MASTERARBEIT

Möglichkeiten des User Interface (Re)Designs in späten Projektphasen

Ausgeführt am
Institut für Gestaltungs- und Wirkungsforschung
der Technischen Universität Wien

unter der Anleitung von
ao. Univ. Prof. Dr. Peter Purgathofer

durch
Eva Ganglbauer
Gerlgasse 1a/2 1030 Wien

Wien, März 2008

Möglichkeiten des User Interface (Re)Design in späten Projektphasen

30. März 2008

Danke an meine Betreuer Peter Purgathofer und Wilfried Reinthaler, an Bettina Gröschl und Elisabeth Baldauf.

Kurzfassung

Ein Bewusstsein für Design erfreut sich in Softwareunternehmen zunehmender Beliebtheit, dennoch sind nach wie vor vorwiegend die EntwicklerInnen gleichzeitig die DesignerInnen eines Softwareprodukts. Erst wenn sich ein Produkt weniger erfolgreich entwickelt als erwartet oder Probleme auftauchen, wird Design als Aspekt einer Produktentwicklung in Betracht gezogen. In dieser Masterarbeit soll diese Situation, deren Ursachen und die Auswirkungen auf die Qualität eines Softwareprodukts beleuchtet werden. Als Gegensatz dazu sollen ein idealer Designprozess, insofern ein solcher existiert, und dessen notwendige Schritte theoretisch abgehandelt werden. Die Masterarbeit wurde von einem Projekt, in dem das Interface eines Document Designers entwickelt wurde, begleitet. Das Projekt wird unter den beschriebenen Aspekten beleuchtet und einer Reflexion unterzogen. Abschließend werden mit Hilfe der Auseinandersetzung mit dem Projekt und den theoretischen Gesichtspunkten die Möglichkeiten des Designs in späten Projektphasen diskutiert. Ziel ist es, ein Bewusstsein für Design und die Notwendigkeit einer Preproduction-Phase in Softwareprojekten zu schaffen.

Abstract

Although there is a growing awareness for design in software companies, programmers still assume the role of the designer too. Only when a product is less successful than expected, or when problems arise, design is taken into consideration as an integral part of the software development process. In this master's thesis this common approach in product development, its causes and its effects on a software product shall be investigated. Secondly, in a theoretical discussion, this approach will be contrasted with a suggestion for an ideal design process including its necessary steps. The master thesis was accompanied by a project, where the Interface of a Document Designer was developed. This project will be considered and reflected against the background of the described theoretical aspects. Finally, the project and the theoretical basics, as well as the possibilities of Design in late project phases will be debated. The objective is to raise awareness of the importance of a preproduction phase before implementing in software projects.

Inhaltsverzeichnis

1 Einleitung

Täglich wird Software verwendet, die aufgabenspezifische Aktivitäten mit einem bestimmten Ziel leitet, organisiert und erleichtert. Manche Softwareprodukte verwenden wir spielerisch einfach und gerne, andere sind schwierig zu gebrauchen, und im schlechtesten Fall verursachen sie Missverständnisse und Frustration. Sie fühlen sich nicht gut an, der Ablauf der Aktionen wird nicht richtig verstanden, sie sind also nicht so wie sie im besten Falle sein sollten: Intuitiv und einfach zu benutzen. Nach wie vor stehen im Softwareentwicklungsprozess meist nicht die EndanwenderInnen und deren Bedürfnisse im Vordergrund, sondern technische Aspekte. Es existieren viele Ursachen für weniger gut gebrauchstaugliche Software, die den Bedürfnissen der EndanwenderInnen minimal entspricht. Eine Faktor ist, dass in vielen Softwareprojekten ein Bewusstsein für Design nach wie vor wenig Verbreitung findet. Dies ist auch auf die geschichtlichen Entwicklung und das kurze Bestehen des Software Engineerings als Disziplin zurückzuführen.

In dieser Masterarbeit wird in den theoretischen Kapiteln beschrieben, was gutes Design charakterisiert, welche Aspekte in einem Designprozess beachtet werden sollen, welche Methoden existieren und welche Faktoren speziell beim User Interface Design zu berücksichtigen sind, um gutes Design zu gewährleisten. Gutes Design bedeutet zufriedene EndanwenderInnen und ein Produkt, dass sowohl im unkomplizierten Gebrauch als auch in der Qualität des Erlebens überzeugt.

Fragestellung und Motivation

Ein Bewusstsein für Design erfreut sich in Softwareunternehmen zunehmender Beliebtheit, dennoch sind nach wie vor vorwiegend die EntwicklerInnen gleichzeitig die DesignerInnen eines Produkts. Erst wenn sich ein Produkt weniger erfolgreich entwickelt als erwartet oder Probleme auftauchen, wird Design als Aspekt einer Produktentwicklung in Betracht gezogen. In dieser Arbeit soll diese Situation, deren Ursachen und die Auswirkungen auf ein Softwareprodukt beleuchtet werden. Im Projekt, das die Masterarbeit begleitet hat, wurde das Interface eines Document Designers überarbeitet und teilweise auch neu entwickelt. Das Projekt war dadurch gekennzeichnet, dass Design erst sehr spät in den Entwicklungsprozess involviert wurde. Motivation der Arbeit ist es, die Möglichkeiten und Ursachen von Design in späten Projektphasen zu erarbeiten.

Der Fokus der theoretischen Kapitel war mit dem Gedanken verbunden, die wissenschaftlichen Hintergründe, die zur Analyse, Beschreibung und dem Verständnis des Projekts wichtig erscheinen, zu erläutern. Es war nicht meine Absicht einen Aspekt vollständig auszuführen, sondern, im Kontext des Projekts betrachtet, die wichtigsten Punkte aus einer theoretischen Sichtweise zu erarbeiten. Das Projekt wird in einem eigenen Kapitel nachbetrachtend beleuchtet und einer Reflexion unterzogen. Ziel dieser Arbeit ist es, anhand der Auseinandersetzung mit dem Projekt und den theoretischen Grundlagen zu erörtern, mit welchen Aspekten Design in späten Projektphasen verbunden ist.

