping People in Mind Throughout Product Design (The Morgan Kaufmann Series in Interactive Technologies)*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2006

[Constantine et al. 2003] CONSTANTINE, L. ; BIDDLE, R. ; NOBLE, J.: Usage-Centered Design and Software Engineering: Models for Integration. In: *ICSE Workshop on SE-HCI*, 2003, S. 106–113

[Constantine und Campos 2005] CONSTANTINE, L. ; CAMPOS, P.: CanonSketch and TaskSketch: innovative modeling tools for usage-centered design. In: *OOPSLA '05: Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*. New York, NY, USA : ACM, 2005, S. 162–163

[Constantine und Lockwood 1999a] CONSTANTINE, L. ; LOCKWOOD, L.: *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design (ACM Press)*. Addison-Wesley Professional, April 1999

[Constantine und Lockwood 1999b] CONSTANTINE, L. ; LOCKWOOD, L.: Use cases in task modeling and user interface design. In: *CHI '99: CHI '99 extended abstracts on Human factors in computing systems*. New York, NY, USA : ACM, 1999, S. 352

[Constantine und Lockwood 2002] CONSTANTINE, L. ; LOCKWOOD, L.: Usage-Centered Engineering for Web Applications. In: *IEEE Softw.* 19 (2002), Nr. 2, S. 42–50

[Constantine und Lockwood 2003] CONSTANTINE, L. ; LOCKWOOD, L.: Usage-centered software engineering: an agile approach to integrating users, user interfaces, and usability into software engineering practice. In: *ICSE '03: Proceedings of the 25th International Conference on Software Engineering*. Washington, DC, USA : IEEE Computer Society, 2003, S. 746–747

[Cook und Bailey 2005] COOK, D. ; BAILEY, B.: Designers' use of paper and the implications for informal tools. In: *OZCHI '05: Proceedings of the 19th conference of the computer-human interaction special interest group (CHISIG) of Australia on Computer-human interaction*. Narrabundah, Australia, Australia : Computer-Human Interaction Special Interest Group (CHISIG) of Australia, 2005, S. 1–10

[Cooper 2004] COOPER, A.: *The Inmates Are Running the Asylum : Why High Tech Products Drive Us Crazy and How to Restore the Sanity (2nd Edition)*. Sams, February 2004

[Coyette et al. 2007a] COYETTE, A. ; KIEFFER, S. ; VANDERDONCKT, J.: Multi-fidelity Prototyping of User Interfaces. In: *INTERACT (1)*, 2007, S. 150–164

[Coyette und Vanderdonckt 2005] COYETTE, A. ; VANDERDONCKT, J.: A Sketching Tool for Designing Anyuser, Anyplatform, Anywhere User Interfaces. In: COSTABILE, Maria F. (Hrsg.) ; PATERNÒ, Fabio (Hrsg.): *INTERACT* Bd. 3585, Springer, 2005, S. 550–564

[Coyette et al. 2007b] COYETTE, A. ; VANDERDONCKT, J. ; LIMBOURG, Q.: SketchiXML: A Design Tool for Informal User Interface Rapid Prototyping. In: *Rapid Integration of Software Engineering Techniques*. Springer Berlin / Heidelberg, 2007, S. 160–176

[da Silva und Paton 2001] DA SILVA, P. ; PATON, N.: *A UML-Based Design Environment for Interactive Applications*. 2001

[da Silva und Paton 2000a] DA SILVA, P. P. ; PATON, N. W. ; EVANS, Andy (Hrsg.) ; KENT, Stuart (Hrsg.) ; SELIC, Bran (Hrsg.): *UMLi: The Unified Modeling Language for Interactive Applications*. 2000

[da Silva und Paton 2000b] DA SILVA, P. P. ; PATON, N. W.: User Interface Modelling with UML. In: *In Proceedings of the 10th European-Japanese Conference on Information Modelling and Knowledge Representation*, IOS Press, 2000, S. 203–217

[da Silva und Paton 2003] DA SILVA, P. P. ; PATON, N. W.: User interface modeling in UMLi. In: *Software, IEEE* 20 (2003), Nr. 4, S. 62–69

[Damm et al. 2000] DAMM, C. ; HANSEN, K. ; THOMSEN, M. ; TYRSTED, M.: Creative Object-Oriented Modelling: Support for Intuition, Flexibility, and Collaboration in CASE Tools. In: *ECOOP '00: Proceedings of the 14th European Conference on Object-Oriented Programming*. London, UK : Springer-Verlag, 2000, S. 27–43

[Davis 1993] DAVIS, Alan M.: Software requirements: objects, functions, and states. Upper Saddle River, NJ, USA : Prentice-Hall, Inc., 1993. – Forschungsbericht

[Düchting et al. 2007] DÜCHTING, M. ; ZIMMERMANN, D. ; NEBE, K.: Incorporating User Centered Requirement Engineering into Agile Software Development. In: JACKO, Julie A. (Hrsg.): *HCI (1)* Bd. 4550, Springer, 2007, S. 58–67

[Dix 1991] DIX, A.: *Formal methods and interactive systems: principles and practice*. Academic Press, 1991

[Dix 1995a] DIX, A. ; MONK, A. (Hrsg.) ; GILBERT, N. (Hrsg.): *Formal methods an introduction to and overview of the use of formal methods within hci*. London Academic Press, 1995. – 9 – 43 S

[Dix 1995b] DIX, A. ; MONK, A. (Hrsg.) ; GILBERT, N. (Hrsg.): *Formal Methods in HCI In: Perspectives on HCI: Diverse Approaches*. London, Academic Press., 1995. – pp. 9–43 S

[Dix et al. 2003] DIX, A. ; FINLAY, J. ; ABOWD, G. ; BEALE, R.: *Human-Computer Interaction (3rd Edition)*. Prentice Hall, December 2003

[Douglas et al. 2002] DOUGLAS, S. ; TREMAINE, M. ; LEVENTHAL, L. ; WILLS, C. ; MANARIS, B.: Incorporating Human-Computer Interaction into the undergraduate computer science curriculum. In: *SIGCSE '02: Proceedings of the 33rd SIGCSE technical symposium on Computer science education* Bd. 34. New York, NY, USA : ACM, 2002, S. 211–212

[Duke und Harrison 1995] DUKE, D. ; HARRISON, M.: From Formal Models to Formal Methods. In: *ICSE '94: Proceedings of the Workshop on Software Engineering and Human-Computer Interaction*. London, UK : Springer-Verlag, 1995, S. 159–173

[Ehn 1992] EHN, P.: Scandinavian design: On participation and skill. (1992), S. 96–132