Aufbau der Arbeit

In Kapitel 2 wird die geschichtliche Entwicklung der Rolle von Design bei der Softwareentwicklung betrachtet. Es wird der Frage nachgegangen, was gutes Design ausmacht und welche Kriterien und Vorgangsweisen dabei bedeutend sind. Außerdem wird die Rolle von Emotionen im Design von Produkten diskutiert.

Im dritten Kapitel wird neben der wichtigen Rolle der AnwenderInnen und des Produktkontexts die Praxis des Designprozesses an sich erläutert: Methoden des User Research, das Ausprobieren von Ideen durch Sketching, Prototyping, User Testing, und weitere Designmethoden.

User Interface Design ist Gegenstand des vierten Kapitels. Eine Auseinandersetzung mit dem Thema macht eine Diskussion des Begriffs erforderlich. In diesem Abschnitt werden Kriterien diskutiert, die beim Design als Hilfestellung dienen können. Im Hinblick auf das Projekt werden Designkriterien von Editoren berücksichtigt.

Im fünften Kapitel wird das Projekt nachbetrachtend diskutiert. Es wird auf den Ablauf des Projekts, unsere Designvorschläge, Probleme, die besonderen Umstände später Einstiegsphasen für Design und daraus resultierende Möglichkeiten und Erfolge eingegangen.

Im sechsten Kapitel sollen die besonderen Umstände und Aspekte von Designprojekten beleuchtet werden, in denen Design erst spät eingesetzt wird bzw. welche Möglichkeiten und Einschränkungen durch (Re)Design entstehen.

In den abschließenden Bemerkungen werden sie theoretischen Aspekte noch einmal zusammengeführt und in Relation zum Projekt betrachtet.

2 Design

Dieses Kapitel gibt einen Überblick über das Designverständnis in der Literatur. Im Besonderen wird Design als eigener Bereich in Relation zur Softwareentwicklung beleuchtet. Dies ist nicht zu verwechseln mit dem Begriff des Software Designs[1], der sich mit der technischen Seite einer Softwarelösung auseinandersetzt. Die Geschichte der Softwareentwicklung erscheint im Kontext von Design besonders interessant und soll in diesem Kapitel kurz dargelegt werden. Im Zusammenhang dazu ist es wesentlich, die Betrachtungsweise von Design als Disziplin der Problemlösung und deren geschichtliche Entwicklung zu erläutern, denn diese ist mit der Geschichte der Softwareentwicklung unmittelbar verbunden. Eine Auseinandersetzung mit Design wirft außerdem die Frage auf, was gutes Design ausmacht, wobei in dieser Arbeit nur ansatzweise versucht wird, diese Frage zu beantworten. Weiters soll der Frage nachgegangen werden, welche Vorgehensweisen gutes Design begünstigen. Nachfolgend wird erörtert, welche Rolle Emotionen in der Wahrnehmung eines Produkts innehaben und auf welchen Ebenen dies geschieht.

Design als Profession bei Softwareprojekten

Die Disziplin des Software Engineerings ist im vergleichsweise eine sehr junge. Die ersten Programme wurden in den 1950er bzw. den frühen 1960er Jahren entwickelt, und waren im Vergleich zu heute wenig komplex. In den kommenden Jahren steigerte sich mit den technischen Möglichkeiten auch die Komplexität der Anforderungen an ProgrammiererInnen und Entwicklerteams. Softwareprojekte scheiterten im großen Stil, da die bisherige Arbeitsweise nicht

[1] Software Design ist ein Prozess, der sich mit Problemen und der technischen Planung einer Softwarelösung beschäftigt.

auf so große bzw. komplexe Projekte anwendbar war, sodass von einer Softwarekrise die Rede war, die bis heute andauert (Dogs & Klimmer, 2005, S. 19). Der Krise kann zumindest ein positiver Aspekt abgewonnen werden, denn nur aus einer Krise kann weitere Entwicklung stattfinden (Kruchten, 2005). Es wird unter anderem versucht, den komplexen Anforderungen an das Software Design mit agilen Methoden (vgl. Kap. 6, S. 76) entgegenzutreten (Dogs & Klimmer, 2005).

Software Design wurde in seiner Entwicklung anfänglich als reine Ingenieurdisziplin betrachtet, Einflüsse aus anderen Disziplinen (neben den Ingenieur- und Computerwissenschaften) wurden nicht in Betracht gezogen (vgl. Abbildung 2.1). Design als Kunst, als Fertigkeit, als eigene Disziplin ohne dem Ansatz der Ingenieurwissenschaften, war davor schon eine anerkannte Praxis, die sich jenseits von Funktionen um das Erleben und den Gebrauch von Produkten annahm (Taylor, 2001).

In vielen Disziplinen fand historisch eine Arbeitsteilung zwischen DesignerInnen und ErzeugerInnen statt, wie etwa in der Architektur (Lawson, 1997). Der Zustand, Softwaredesign von einer rein technischen Seite zu beleuchten, hält in vielen Fällen noch bis heute in der IT-Industrie an. Am Anfang der Geschichte der Softwareentwicklung agierten ProgrammiererInnen gleichzeitig als DesignerInnen eines Systems, mit der Zeit wurden die Entwicklungsprozesse komplexer und andere Faktoren wie Management, Design und Usability Engineering wurden involviert. In Abbildung 2.1 ist die zeitliche Entwicklung des Software Engineering Prozesses illustriert. Eine Aufgabenverteilung wird in vielen IT-Projekten nach wie vor nicht durchgeführt, und meist entwickeln ProgrammiererInnen das Design der Applikation, mit dem Ergebnis, dass die Software oft dem Modell der ProgrammiererInnen entspricht, anstatt dem der AnwenderInnen (vgl. Abbildung 2.2).

> *It is just as inappropiate to have an engineer manage the design process as it is to have a designer who graduated from art college be responsible for the product's engineering details.* (Buxton, 2007, S. 75)

Es soll hier keine Seite der anderen vorgezogen oder mehr Bedeutung zugesprochen werden, beide Fähigkeiten sind fundamental für den Designprozess, allerdings sollten sie auch an richtiger Stelle eingesetzt werden. In

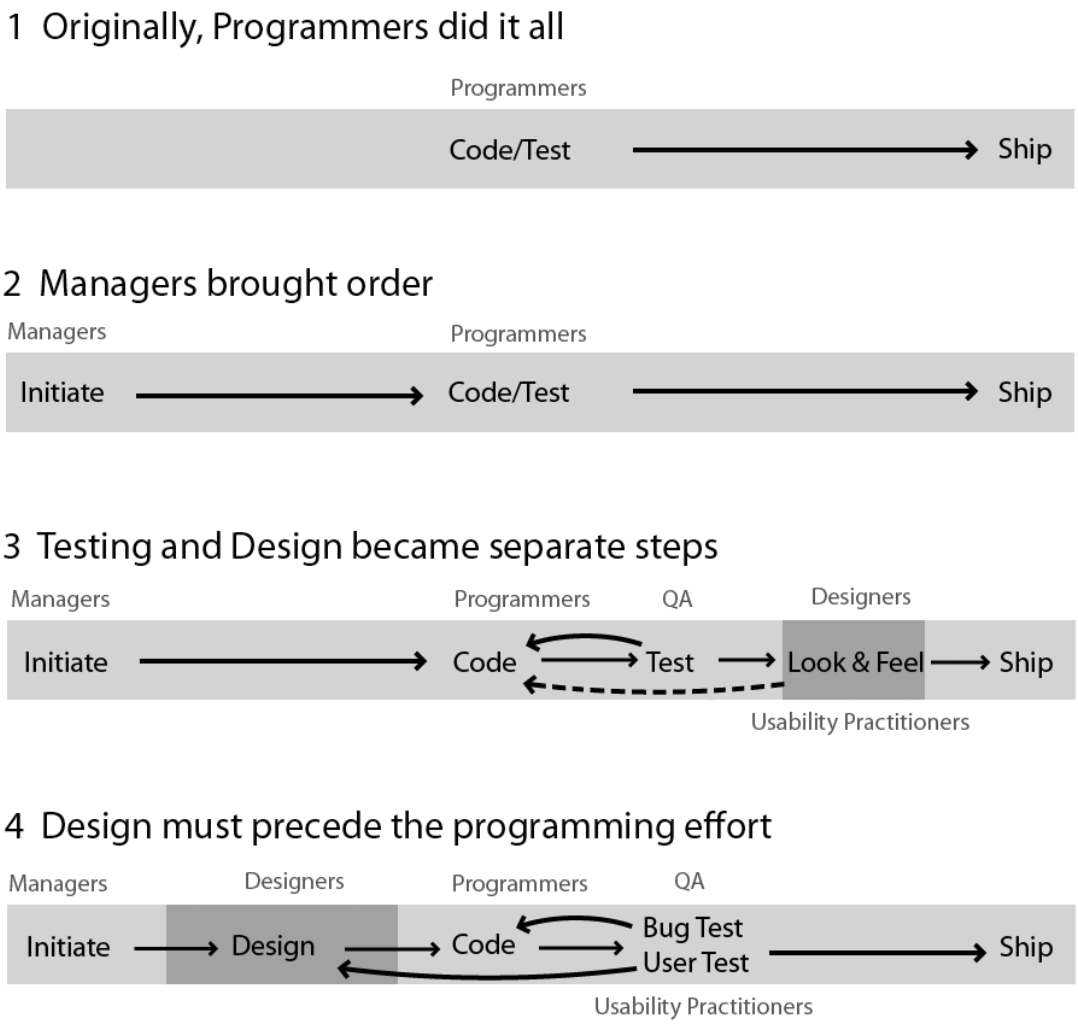

Abbildung 2.1: Zeitliche Entwicklung des Software Engineering Prozesses nach Cooper. Am Beginn wurde alles von ProgrammiererInnen entwickelt. Mit der Zeit wurden die Entwicklungsprozesse komplexer, und andere Faktoren wie Management, Design und Usability wurden involviert. (Quelle: Cooper, 2003, S. 6)

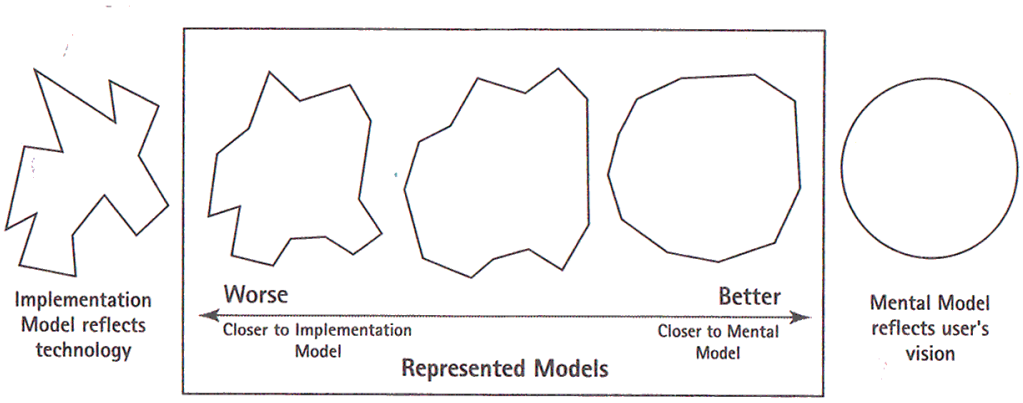

Abbildung 2.2: Software, die ausschließlich von ProgrammierInnen entwickelt wurde, reflektiert meist die technische Implementierung. DesignerInnen sollen ein Produkt näher zu dem Modell bringen, das UserInnen von einem Produkt haben. (Quelle: Cooper, 2003, S. 23)

vielen Softwareunternehmen wird nach wie vor die Ansichtsweise vertreten, dass ProgrammiererInnen auch die DesignerInnen der Software sein sollen, da sie das Wissen über die sachbezogenen technischen Belange besitzen. Obwohl es auf den ersten Blick natürlich erscheint, EntwicklerInnen das Design zu überlassen, ist der Konflikt zwischen den Zielen der AnwenderInnen und denen der EntwicklerInnen vorprogrammiert (Cooper, 2004, S. 108).