[Elkoutbi et al. 1999] ELKOUTBI, M. ; KHRISS, I. ; KELLER, R. K.: Generating user interface prototypes from scenarios. In: *Requirements Engineering, 1999. Proceedings. IEEE International Symposium on*, 1999, S. 150–158

[Elnaffar und Graham 1999] ELNAFFAR, S. ; GRAHAM, N.: Semi-automated linking of user interface design artifacts. In: *Proceedings of the third international conference on Computer-aided design of user interfaces*. Norwell, MA, USA : Kluwer Academic Publishers, 1999, S. 127–138

[Falessi et al. 2006] FALESSI, D. ; CANTONE, G. ; BECKER, M.: Documenting design decision rationale to improve individual and team design decision making: an experimental evaluation. In: *ISESE '06: Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*. New York, NY, USA : ACM, 2006, S. 134–143

[Ferreira et al. 2007] FERREIRA, J. ; NOBLE, J. ; BIDDLE, R.: Agile Development Iterations and UI Design. In: *AGILE 2007* (2007), S. 50–58

[Fields et al. 1994] FIELDS, B. ; HARRISON, M. ; WRIGHT, P.: From informal requirements to agent-based specification. In: *SIGCHI Bull.* 26 (1994), Nr. 2, S. 65–68

[Finkelstein 1993] FINKELSTEIN, A.: Requirements Engineering: An Overview. In: *2nd Asia-Pacific Software Engineering Conference (APSEC'93)*. Tokyo, Japan, 1993

[Fitton et al. 2005] FITTON, D. ; CHEVERST, K. ; KRAY, C. ; DIX, A. ; ROUNCEFIELD, M. ; SASLIS-LAGOUDAKIS, G.: Rapid Prototyping and User-Centered Design of Interactive Display-Based Systems. In: *IEEE Pervasive Computing* 4 (2005), Nr. 4, S. 58–66

[Floyd 1984] FLOYD, C.: A Systematic Look at Prototyping. In: BUDDE, R. (Hrsg.) ; KUHLENKAMP, K. (Hrsg.) ; MATHIASSEN, Lars (Hrsg.) ; ZULLIGHOVEN, H. (Hrsg.): *Approaches to Prototyping*. Springer-Verlag, 1984, S. 1–17

[Frank 1995] FRANK, M.: *Model-Based User Interface Design By Demonstration and By Interview*, Georgia Institute of Technology, Dissertation, 1995

[Furnas und Bederson 1995] FURNAS, G. ; BEDERSON, B.: Space-scale diagrams: understanding multiscale interfaces. In: *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM Press/Addison-Wesley Publishing Co., 1995, S. 234–241

[Furniss et al. 2005] FURNISS, D. ; DIX, A. ; PONSARD, C. ; ZHANG, G.: Outdated Ideas of the Design Process and the Future of Formal Models, Methods and Notations. In: *DSV-IS*, 2005, S. 265

[Geiger und Müller 1998] GEIGER, C. ; MÜLLER, W.: Visuelle Spezifikation, Modellierung und Animation im Systementwurf. In: *SimVis*, 1998, S. 206–220

[Geyer 2008] GEYER, F.: *A Zoom-Based Specification Tool supporting Interdisciplinary User Interface Design Processes*, University of Konstanz, Master Thesis, 2008

[Glinz 2000] GLINZ, M.: Problems and Deficiencies of UML as a Requirements Specification Language. In: *IEEE Tenth International Workshop on Software Specification and Design*, 2000, S. 11–22

[Goldberg und Robson 1983] GOLDBERG, A. ; ROBSON, D.: *Smalltalk-80: the language and its implementation*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1983

[Gotel und Finkelstein 1994] GOTEL, O. ; FINKELSTEIN, C.: An analysis of the requirements traceability problem. In: *Requirements Engineering, 1994., Proceedings of the First International Conference on* (1994), Apr, S. 94–101

[Gould und Lewis 1985] GOULD, J. ; LEWIS, C.: Designing for usability: Key principles and what designers think. In: *Commun. ACM* 28 (1985), Nr. 3, S. 300–311

[Granic und Glavinic 2002] GRANIC, A. ; GLAVINIC, V.: User interface specification issues for computerized educational systems. In: *Information Technology Interfaces, 2002. ITI 2002. Proceedings of the 24th International Conference on* (2002)

[Gross und Yi-Luen Do 1996] GROSS, M. ; YI-LUEN DO, E.: Ambiguous intentions: a paper-like interface for creative design. In: *UIST '96: Proceedings of the 9th annual ACM symposium on User interface software and technology*. New York, NY, USA : ACM, 1996, S. 183–192

[Gruhn und Wellen 1998] GRUHN, V. ; WELLEN, U.: From business process models to distributed software architecture. In: *ISAW '98: Proceedings of the third international workshop on Software architecture*. New York, NY, USA : ACM, 1998, S. 53–56

[Gunaratne et al. 2004] GUNARATNE, J. ; HWONG, C. ; RUDORFER, A.: Using evolutionary prototypes to formalize product requirements. In: *Bridging the Gaps II: Bridging the Gaps Between Software Engineering and Human-Computer Interaction: W1L Workshop*, 2004, S. 17–20

[Gundelsweiler et al. 2004] GUNDELSWEILER, F. ; MEMMEL, T. ; REITERER, H.: Agile Usability Engineering. In: *Proceedings of the 4th Mensch & Computer conference (MCI 2007, Paderborn, Germany)*, Oldenbourg Verlag, 2004, S. 33–42

[Hanna 2005] HANNA, P.: *Customer storytelling at the heart of business success*. Boxes and Arrows. 2005. – URL `http://www.boxesandarrows.com/view/customer_storytelling_at_the_heart_of_business_success`. – (Zugriff: 15 Sep 2008).

[Hardtke 2004] HARDTKE, F.: Rapid Prototyping for User-Friendly and Useful Human-Machine Interfaces. / Raytheon Australia Pty Ltd Naval and Maritime Integrated Systems. Canberra, Australia., 2004. – Forschungsbericht

[Hartson et al. 1990] HARTSON, R. ; SIOCHI, A. ; HIX, D.: The UAN: a user-oriented representation for direct manipulation interface designs. In: *ACM Trans. Inf. Syst.* 8 (1990), Nr. 3, S. 181–203

[Haumer 2000] HAUMER, P.: *Requirements Engineering with Interrelated Conceptual Models and Real World Scenes*, Technische Hochschule Aachen, Dissertation, 2000

[Heimdahl und Thompson 2000] HEIMDAHL, M. ; THOMPSON, J.: Specification based prototyping of control systems. In: *19th IEEE Digital Avionics Systems Conference*. Philadelphia, October 2000