Ein idealer Designprozess, der aus dem Blickwinkel Buxtons beleuchtet wird, ist in Abbildung 2.3 dargestellt. Der Implementierung geht eine Preproduction-Phase voran, erst dann wird grünes Licht zur Implementierung gegeben. Im Gegensatz zu Coopers Ansichtsweise begleitet Designarbeit den ganzen Entwicklungsprozess eines Produkts (Buxton, 2007, S. 73f).

Design als Problemlösung

Design kann als eine Disziplin gesehen werden, deren Aufgabe darin besteht, Probleme zu erfassen und zu lösen (Lawson, 2004, S. 19).

> *Design is to be viewed as the process of problem understanding and problem solving with the aim of producing an artefact.* ((Kushalani, Smith & Howard, 1994), zitiert nach (Gasson, 2007))

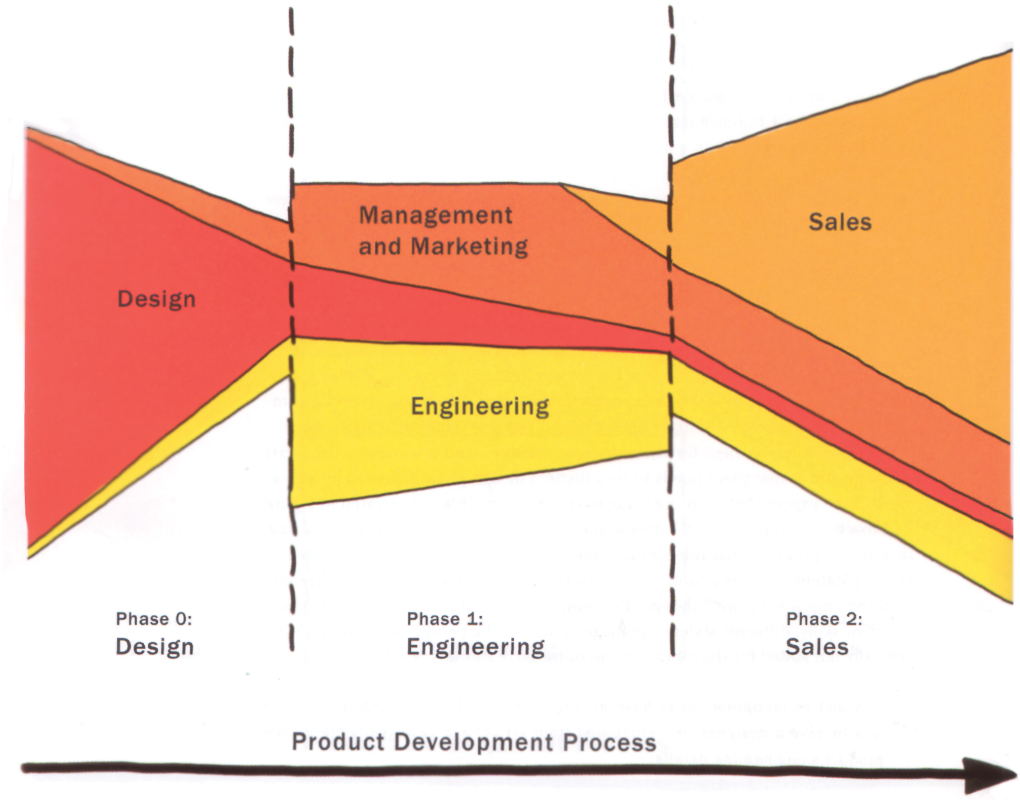

Abbildung 2.3: Der Entwicklungsprozess eines Produkts nach Buxton: Design begleitet die ganze Entwicklung und ist ein fixer Bestandteil im Prozess. Vor allem am Anfang eines Projekts spielt Design eine tragende Rolle. (Quelle: Buxton, 2007, S. 76)

Designmethoden

Eine geschichtliche Betrachtungsweise des Designprozesses steht unweigerlich im Zusammenhang mit der Auffassung von Design als Problemstellung und -lösung. Die Anschauungsweise von Designmethoden war in der Literatur einem stetigen Wandel unterworfen, einhergehend damit hat sich das Designverständnis fortwährend verändert. Löwgren beschreibt die Entwicklung des Designverständnisses mithilfe von drei Generationen (Löwgren, 1995):

In der ersten Entwicklungsstufe wird Design als funktionelle Analyse aufgefasst: Ein Designproblem wird wie ein typisches Ingenieurproblem behandelt, es wird als einheitlich und als wohldefiniert betrachtet, aufgeteilt in die drei iterierenden Phasen Analyse, Synthese und Evaluierung. Designziele und -kriterien sind

vordefiniert und die Mittel zum Erreichen des festgelegten Ziels werden im Designprozess rationell erfasst ((Lanzara, 1983), zitiert nach (Gasson, 2007)).

Da Designprobleme in der Regel nicht wohldefiniert sind, ist keine konkrete Vorgangsweise gegeben, um zu einer universell richtigen Lösung für ein bestehendes Designproblem zu gelangen. Ein wohldefiniertes Problem wäre etwa das Acht-Damen-Problem[2]. Horst Rittel und Webber sprechen im Zusammenhang von nicht wohldefinierten Problemen von "wicked problems", die folgendermaßen beschrieben werden (Conklin, 2006), (Rittel & Webber, 1973):

1. *You don't understand the problem until you have developed a solution.*
2. *Wicked Problems have no stopping rule [3].*
3. *Solutions to wicked problems are not right or wrong.*
4. *Every wicked problem is essentially unique and novel.*
5. *Every solution to a wicked problem is a "one shot operation".*
6. *Wicked problems have no given alternative solutions.*