[Heinilä et al. 2005] HEINILÄ, J. ; STRÖMBERG, H. ; LEIKAS, J. ; IKONEN, V. ; IIVARI, N. ; JOKELA, T. ; AIKIO, K. ; JOUNILA, I. ; HOONHOUT, J. ; LEURS, N.: *Nomadic Media (ITEA if02019): User-Centred Design - Guidelines for Methods and Tools*. 2005. – URL `http://www.vtt.fi/inf/julkaisut/muut/2005/UCD_Guidelines.pdf`. – (Zugriff: 15 Aug 2008)

[Hill und Bartek 2007] HILL, V. ; BARTEK, V.: Telling the user's story. In: *CHIMIT '07: Proceedings of the 2007 symposium on Computer human interaction for the management of information technology*. New York, NY, USA : ACM, 2007, S. 6

[Hix und Hartson 1993] HIX, D. ; HARTSON, R.: *Developing user interfaces: Ensuring usability through product and process*. Wiley Professional Computing, June 1993

[Hofmann und Lehner 2001] HOFMANN, H. ; LEHNER, F.: Requirements engineering as a success factor in software projects. In: *Software, IEEE* 18 (2001), Nr. 4, S. 58–66

[Holt 2005] HOLT, J.: *A Pragmatic Guide to Business Process Modelling*. London : British Computer Society, 2005

[Holtzblatt und Beyer 1995] HOLTZBLATT, K. ; BEYER, H.: Requirements gathering: the human factor. In: *Commun. ACM* 38 (1995), Nr. 5, S. 31–32

[Hong et al. 2001] HONG, J. ; LI, F. ; LIN, J. ; LANDAY, J.: End-user perceptions of formal and informal representations of web sites. In: *CHI '01: CHI '01 extended abstracts on Human factors in computing systems*. New York, NY, USA : ACM, 2001, S. 385–386

[Horrocks 1999] HORROCKS, I.: *Constructing the user interface with statecharts*. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 1999

[Houde und Hill 1997] HOUDE, S. ; HILL, C.: *What do Prototypes Prototype?* Elsevier Science B.V. 1997

[Hudson 1994] HUDSON, S.: User interface specification using an enhanced spreadsheet model. In: *ACM Trans. Graph.* 13 (1994), Nr. 3, S. 209 – 239

[Hussey 2000] HUSSEY, A.: Formal object-oriented user-interface design. In: *ASWEC '00: Proceedings of the 2000 Australian Software Engineering Conference*. Washington, DC, USA : IEEE Computer Society, 2000, S. 129

[Hussey und Carrington 1996] HUSSEY, A. ; CARRINGTON, D.: Using Object-Z to specify a web browser interface. In: *OZCHI '96: Proceedings of the 6th Australian Conference on Computer-Human Interaction (OZCHI '96)*. Washington, DC, USA : IEEE Computer Society, 1996, S. 236

[IEEE 1998a] IEEE, Std 1.: IEEE Guide for developing system requirements specifications. In: *IEEE Std 1233, 1998 Edition* (1998)

[IEEE 1990] IEEE, Std 6.: IEEE Standard glossary of software engineering terminology. In: *IEEE Std 610.12-1990* (1990)

[IEEE 1998b] IEEE, Std 8.: IEEE Recommended practice for software requirements specifications. In: *IEEE Std 830-1998* (1998)

[Iivari 1996] IIVARI, J.: Why are CASE tools not used? In: *Commun. ACM* 39 (1996), Nr. 10, S. 94–103

[ISO/IEC 1999] ISO/IEC, 13407: Human-centred design processes for interactive systems. In: *ISO/IEC 13407: 1999* (1999)

[ISO/IEC 2002] ISO/IEC, 13568: Information technology - Z formal specification notation - Syntax, type system and semantics. In: *ISO/IEC 13568: 2002* (2002)

[ISO/IEC 1998] ISO/IEC, 9241-14: Ergonomic requirements for office work with visual display terminals. In: *ISO/IEC 9241-14: 1998* (1998)

[Jackson 1998] JACKSON, M.: Defining a discipline of description. In: *IEEE Softw.* 15 (1998), Nr. 5, S. 14–17

[Jacob 1983] JACOB, R.: Using formal specifications in the design of a human-computer interface. In: *Commun. ACM* 26 (1983), Nr. 4, S. 259–264

[Jacob 1986] JACOB, R.: A specification language for direct-manipulation user interfaces. In: *ACM Trans. Graph.* 5 (1986), Nr. 4, S. 283–317

[Jacobson 2004] JACOBSON, I.: *Object-oriented software engineering: A use case driven approach*. Redwood City, CA, USA : Addison Wesley Longman Publishing Co., Inc., 2004

[Jacobson et al. 1999] JACOBSON, I. ; BOOCH, G. ; RUMBAUGH, J.: *The unified software development process*. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 1999

[Jarczyk et al. 1992] JARCZYK, A. ; LOFFLER, P. ; SHIPMANN, F.: Design rationale for software engineering: A survey. In: *Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences* 2 (1992), Jan, S. 577–586

[Jarke 1999] JARKE, M.: Scenarios for modeling. In: *Commun. ACM* 42 (1999), Nr. 1, S. 47–48

[Jarzabek und Huang 1998] JARZABEK, S. ; HUANG, R.: The case for user-centered CASE tools. In: *Commun. ACM* 41 (1998), Nr. 8, S. 93–99

[Jokela et al. 2003] JOKELA, T. ; IIVARI, N. ; MATERO, J. ; KARUKKA, M.: The standard of user-centered design and the standard definition of usability: analyzing ISO 13407 against ISO 9241-11. In: *CLIHC '03: Proceedings of the Latin American conference on Human-computer interaction*. New York, NY, USA : ACM, 2003, S. 53–60

[Kemerer 1992] KEMERER, C.: How the Learning Curve Affects CASE Tool Adoption. In: *IEEE Softw.* 9 (1992), Nr. 3, S. 23–28

[Kensing und Blomberg 1998] KENSING, F. ; BLOMBERG, J.: Participatory Design: Issues and concerns. In: *Comput. Supported Coop. Work* 7 (1998), Nr. 3-4, S. 167–185

[Khazaei und Roast 2001] KHAZAEI, B. ; ROAST, C.: *The usability of formal specification representations*. 2001

[Kholkar et al. 2005] KHOLKAR, D. ; KRISHNA, M. ; SHROTRI, U. ; VENKATESH, R.: Visual specification and analysis of use cases. In: *SoftVis '05: Proceedings of the 2005 ACM symposium on Software visualization*. New York, NY, USA : ACM, 2005, S. 77–85

[Kimmond 1995] KIMMOND, R. M.: Survey into the acceptance of prototyping in software development. In: *rsp* 00 (1995)

[Klemmer et al. 2006] KLEMMER, S. ; HARTMANN, B. ; TAKAYAMA, L.: How bodies matter: five themes for interaction design. In: *DIS '06: Proceedings of the 6th conference on Designing Interactive systems*. New York, NY, USA : ACM, 2006, S. 140–149

[Kobsa 1995] KOBSA, A.: Supporting User Interfaces for All Through User Modeling. In: *Proceedings HCI International 95*, Elsevier, 1995, S. 155–157

[König 2008] KÖNIG, D.: *Anforderungen an Requirements Engineering Werkzeuge für die visuelle Spezifikation*, University of Konstanz, Master Thesis, April 2008

[Kwon et al. 2007] KWON, G. ; HAM, D. ; YOON, W.: Evaluation of software usability using scenarios organized by abstraction structure. In: *ECCE '07: Proceedings of the 14th European conference on Cognitive ergonomics*. New York, NY, USA : ACM, 2007, S. 19–22

[Landay und Myers 1995] LANDAY, J. ; MYERS, B.: Interactive sketching for the early stages of user interface design. In: *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM Press/Addison-Wesley Publishing Co., 1995, S. 43–50

[Landay und Myers 2001] LANDAY, J. A. ; MYERS, B. A.: Sketching interfaces: toward more human interface design. In: *Computer* 34 (2001), Nr. 3, S. 56–64

[Larman 2003] LARMAN, C.: *Agile and iterative development: A manager's guide*. Pearson Education, 2003

[Lecolinet 1998] LECOLINET, E.: Designing GUIs by sketch drawing and visual programming. In: *AVI '98: Proceedings of the working conference on Advanced visual interfaces*. New York, NY, USA : ACM, 1998, S. 274–276

[Lee 1997] LEE, J.: Design rationale systems: understanding the issues. In: *IEEE Expert* 12 (1997), Nr. 3, S. 78–85

[Lim und Sato 2006] LIM, Y. K. ; SATO, K.: Describing multiple aspects of use situation: applications of Design Information Framework (DIF) to scenario development'. In: *Design Studies* Bd. 27, 2006, S. 57–76

[Lin und Landay 2003] LIN, J. ; LANDAY, J.: Damask: A Tool for Early-Stage Design and Prototyping of Cross-Device User Interfaces. In: *CHI 2003 workshop on HCI Patterns: Concepts and Tools, Fort Lauderdale, Florida* (2003), S. 573–580

[Lin et al. 2000] LIN, James ; NEWMAN, Mark W. ; HONG, Jason I. ; LANDAY, James A.: DENIM: finding a tighter fit between tools and practice for Web site design. In: *CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM Press, 2000, S. 510–517

[Lopez-Nores et al. 2006] LOPEZ-NORES, M. ; PAZOS-ARIAS, J. ; GARCIA-DUQUE, J. ; BARRAGANS-MARTINEZ, Belen: An Agile Approach to Support Incremental Development of Requirements Specifications. In: *ASWEC '06: Proceedings of the Australian Software Engineering Conference*. Washington, DC, USA : IEEE Computer Society, 2006, S. 9–18

[Loucopoulos und Karakostas 1995] LOUCOPOULOS, P. ; KARAKOSTAS, V.: *System Requirements Engineering*. New York, NY, USA : McGraw-Hill, Inc., 1995

[Löwgren 1995] LÖWGREN, J.: Perspectives on Usability / University of Linköping. 1995 (R-95-23). – Forschungsbericht

[Lozano et al. 2001] LOZANO, M. ; GONZÁLEZ, P. ; RAMOS, I.: User interface generation: Current trends. In: *Informatik/informatique, No. 2. Swiss Informatics Society, Abril.* (2001)

[Lozano et al. 2002] LOZANO, M. ; GONZÁLEZ, P. ; MONTERO, F. ; MOLINA, J. ; RAMOS, I.: *Integrating Usability within the User Interface Development Process of Web Applications*, Springer LNCS 2374, Springer LNCS 2374, 2002

[Mackay et al. 2003] MACKAY, D. ; NOBLE, J. ; BIDDLE, R.: A lightweight web-based case tool for UML class diagrams. In: *AUIC '03: Proceedings of the Fourth Australasian user interface conference on User interfaces 2003*. Darlinghurst, Australia, Australia : Australian Computer Society, Inc., 2003, S. 95–98

[Markopoulos und Marijnissen 2000] MARKOPOULOS, P. ; MARIJNISSEN, P.: UML as a representation for interaction design. In: *CHISIG 2000: Australian Conf. of Computer-Human Interaction*, 2000, S. 240–249

[Mayhew 1999] MAYHEW, D.: *The usability engineering lifecycle: A practitioner's handbook for user interface design*. Morgan Kaufmann, April 1999

[Mayhew 2005] MAYHEW, D.: *Cost-Justifying usability: An update for the internet age, Second Edition*. Morgan Kaufmann, April 2005

[McCoy 2002] MCCOY, T.: Letter from the dark side: confessions of an applications developer. In: *interactions* 9 (2002), Nr. 6, S. 11–15

[McCracken und Jackson 1982] MCCRACKEN, Daniel D. ; JACKSON, Michael A.: Life cycle concept considered harmful. In: *SIGSOFT Softw. Eng. Notes* 7 (1982), Nr. 2, S. 29–32

[McInerney und Sobiesiak 2000] MCINERNEY, P. ; SOBIESIAK, R.: The UI design process. In: *SIGCHI Bull.* 32 (2000), Nr. 1, S. 17–21

[Memmel et al. 2007a] MEMMEL, T. ; BOCK, C. ; REITERER, H.: Model-driven prototyping for corporate software specification. In: *Proceedings of the EHCI-HCSE-DSVIS'07*, Mar 2007

[Memmel et al. 2008a] MEMMEL, T. ; GEYER, F. ; RINN, J. ; REITERER, H.: Tool-Support for Interdisciplinary and Collaborative User Interface Specification. In: *To be published in: Proceedings of the IADIS International Conference on Interfaces and Human Computer Interaction (IHCI 2008, Amsterdam, The Netherlands)*, Jul 2008, S. 51–60

[Memmel et al. 2008b] MEMMEL, T. ; GEYER, F. ; RINN, J. ; REITERER, H.: A Zoom-Based Specification Tool for Corporate User Interface Development. In: *In proceedings of the IADIS International Conference on Interfaces and Human Computer Interaction (IHCI 2008, Amsterdam, The Netherlands)*, Jul 2008, S. 368–370

[Memmel et al. 2007b] MEMMEL, T. ; GUNDELSWEILER, F. ; REITERER, H.: Agile Human-Centered Software Engineering. In: *Proceedings of 21st BCS HCI Group conference (HCI 2007, University of Lancater, UK)*, British Computer Society, 2007, S. 167–175