Aus der Definition der Wicked Problems wurde in der nächsten Entwicklungsstufe Design als Problemlösungsmodell gesehen. Der Designkontext enthält Hinweise, die dem/der DesignerIn erlauben, der Situation inhärente Strukturen zu erkennen und einen Zusammenhang zwischen den Strukturen und dem Kontext herzustellen. Ein Designproblem wird auf mehrere kleinere Probleme reduziert, die wiederum neue Probleme aufwerfen. Die Probleme werden so klein, bis sie schließlich gelöst werden können. Dieser Prozess reduziert demzufolge die Komplexität des Problems (Rittel & Webber, 1973). Analyse und Synthese bedingen sich unter dieser Betrachtungsweise gegenseitig, was im Widerspruch zur ersten Entwicklungsstufe steht. Nachdem aber Designprobleme nicht vollständig erfasst werden können, können sie auch nicht vollständig gegeben sein. Schön definierte den Begriff des "problem settings" (Schön, 1990) im Gegensatz zum "problem solving". Da ein Designproblem aufgrund seiner nicht wohldefinierten Natur nur unvollständig erfasst werden kann, müssen die

[2] Eine Stellung für acht Damen auf einem Schachbrett, sodass sich keine zwei Damen gegenseitig nach den Regeln des Schach schlagen können. Jede Figur darf jede andere angreifen.
[3] Eine Bedingung, unter welcher ein Vorgang beendet werden kann

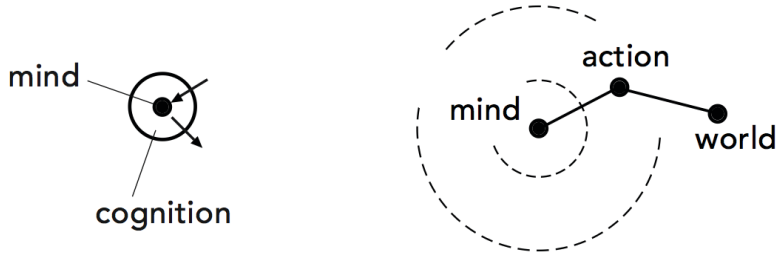

Abbildung 2.4: Die linke Abbildung entspricht der traditionellen Ansichtsweise der Wahrnehmung. Unsere Kognition ist aber durch das Handeln mit der Welt bedingt. (Quelle: Gedenryd, 1998, S. 12)

Anforderungen selbst abgesteckt werden (Gedenryd, 1998, S. 69). Gedenryd führt weiter aus:

> *The need for problem solving leads to the impossibility of separating problem definition from problem solving, which in turn provides the basis for the argument that in genuine cases, design cannot be separated into stages.*

Die dritte Generation legt den Schwerpunkt auf das Handeln und Tun der DesignerInnen, weniger auf eine rationale Ansichtsweise von Design als Problemstellung und -lösung, das einem bestimmten Prozess folgt. Gedenryd beschreibt den Begriff der "theory of inquiry", welche besagt, dass durch das Anwenden und implizite Testen von Wissen eine neue Sichtweise über das Problem und die Lösung entsteht (Gedenryd, 1998, S. 79f). Das Handeln ist wesentlich für einen Erkenntnisprozess und steht in enger Beziehung zum Wissen und der Welt (vgl. Abb. 2.4). Gedenryd führt weiter aus, dass die interaktive Wahrnehmung einen Prozess darstellt, in dem Kognition und Aktion, oder Wissen und Tun, eng miteinander verbunden sind. Demzufolge besteht eine eng gekoppelte Interaktion zwischen der Welt und dem Verstand (Gedenryd, 1998, S. 112).

Abschließend betrachtet kann folgender Unterschied zwischen den beiden Ansätzen des technischen und des kreativen Designs festgestellt werden: In den Ingenieurdisziplinen wird davon ausgegangen, dass das zu lösende Problem vollständig gegeben ist und beschrieben werden kann, es muss nur noch eine Lösung zum spezifizierten Problem gefunden werden. Technisches Design ist

offen für strukturierte Beschreibungen und wird als eine Kette von Transformationen betrachtet, die die abstrakten Anforderungen in ein konkretes Artefakt umwandelt. Im Gegensatz dazu wird im kreativen Design respektiert, dass Problem und Lösung eng aneinander gekoppelt sind und sich gegenseitig bedingen. In diesem Wechselspiel werden die Möglichkeiten des Designs durch die parallele Erzeugung vieler Ideen und Konzepte ausgelotet. Es liegt in der Natur des kreativen Designs, die Lösung eines Designproblems als unbestimmt zu betrachten (Löwgren, 1995).

Gutes Design

Welche Eigenschaften hat gutes Design im Gegensatz zu schlechtem? Diese Frage ist schwer wenn überhaupt zu beantworten. An dieser Stelle sollen einige Lösungsansätze diskutiert werden, beginnend mit einem Zitat von Jonas Loewgren (Löwgren & Stolterman, 2007, S. 5):

> *Our basic assumption ist that good design is determined by many factors. To begin with, the digital artifact has to be evaluated in relation to a situation. Even though certain aspects of a digital artifact might be independent of the context, its most crucial qualities are always deeply context-dependent.*

Nach Jonas Löwgren ist eine Vorraussetzung für gutes Design, den Kontext, in dem das Produkt eingebettet ist, zu betrachten und zu erforschen. Die Qualität eines Produkts muss immer in der Beziehung der Intentionen und Erwartungen zu einer spezifischen Situation gemessen werden. Folgend bestimmen die Fertigkeiten und Kompetenzen der AnwenderInnen die Güte eines Produkts. Mit der Zielgruppe von beginnenden AnwenderInnen ist ein komplexes Produkt wahrscheinlich unzufriedenstellend, dagegen ein simples sehr brauchbar (Löwgren & Stolterman, 2007, S. 5).

Welche Kriterien sollen nun erfüllt werden, um eine befriedigende „Qualität des Erlebens" sicherzustellen? Qualität im Erleben meint in diesem Zusammenhang beispielsweise wie das Produkt in den Händen gefühlt wird, wie gut die Funktionsweise verstanden wird, wie die Anwendung erlebt wird, und ob es im Kontext in dem es angewendet wird, zweckerfüllend ist. Um die Qualität des

Erlebens zu beschreiben, wurden folgende Kriterien von der Association for Computing Machinery (ACM) für Interaktionsdesign formuliert (Alben, 1996):

- **Verstehen der AnwenderInnen:** Die Bedürfnisse der AnwenderInnen werden betreffend den Aufgaben und Umgebungen der Menschen, für die das Produkt entworfen wurde, zufrieden gestellt.
- **Effektiver Designprozess:** Das Produkt ist ein Ergebnis eines gut durchdachten und ausgeführten Designprozesses. Es wurden Methoden verwendet, die das Produkt zufriedenstellend entworfen und verwirklicht haben.
- **Bedürfnis:** Das Bedürfnis, das vom Produkt gestillt wird, leistet einen sozialen, ökonomischen und ökologischen Beitrag.
- **Erlernbar- und Verwendbarkeit:** Das Produkt ist in der Bedienung einfach zu erlernen und zu verwenden. Ansonsten unterstützt das Produkt viele mögliche Wege, Lösungen für ein bestimmtes Problem zu finden. Die Fähigkeiten und Strategien für Problemlösungen sollen leicht erlernbar sein.
- **Adäquatheit:** Das Design des Produkts löst das aufgeworfene Problem effizient auf einer zweckdienlichen Ebene, und dient den AnwenderInnen auf praktische Art und Weise.
- **Ästhetisches Erleben:** Das Produkt ist ästhetisch und sinnlich ansprechend, es besteht eine Konsistenz zwischen der Seele des Produkts und seinem Stil.
- **Variabilität:** Das Produkt kann so adaptiert werden, dass es sich für bestimmte Bedürfnisse und Vorlieben von individuellen AnwenderInnen und Gruppen eignet. Das Design erlaubt dem Produkt, sich zu verändern und sich für neue, vielleicht unvorhergesehene Anwendungen zu entfalten.
- **Handhabbarkeit:** Das Design des Produkts unterstützt mehr als die Funktionalität, es befürwortet den ganzheitlichen "context of use".

Es stellt sich in diesem Zusammenhang die Frage, inwieweit solche Kriterien eine tragende Rolle zur Bewertung eines Produktes spielen. Sicher können Kriterien als Hilfestellung und als Anhaltspunkte verwendet werden, sie sollten aber nicht als Maß aller Dinge gelten und können eine objektive Bewertung einschränken.

Gute Designarbeit

Gute Designarbeit kann nur dann entstehen wenn diese von der Umgebung gefördert wird. Das Design von Produkten wird von Jones et al. (Jones & Greene, 2000) metaphorisch als dreibeiniger Stuhl beschrieben. Die drei Beine bzw. Säulen beinhalten das Verstehen der Technologie, welche im System eingesetzt wird, das Verstehen der AnwenderInnen und des Verwendungskontexts („use context"), sowie kreatives Verständnis. Die Metapher beschreibt, dass der Stuhl durch das Fehlen einer der drei Beine zusammenbricht.

Nach Ansicht der AutorInnen benötigt gutes Design vor allem Können, Fachkenntnis und kreatives Verständnis. Die Arbeitsumgebung und die Atmosphäre sollen diese Eigenschaften bestmöglich unterstützen. Im Folgenden werden die Kriterien beschrieben, die eine Umgebung schaffen, in der Kreativität gefördert wird. Die AutorInnen schöpfen dabei aus eigenen Erfahrungen in interdisziplinären Designteams (Jones & Greene, 2000).

Interdisziplinäre Teams von Anfang bis Ende

Kreativität soll nicht auf DesignerInnen spezieller Gebiete beschränkt sein. Jedes der Teammitglieder soll sich in kreativem Denken üben. Außerdem sollen die MitarbeiterInnen verschiedener Disziplinen eng miteinander zusammenarbeiten und kommunizieren, da nach Meinung der AutorInnen gerade die Interdisziplinarität das kreative Verständnis fördert. Es kann Wissen, kommend aus den unterschiedlichsten Hintergründen, ausgetauscht werden, was zu neuen Ideen und Kenntnissen in anderen Fachgebieten führt.

Gemeinsame Werte und gemeinsame Sprache

Das Finden einer gemeinsamen Sprache stellt für Menschen aus unterschiedlichen Disziplinen eine weitere Herausforderung dar. Sie sollen lernen sich gegenseitig zuzuhören, ihre Ideen zu kommunizieren und dabei eine Sprache zu verwenden, die auch die anderen Projektmitglieder verstehen.

Alle Teammitglieder haben Zugang zu Rohmaterial

Alle Teammitglieder sollen über das Gebiet, in welchem das Projekt durchgeführt wird, Bescheid wissen. Dies ermöglicht, dass Vorwissen und spezielle Fähigkeiten auf die Rohdaten angewendet werden können. Vorbearbeitete Daten sind bereits in eine bestimmte Richtung interpretiert und mit einer Perspektive belegt.

Einbeziehung der AnwenderInnen und Interessensgruppen während des ganzen Designprozesses

Nach Meinung der AutorInnen ist es nicht nur notwendig dass die EndanwenderInnen von Anfang an in das Projekt miteinbezogen werden, sondern auch Vertreter anderer Interessensgruppen, wie etwa AuftraggeberInnen. Die Einbeziehung dieser Personen eröffnet mehr Quellen für Kreativität und neue Ideen.

AnwenderInnen zeigen vielleicht Probleme auf, aber nicht Lösungen

In einem Designprozess soll es möglich sein, in jede Richtung zu denken und jede Perspektive einzunehmen. AnwenderInnen und andere Interessensgruppen beschreiben meist Probleme während sie schon die Lösung mitaufzeigen. Eine innovative Lösung des Problems soll aber vom Team selbst durch Verstehen der Anliegen und Expertise gestellt werden.

Entwicklung in der Umgebung der EndanwenderInnen

Es ist wichtig, zu jedem Schritt im Designprozess Feedback der EndanwenderInnen zu bekommen, weil es schließlich auch sie sind, die das Produkt schlussendlich verwenden werden. Durch wiederholtes Feedback ist es möglich, gute Ideen, die angenommen werden, weiter zu verfolgen, und schlechte Ideen zu verwerfen. Je früher eine schlechte Idee verworfen wird, desto günstiger, da sie nicht erst am Ende des Entwicklungsprozesses korrigiert werden muss, was teurer wäre. Feedback erhält man in der Regel durch Prototyping und User Testing.