[Memmel et al. 2007c] MEMMEL, T. ; GUNDELSWEILER, F. ; REITERER, H.: Prototyping Corporate User Interfaces - Towards A Visual Specification Of Interactive Systems. In: *Proceedings of the 2nd IASTED International Conference on Human Computer Interaction (IASTED-HCI '07), Chamonix, France: Proceedings*, Acta Press, Canada, Mar 2007, S. 177–182

[Memmel und Heilig 2007] MEMMEL, T. ; HEILIG, M.: Model-Based visual software specification. In: *Proceedings of the IADIS International Conference on Interfaces and Human Computer Interaction (IHCI 2007, Lisbon, Portugal)*, IADIS Press, Lisbon, Portugal, Jul 2007, S. 3–10

[Memmel und Reiterer 2007] MEMMEL, T. ; REITERER, H.: Visuelle Spezifikation interaktiver Systeme mit Modell- und XML-basierten Prototyping-Werkzeugen und Werkzeugketten. In: *Proceedings of the 1st conference on Software Engineering Essentials (SEE 2007, Munich, Germany)*, IfI Technical Report Series IfI-07-07, 2007, S. 78–92

[Memmel und Reiterer 2008a] MEMMEL, T. ; REITERER, H.: Inspector: Interactive UI Specification Tool. In: *In proceedings of the 7th International Conference On Computer Aided Design of User Interfaces (CADUI) 2008*, Springer, Jun 2008, S. 161–174

[Memmel und Reiterer 2008b] MEMMEL, T. ; REITERER, H.: Inspector: Method and Tool for Visual UI Specification. In: *Proceedings of the 3rd IASTED International Conference on Human Computer Interaction (IASTED-HCI '08)*, Acta Press, Canada, Mar 2008, S. 170–179

[Memmel und Reiterer 2008c] MEMMEL, T. ; REITERER, H.: User Interface Entwicklung mit interaktiven Spezifikationen. In: *In Proceedings of Mensch & Computer 2008: Viel mehr Interaktion, 8. Konferenz für interaktive und kooperative Medien, Oldenbourg Verlag*, Sep 2008, S. 357–366

[Memmel et al. 2007d] MEMMEL, T. ; REITERER, H. ; HOLZINGER, A.: Agile Methods and Visual Specification in Software Development: a chance to ensure Universal Access. In: *Proceedings of the 12th International Conference on Human-Computer Interaction (HCII 2007, Beijing, China)*, Springer-Verlag, Berlin, Heidelberg 2007, 2007

[Memmel et al. 2007e] MEMMEL, T. ; REITERER, H. ; ZIEGLER, H. ; OED, R.: Visuelle Spezifikation zur Stärkung der Auftraggeberkompetenz bei der Gestaltung interaktiver Systeme. In: *Proceedings of 5th Workshop of the German Chapter of the Usability Professionals Association e.V.*, Frauenhofer IRB Verlag, Stuttgart, 2007, S. 99–104

[Memmel et al. 2008c] MEMMEL, T. ; VANDERDONCKT, J. ; H., Reiterer: Multi-fidelity User Interface Specifications. In: *In Proceedings of the 15th International Workshop on the Design, Verification and Specification of Interactive Systems (DSV-IS 2008)*, Jul 2008, S. 43–57

[Miller 2006] MILLER, R.: *Practical UML: A Hands-On Introduction for Developers*. CodeGear from Borland Developer Network. 2006. – URL `http://dn.codegear.com/article/31863`. – (Zugriff: 15 Okt 2008)

[Minocha 1999] MINOCHA, S.: Requirements development in user-centred system design. In: *Making User-Centred Design Work in Software Development (Ref. No. 1999/010)* (1999)

[Monk et al. 1993] MONK, A. ; CURRY, M. ; WRIGHT, P. ; GILMORE, D. (Hrsg.): *Why industry doesn't use the wonderful notations we researchers have given them to reason about their designs.* Berlin, Germany : Springer Verlag, 1993. – 253 – 261 S

[Monk et al. 1995] MONK, S. ; SOMMERVILLE, I. ; PENDARIES, J. ; DURIN, B.: Supporting Design Rationale for System Evolution. In: *Proceedings of the 5th European Software Engineering Conference*. London, UK : Springer-Verlag, 1995, S. 307–323

[Moore 2003] MOORE, J.: Communicating requirements using end-user GUI constructions with argumentation. In: *Automated Software Engineering, 2003. Proceedings. 18th IEEE International Conference on* (2003), S. 360–363

[Muller 2003] MULLER, M.: Participatory design: the third space in HCI. (2003), S. 1051–1068

[Muller und Kuhn 1993] MULLER, M. ; KUHN, S.: Participatory design. In: *Commun. ACM* 36 (1993), Nr. 6, S. 24–28

[Myers 1995] MYERS, B.: User Interface Software Tools. In: *ACM Transactions on Computer-Human Interaction* 2 (1995), S. 64–103

[Myers und Buxton 1986] MYERS, B. ; BUXTON, W.: Creating highly-interactive and graphical user interfaces by demonstration. In: *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*. New York, NY, USA : ACM, 1986, S. 249–258

[Myers und Rosson 1992] MYERS, B. ; ROSSON, M.: Survey on user interface programming. In: *CHI '92: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM Press, 1992, S. 195–202

[Newman und Landay 2000] NEWMAN, M. ; LANDAY, J.: Sitemaps, storyboards, and specifications: a sketch of Web site design practice. In: *DIS '00: Proceedings of the 3rd conference on Designing interactive systems*. New York, NY, USA : ACM, 2000, S. 263–274

[Nielsen 1994] NIELSEN, J.: *Usability Engineering*. Morgan Kaufmann, September 1994

[Nielsen 1992] NIELSEN, Jakob: The Usability Engineering Life Cycle. In: *Computer* 25 (1992), Nr. 3, S. 12 – 22

[Norman und Draper 1986] NORMAN, D. ; DRAPER, S.: *User Centered System Design: New Perspectives on Human-computer Interaction*. TF-CRC, January 1986

[Nunes 2001] NUNES, N.: *Object Modeling for User-Centered Development and User Interface Design: The Wisdom Approach*, Universidade da Madeira, Dissertation, 2001

[Nunes und Cunha 2000a] NUNES, N. J. ; CUNHA, J. F.: Towards a UML profile for interaction design: the Wisdom approach. In: EVANS, Andy (Hrsg.) ; KENT, Stuart (Hrsg.) ; SELIC, Bran (Hrsg.): *UML 2000 - The Unified Modeling Language. Advancing the Standard. Third International Conference, York, UK, October 2000, Proceedings* Bd. 1939, Springer, 2000, S. 101–116