Testen in der Welt der AnwenderInnen

Testen außerhalb der Realität der AnwenderInnen wirft nicht alle möglichen Probleme auf, die auftreten können. Am besten ist es, AnwenderInnen während des Testens zu beobachten, laut Meinung der AutorInnen soll dies nicht nur durch HCI-ExpertInnen geschehen. Es ist für EntwicklerInnen belehrend selbst zu sehen, wie EndanwenderInnen mit dem Produkt oder dem System umgehen und welche Probleme dabei auftauchen. Auch andere Interessensgruppen sollen im besten Fall bei den Tests anwesend sein.

Einige dieser Punkte, wie das Testen in der wahren Umgebung der AnwenderInnen werden in dieser Arbeit noch behandelt werden. Der interessante Ansatz, dass alle Mitglieder des Teams dasselbe Rohmaterial als Ausgangspunkt verwenden sollen und keine Berichte oder Zusammenfassungen, da diese schon vorgefasste Meinungen und Beurteilungen in sich tragen, ist in der Praxis aus ressourcenintensiven Gründen wahrscheinlich schwer durchzusetzen. Wie bei allen Vorschlägen zu Gestaltungsrichtlinien sind sie mit Vorsicht zu genießen, da sie nicht für jedes Projekt anwendbar sind und als Hilfestellung, aber nicht als Credo dienen sollen.

Emotionales Design

> *More Emotions are better than less.* (Maeda, 2006, S. 63)

Pelle Ehn schreibt über Design, dass es in der Verwendung von postmodernen Medien und Technologien nicht darum geht, technisch möglichst fortschrittlich zu agieren, sondern reichhaltige Sinnesfreude zu bereiten. Es sollen neue Technologien nicht angebetet werden, sondern kritische und ästhetische technische Produkte entwickelt werden, die Informations- und Kommunikationstechnologie mit Design, Kunst und Gesellschaft vereinen. Es sollen offene Dialoge geführt, und zukünftige AnwenderInnen im Designprozess eingebunden werden. Es sollen nützliche, interessante, zweckmäßige und möglicherweise schöne und unterhaltende alltägliche Dinge für einfache Menschen entwickelt werden (Ehn, 2002). Dies sind sehr wichtige Punkte und Erkenntnisse, aber auch der emotionale Aspekt beim Erleben eines Produktes darf nicht außer Acht gelassen werden. Ehn hat diesen Punkt mit reichhaltiger

Sinnesfreude schon angedeutet. Andere Aspekte außer Emotionen sind von physischer, sinnlicher, kognitiver und ästhetischer Natur.

Forlizzi et al. argumentiert, dass Emotion das Herz jedes menschlichen Erlebens, und essentiell für die Interaktion und das Erleben zwischen AnwenderInnen und System ist.

> *Emotion affects how we plan to interact with products, how we actually interact with products, and the perceptions and outcomes that surround those interactions. Emotion serves as a resource for understanding and communicating about what we experience.*
> (Forlizzi & Battarbee, 2004)

Eine interaktionszentrierte Ansichtsweise ist wertvoll wenn nicht obligatorisch wie einE AnwenderIn ein Produkt erlebt, da solche Sichtweisen die Kluft zwischen AnwenderInnen und DesignerInnen zu überbrücken versuchen. Aus diesem Grund ist das Verstehen des Erlebens ein notwendiger Aspekt beim Design von Interaktivität (Forlizzi & Battarbee, 2004).

Donald Norman definierte drei Ebenen, die Produkte für uns interessant machen und uns vor allem auf einer emotionalen Ebene ansprechen. Emotionen spielen eine entscheidende Rolle wie wir die Welt wahrnehmen und neue Dinge lernen, z.B. werden ästhetisch schöne Produkte als effektiver wahrgenommen.

Ebenen des emotionalen Designs

Gutes Design war oft darin begründet, dass ein System gebrauchstauglich sein musste. Norman entwickelte sich von dieser Ansichtsweise weg, sodass neben Usability auch andere gewichtige Faktoren gutes Design beeinflussen können. Er spricht von drei Ebenen, die gutes Design ausmachen. Manche Produkte bestechen durch eine bestimmte Ebene, die besten Produkte sprechen uns auf allen drei Ebenen an.

Visceral Design

Wir Menschen sind so beschaffen, dass wir starke emotionale Signale von unserer Umgebung wahrnehmen, und diese automatisch auf der visceralen[4] Ebene

[4] Auch instinktiv. Abgeleitet oder veranlasst durch eine naturgegebene Neigung oder einen Impuls.

interpretieren. Wenn wir etwas als "attraktiv" empfinden, kommt diese Beurteilung direkt von der visceralen Ebene. Attraktiv muss dabei nicht künstlerischen, politischen, oder anderen Forderungen entsprechen. Auf der visceralen Ebene dominieren das Aussehen, das Gefühl, und der Klang. Gute Grafik, Klarheit und Schönheit spielen eine Rolle, die Form, die Gestalt, das physisches Gefühl, das Gewicht; alle Punkte, die eine unmittelbare emotionale Auswirkung auf uns haben (Norman, 2005, S. 65f). Eine kurze Beschreibung mit den Worten Donald Normans lautet:

> *Visceral Design is what nature does.*

Attraktive Dinge betrachten wir nicht nur als schön, wir empfinden sie auch als besser funktionierend, da sie positive Emotionen hervorrufen. Positive Gefühle bewirken mentale Prozesse, die uns kreativer und toleranter gegenüber kleineren Schwierigkeiten machen (Norman, 2005, S. 60).

Behavioral Design

Behaviorales Design hat weniger mit dem Aussehen eines Produkts als viel mehr mit der Anwendbarkeit zu tun. Norman bezeichnet die Funktion, das Verstehen, die Gebrauchstauglichkeit und das physische Gefühl als Komponenten eines Produkts, die bedeutend für gutes Design sind. Gutes behaviorales Design muss von Anfang an ein fundamentaler Teil eines Designprozesses sein, es kann nicht nach der Fertigstellung eines Produkts angewendet werden. Es beginnt mit dem Verstehen der Bedürfnisse der AnwenderInnen und endet damit, dass im Idealfall gründlich getestete Prototypen implementiert werden (Norman, 2005, S. 69f).