[Nunes und Cunha 2000b] NUNES, N. J. ; CUNHA, J. F.: Wisdom: a software engineering method for small software development companies. In: *Software, IEEE* 17 (2000), Nr. 5, S. 113–119

[Nunes und Cunha 2001] NUNES, N. J. ; CUNHA, J. F.: Wisdom - A UML Based Architecture for Interactive Systems. In: *Lecture Notes in Computer Science* 1946 (2001), S. 191–205

[Nuseibeh und Easterbrook 2000] NUSEIBEH, Bashar ; EASTERBROOK, Steve: Requirements engineering: a roadmap. In: *ICSE '00: Proceedings of the Conference on The Future of Software Engineering*. New York, NY, USA : ACM, 2000, S. 35–46

[Obendorf und Finck 2008] OBENDORF, H. ; FINCK, M.: Scenario-based usability engineering techniques in agile development processes. In: *CHI '08: CHI '08 extended abstracts on Human factors in computing systems*. New York, NY, USA : ACM, 2008, S. 2159–2166

[Oestereich 2005] OESTEREICH, B.: Objektorientierte Geschäftsprozessmodellierung und modellgetriebene Softwareentwicklung. In: STRAHRINGER, S. (Hrsg.): *Business Engineering*. HMD - Praxis der Wirtschaftsinformatik, 2005, S. 27 – 34

[Oestereich et al. 2003] OESTEREICH, B. ; WEISS, C. ; SCHRÖDER, C. ; WEILKIENS, T. ; LENHARD, A.: *Objektorientierte Geschäftsprozessmodellierung mit der UML*. dpunkt.verlag Heidelberg, 2003

[Offergeld und Oed 2006] OFFERGELD, M. ; OED, R.: Usability Engineering als Auftraggeberkompetenz. In: *Proceedings Usability Professionals. Tim Bosenick, Marc Hassenzahl, Matthias Müller-Prove, Matthias Peissner*, German Chapter der Usability Professionals Association e.V., Stuttgart, 2006

[Okawa et al. 2007] OKAWA, T. ; HIRABAYASHI, S. ; KAMINISHI, T. ; KOIZUMI, H. ; SAWAMOTO, J.: A Method of Linking Business Process Modeling with Information System Design Using UML and its Evaluation by Prototyping. In: *APSCC '07: Proceedings of the The 2nd IEEE Asia-Pacific Service Computing Conference*. Washington, DC, USA : IEEE Computer Society, 2007, S. 458–465

[O'Neill und Johnson 2004] O'NEILL, E. ; JOHNSON, P.: Participatory task modelling: users and developers modelling users' tasks and domains. In: *TAMODIA '04: Proceedings of the 3rd annual conference on Task models and diagrams*. New York, NY, USA : ACM, 2004, S. 67–74

[Osterweil 2007] OSTERWEIL, L.: A Future for Software Engineering? In: *FOSE '07: 2007 Future of Software Engineering*. Washington, DC, USA : IEEE Computer Society, 2007, S. 1–11

[Paech und Kohler 2003] PAECH, B. ; KOHLER, K.: Usability Engineering integrated with Requirements Engineering. In: KAZMAN, Rick (Hrsg.) ; BASS, Len (Hrsg.) ; BOSCH, Jan (Hrsg.): *ICSE Workshop on SE-HCI*, IFIP, 2003, S. 36–40

[Paternò 1999] PATERNÒ, F.: Special Issue on Formal Methods for Visual Interaction. In: *Journal of Visual Languages and Computing* 10 (1999), S. 451 – 453

[Paternò 2001] PATERNÒ, F.: Towards a UML for Interactive Systems. In: *EHCI '01: Proceedings of the 8th IFIP International Conference on Engineering for Human-Computer Interaction*. London, UK : Springer-Verlag, 2001, S. 7–18

[Paternò 2003] PATERNÒ, F.: From Model-based to Natural Development. In: *Proceedings of the Tenth International Conference on Human-Computer Interaction*, Lawrence Erlbaum Associates, 2003, S. 592–596

[Paternò und Volpe 2005] PATERNÒ, F. ; VOLPE, M.: Natural Modelling of Interactive Applications. In: *DSV-IS*, 2005, S. 67–77

[Petersen und Wiil 2008] PETERSEN, R. ; WIIL, U.: Asap: a planning tool for agile software development. In: *HT '08: Proceedings of the nineteenth ACM conference on Hypertext and hypermedia*. New York, NY, USA : ACM, 2008, S. 27–32

[Petrie und Schneider 2006] PETRIE, J. ; SCHNEIDER, K.: Mixed-Fidelity Prototyping of User Interfaces. In: *DSV-IS*, 2006, S. 199–212

[Pfaff 1985] PFAFF, G.: *User Interface Management Systems*. Springer Verlag, 1985

[Phanouriou 2000] PHANOURIOU, C.: *UIML: A Device-Independent User Interface Markup Language*, Virginia Polytechnic Institute and State University, Dissertation, 2000

[Phillips und Joe 2005] PHILLIPS, C. ; JOE, R.: Supporting content design of interaction spaces. In: *CHINZ '05: Proceedings of the 6th ACM SIGCHI New Zealand chapter's international conference on Computer-human interaction*. New York, NY, USA : ACM, 2005, S. 39–44

[Phillips und Kemp 2002] PHILLIPS, C. ; KEMP, E.: In support of user interface design in the rational unified process. In: *Aust. Comput. Sci. Commun.* 24 (2002), Nr. 4, S. 21–27

[Pohl 1997] POHL, K.: Requirements Engineering: An Overview. In: KENT, A. (Hrsg.) ; WILLIAMS, J. (Hrsg.): *Encyclopedia of Computer Science and Technology* Bd. 36. New York, NY, USA : Marcel Dekker, Inc., 1997

[Pohl 2007] POHL, K.: *Requirements Engineering. Grundlagen, Prinzipien, Techniken*. Dpunkt Verlag, 2007

[Preece et al. 1994] PREECE, J. ; ROGERS, Y. ; SHARP, H. ; BENYON, D. ; HOLLAND, S. ; CAREY, T.: *Human-Computer Interaction: Concepts And Design (ICS)*. Addison Wesley, April 1994

[Pyla et al. 2004] PYLA, P. S. ; A., Pèrez-Quiñones M. ; ARTHUR, J. D. ; HARTSON, H. R.: What we should teach, but don't: proposal for cross pollinated HCI-SE curriculum. In: *Frontiers in Education, 2004. FIE 2004. 34th Annual* (2004)

[Rashid et al. 2006] RASHID, A. ; MEDER, D. ; WIESENBERGER, J. ; BEHM, A.: Visual Requirement Specification In End-User Participation. In: *Multimedia Requirements Engineering, 2006. MERE '06. First International Workshop on* (2006), S. 6

[Reeps 2004] REEPS, I.: *Joy-of-Use eine neue Qualität für interaktive Produkte*, University of Konstanz, Master Thesis, Sep 2004

[Reeve 2005] REEVE, G.: *A Refinement Theory for μCharts*, The University of Waikato, Dissertation, 2005

[Robertson und Robertson 1999] ROBERTSON, S. ; ROBERTSON, J.: *Mastering the Requirements Process*. Addison-Wesley Professional, August 1999

[Rosson und Carroll 2002] ROSSON, M. ; CARROLL, J.: *Usability Engineering: Scenario-Based Development of Human Computer Interaction (Interactive Technologies)*. Morgan Kaufmann, October 2002

[Royce 1987] ROYCE, W.: Managing the development of large software systems: concepts and techniques. In: *ICSE '87: Proceedings of the 9th international conference on Software Engineering*. Los Alamitos, CA, USA : IEEE Computer Society Press, 1987, S. 328–338

[Rudd et al. 1996] RUDD, J. ; STERN, K. ; ISENSEE, S.: Low vs. high-fidelity prototyping debate. In: *interactions* 3 (1996), Nr. 1, S. 76–85

[Rumbaugh et al. 1998] RUMBAUGH, J. ; JACOBSON, I. ; BOOCH, G.: *The unified modeling language reference manual*. Boston [u.a.] : Addison-Wesley, 1998

[Rupp 2007] RUPP, C.: *Requirements Engineering und Management. 4., aktualis. u. erw. A.* München, Germany. : Hanser Fachbuchverlag, 2007

[Salvucci et al. 2005] SALVUCCI, D. ; ZUBER, M. ; BEREGOVAIA, E. ; MARKLEY, D.: Distract-R: rapid prototyping and evaluation of in-vehicle interfaces. In: *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM, 2005, S. 581–589

[Sauve et al. 2006] SAUVE, J. ; MOURA, A. ; M., Sampaio: An Introductory Overview and Survey of Business-Driven IT Management. In: *1st IEEE/IFIP International Workshop on Business-Driven IT Management (BDIM'06)*. Vancouver, Canada, 2006, S. 1–10

[Schlungbaum 1997] SCHLUNGBAUM, E.: Individual user interfaces and model-based user interface software tools. In: *IUI '97: Proceedings of the 2nd international conference on Intelligent user interfaces*. New York, NY, USA : ACM, 1997, S. 229–232

[Schön 1983] SCHÖN, D.: *The Reflective Practitioner: How Professionals Think in Action*. Basic Books, June 1983

[Schrage 1996] SCHRAGE, M.: Cultures of prototyping. (1996), S. 191 – 213

[Schuler und Namioka 1993] SCHULER, D. ; NAMIOKA, A. ; SCHULER, D. (Hrsg.) ; NAMIOKA, A. (Hrsg.): *Participatory Design: Principles and Practices*. Hillsdale, NJ, USA : L. Erlbaum Associates Inc., 1993

[Schwaber und Beedle 2001] SCHWABER, K. ; BEEDLE, M.: *Agile software development with Scrum*. Upper Saddle River, NJ, USA : Prentice Hall PTR, 2001

[Seshu 1963] SESHU, S.: Introduction to the theory of finite-state machines. In: *Proceedings of the IEEE* 51 (1963), Sept., Nr. 9, S. 1275–1275

[Shackel 1986] SHACKEL, B.: Ergonomics in design for usability. In: HARRISON, M. (Hrsg.) ; MONK, A. (Hrsg.): *People and computers: Proc. 2nd conf. BCS HCI specialist group*. Cambridge University Press., 1986, S. 45–64

[Sharp et al. 2007] SHARP, H. ; ROGERS, Y. ; PREECE, J.: *Interaction Design: Beyond Human Computer Interaction*. Wiley, March 2007

[Shneiderman 1998] SHNEIDERMAN, B.: *Designing the user interface (3rd ed.): strategies for effective human-computer interaction*. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 1998

[Sidoran et al. 1995] SIDORAN, J. ; BURNS, C. ; MAETHNER, S. ; SPENCER, D. ; BOND, H.: A case study on rapid systems prototyping and its impact on system evolution. In: *Rapid System Prototyping, 1995. Proceedings., Sixth IEEE International Workshop on* (1995), S. 125–130

[SIGCHI Bull. 1992] SIGCHI BULL.: A metamodel for the runtime architecture of an interactive system: the UIMS tool developers workshop. In: *SIGCHI Bull.* 24 (1992), Nr. 1, S. 32–37

[SIGGRAPH 1987] SIGGRAPH: Bibliography of software tools for user interface development. In: *SIGGRAPH Comput. Graph.* 21 (1987), Nr. 2, S. 145–147

[Simonsen und Kensing 1997] SIMONSEN, Jesper ; KENSING, Finn: Using ethnography in contextural design. In: *Commun. ACM* 40 (1997), Nr. 7, S. 82–88

[Smith 1985] SMITH, Brian C.: The limits of correctness. In: *SIGCAS Comput. Soc.* 14,15 (1985), Nr. 1,2,3,4, S. 18–26

[Smith 2006] SMITH, R.: *Business process management and the balanced scorecard: using processes as strategic drivers*. New York, NY, USA : John Wiley & Sons, Inc., 2006

[Sommerville 2005] SOMMERVILLE, I.: Integrated requirements engineering: a tutorial. In: *Software, IEEE* 22 (2005), Nr. 1, S. 16–23

[Sousa und Furtado 2003] SOUSA, K. ; FURTADO, E.: An Approach to Integrate HCI and SE in Requirements Engineering. In: HARNING, M. (Hrsg.) ; VANDERDONCKT, J. (Hrsg.): *Proc. of Interact'2003 Workshop on Closing the Gaps: Software Engineering and Human-Computer Interaction*. IOS Press, 2003, S. 81–88

[Sousa und Furtado 2005] SOUSA, K. ; FURTADO, E.: From usability tasks to usable user interfaces. In: *TAMODIA '05: Proceedings of the 4th international workshop on Task models and diagrams*. New York, NY, USA : ACM, 2005, S. 103–110

[Sousa et al. 2008a] SOUSA, K. ; MENDONCA, H. ; VANDERDONCKT, J.: Addressing the impact of business process changes on software user interfaces. In: *Business-driven IT Management, 2008. BDIM 2008. 3rd IEEE/IFIP International Workshop on* (2008), April, S. 11–20

[Sousa et al. 2008b] SOUSA, K. ; MENDONÇA, H. ; VANDERDONCKT, J. ; ROGIER, E. ; VANDERMEULEN, J.: User interface derivation from business processes: a model-driven approach for organizational engineering. In: *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*. New York, NY, USA : ACM, 2008, S. 553–560

[Österle 1995] ÖSTERLE, H.: *Business Engineering-Prozess-und Systementwicklung, Bd 1: Entwurfstechniken 2. Aufl.* Berlin : Springer, 1995

[Österle und Blessing 2003] ÖSTERLE, H. ; BLESSING, D.: Business Engineering Modell. In: ÖSTERLE, H. (Hrsg.) ; WINTER, R (Hrsg.): *Business Engineering-Auf dem Weg zum Unternehmen des Informationszeitalters, 2. Aufl.* Berlin : Springer, 2003, S. 65–86

[Strahonja und Picek 2005] STRAHONJA, V. ; PICEK, R.: User interface modeling within application system development. In: *Information Technology Interfaces, 2005. 27th International Conference on* (2005), S. 269–275

[Sukaviriya et al. 2007] SUKAVIRIYA, Noi ; SINHA, Vibha ; RAMACHANDRA, Thejaswini ; MANI, Senthil ; STOLZE, Markus: User-Centered Design and Business Process Modeling: Cross Road in Rapid Prototyping Tools. In: *INTERACT (1)*, 2007, S. 165–178

[Sutcliffe 2003] SUTCLIFFE, A.: Scenario-based requirements engineering. In: *Requirements Engineering Conference, 2003. Proceedings. 11th IEEE International* (2003), Sept., S. 320–329

[Sutherland 1964] SUTHERLAND, Ivan E.: Sketch pad a man-machine graphical communication system. In: *DAC '64: Proceedings of the SHARE design automation workshop*. New York, NY, USA : ACM Press, 1964

[Tick 2005] TICK, J.: Software user interface modelling with UML support. In: *Computational Cybernetics, 2005. ICCC 2005. IEEE 3rd International Conference on* (2005), S. 325–328

[Trætteberg 2002a] TRÆTTEBERG, H.: *Model-Based User Interface Design*, Norwegian University of Science and Technology, Dissertation, 2002

[Trætteberg 2002b] TRÆTTEBERG, Hallvard: Using User Interface Models in Design. In: *CADUI*, 2002, S. 131–142

[van der Veer und van Welie 2000] VAN DER VEER, G. ; VAN WELIE, M.: Task based groupware design: Putting theory into practice. In: *DIS '00: Proceedings of the 3rd conference on Designing interactive systems*. New York, NY, USA : ACM, 2000, S. 326–337

[Virzi et al. 1996] VIRZI, R. ; SOKOLOV, J. ; KARIS, D.: Usability problem identification using both low- and high-fidelity prototypes. In: *CHI '96: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM, 1996, S. 236–243

[Vredenburg et al. 2002] VREDENBURG, K. ; MAO, J. ; SMITH, P. ; CAREY, T.: A survey of user-centered design practice. In: *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM, 2002, S. 471–478

[Walker et al. 2002] WALKER, M. ; TAKAYAMA, L. ; LANDAY, J.: High-fidelity or low-fidelity, paper or computer? Choosing attributes when testing web prototypes. In: *Procedings of the Human Factors and Ergonomics Society 46th Annual Meeting*. Baltimore, USA., 2002, S. 661 – 665

[Weinberg und Stephen 2002] WEINBERG, J. ; STEPHEN, M.: Participatory design in a human-computer interaction course: teaching ethnography methods to computer scientists. In: *SIGCSE Bull.* 34 (2002), Nr. 1, S. 237–241

[van Welie 2001] WELIE, M. van: *Task-based User Interface Design*, de Vrije Universiteit Amsterdam, Dissertation, 2001

[Whiteside et al. 1988] WHITESIDE, J. ; BENNETT, J. ; HOLTZBLATT, K.: Usability Engineering: Our Experience and Evolution. In: HELANDER, M. (Hrsg.): *Handbook of Human-Computer Interaction*. Amsterdam : Elsevier Science Publishers, 1988, S. 791–817

[Williams 2003] WILLIAMS, A.: Examining the use case as genre in software development and documentation. In: *SIGDOC '03: Proceedings of the 21st annual international conference on Documentation*. New York, NY, USA : ACM, 2003, S. 12–19

[Wilson et al. 1993] WILSON, S. ; JOHNSON, P. ; KELLY, C. ; MARKOPOULOS, P.: Beyond hacking: A model based approach to user interface design. In: *In Proceedings of HCI'93*, University Press, 1993, S. 217–231

[Winograd und Flores 1987] WINOGRAD, T. ; FLORES, F: *Understanding Computers and Cognition: A New Foundation for Design*. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 1987

[Xia 2005] XIA, Y.: *A Language Definition Method for Visual Specification Languages*, UNIVERSITäT ZüRICH, Dissertation, 2005

[Zave und Jackson 1997] ZAVE, P. ; JACKSON, M.: Four dark corners of requirements engineering. In: *ACM Trans. Softw. Eng. Methodol.* 6 (1997), Nr. 1, S. 1–30

[Zenka und Slavik 2004] ZENKA, R. ; SLAVIK, P.: Supporting UI design by sketch and speech recognition. In: *TAMODIA '04: Proceedings of the 3rd annual conference on Task models and diagrams*. New York, NY, USA : ACM, 2004, S. 83–90

[Zetie et al. 2005] ZETIE, Carl ; VISITACION, Margo ; DOWLING, Kimberly Q.: Show, Don't Tell - How High-Fidelity Prototyping Tools Improve Requirements Gathering / Forrester Research, Inc. August 2005. – Forschungsbericht

[Zhang et al. 2004] ZHANG, Kang ; SONG, Guang L. ; KONG, Jun: Rapid Software Prototyping Using Visual Language Techniques. In: *RSP '04: Proceedings of the 15th IEEE International Workshop on Rapid System Prototyping*. Washington, DC, USA : IEEE Computer Society, 2004, S. 119–126

[Zhu und Gorton 2007] ZHU, L. ; GORTON, I.: UML Profiles for Design Decisions and Non-Functional Requirements. In: *SHARK-ADI '07: Proceedings of the Second Workshop on SHAring and Reusing architectural Knowledge Architecture, Rationale, and Design Intent*. Washington, DC, USA : IEEE Computer Society, 2007, S. 8

[Zuser et al. 2005] ZUSER, W. ; HEIL, S. ; GRECHENIG, T.: Software quality development and assurance in RUP, MSF and XP: A comparative study. In: *3-WoSQ: Proceedings of the third workshop on Software quality*. New York, NY, USA : ACM, 2005, S. 1–6