Reflective Design

Reflektives Design beinhaltet viele Faktoren, vor allem handelt es von der Botschaft, der Kultur und Bedeutung eines Produkts und deren Anwendung. Das kann beispielsweise ein persönliche Andenken sein, dass ein Produkt hervorruft, oder das vermittelte Bild und die Botschaft, die ein Produkt sendet. Attraktivität ist ein Phänomen der visceralen Ebene, die nur vom oberflächlichen Aussehen eines Objekts bestimmt wird, wohingegen Schönheit auf der reflektiven Ebene liegt. Schönheit liegt auch unter der Oberfläche, Schönheit kommt von bewusster Reflexion und Erfahrung, und wird durch Wissen, Bildung und Kultur

beeinflusst. An der Oberfläche hässliche Objekte können schön sein, etwa atonale Musik oder hässliche Kunst. Reflektives Design liegt dementsprechend im Auge des Betrachters (Norman, 2005, S. 83 f).

Jede dieser drei Ebenen spielt eine entscheidende Rolle im menschlichen Verhalten, es wird durch Design, Marketing oder die Verwendung von Produkten beeinflusst. Das Zusammenspiel der Ebenen ist in Abbildung 2.5 illustriert: Die viscerale Ebene entspricht unserem Instinkt, es wird sofort entschieden, was gut oder schlecht ist. Die höchste Ebene ist die der Reflexion. Dazwischen befindet sich die behavioristische Ebene, die durch die Reflexions-Ebene kontrolliert wird und das Handeln bestimmt.

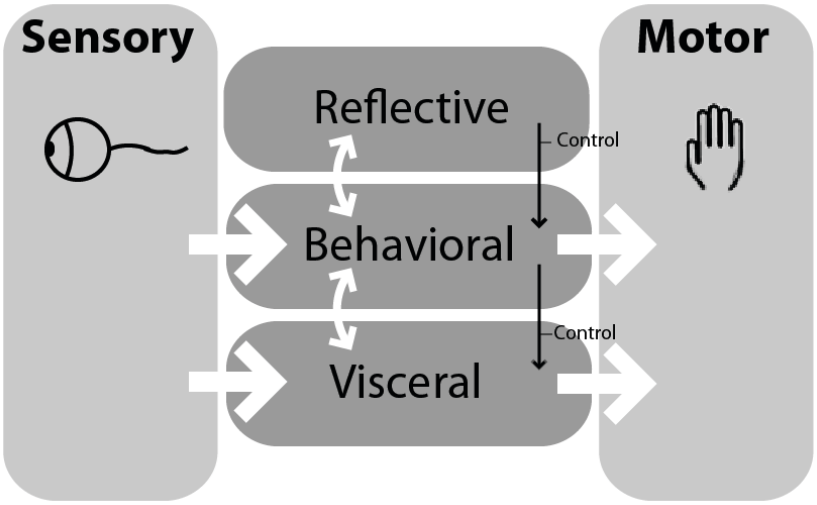

Abbildung 2.5: Das Zusammenspiel der drei Ebenen: Auf der visceralen Ebene wird sofort entschieden, was gut oder schlecht, sicher oder gefährlich ist. Die Aktionen der behavioristischen Ebene können durch die reflektive Ebene gefördert oder behindert werden. Die höchste Ebene ist die der Reflexion, sie hat keinen direkten Einfluss auf die beiden anderen, aber sie passt auf, reflektiert, und beeinflusst die behavioristische Ebene.

Menschliche Reaktionen sind komplex und determiniert durch verschiedenste Faktoren: Manche kontrolliert durch DesignerInnen, dem Hersteller, Werbung oder dem Markenimage des Produkts, andere dieser Faktoren entstehen durch unsere eigenen privaten Erfahrungen. (Norman, 2005, S. 21f)

Zusammenfassung

Der erste Abschnitt dieses Kapitel beschäftigte sich mit der Geschichte der Softwareentwicklung. Diese betrachtet sich verbreitet als Disziplin, die Probleme rationell erfasst und daraus logische Lösungen ableitet. Das entspricht der Vorgangsweise der ersten Generation der Designmethodologie. Probleme beim Design von Softwareprodukten können meist als Wicked Problems definiert werden und können nicht rationell abstrakt gelöst werden, denn Wissen und Tun sind eng miteinander verbunden. Ein weiteres Problem der Softwareentwicklung sind die sich ständig verändernden Anforderungen in einem Projekt. Überdies fehlt ein Bewusstsein für Design als wichtiger Aspekt eines Softwareentwicklungsprozesses, nach wie vor sind oft die ProgrammiererInnen gleichzeitig auch die DesignerInnen eines Softwareprodukts.

Der zweite Abschnitt beschäftigte sich mit der Frage, was gutes Design charakterisiert. Es wurde versucht, Merkmale guten Designs zu schildern und für eine erfolgreiche Designarbeit die Metapher eines dreibeinigen Stuhls heranzuziehen: Nur wenn das Verstehen der verwendeten Technik, der AnwenderInnen und das kreative Verständnis gegeben sind, kann der Stuhl stehen. Emotionen haben ebenfalls eine große Auswirkung darauf, wie wir ein Produkt wahrnehmen. Lange Zeit war vor allem Usability ein Kriterium für gute Produkte, Norman erweitert dies noch um zwei weitere Ebenen, die subjektive Bedeutung in uns und das ästhetische Empfinden.

3 Designprozess

> *We fail more often because we solve the wrong problem, than because we get the wrong solution to the right problem.* ((Ackoff, 1974), zitiert nach (Gasson, 2007))

In diesem Kapitel wird der Designprozess und die Designarbeit genauer erläutert. Die Tätigkeiten die einen Designprozess begleiten, beginnend mit einem Auftrag und im endend mit einem guten Produkt, werden in diesem Kapitel beschrieben. Es wird dargelegt, welche Möglichkeiten und Nutzen sich aus unterschiedlichen Methoden gewinnen lassen. Zu den Methoden gehören etwa das Erforschen des Kontexts um das Produkt, verschiedene Methoden, um Ideen zu finden, diese zu kommunizieren, oder das Produkt durch Prototypen zu testen. Ein Designprozess lässt sich nicht in eine lineare Vorgehensweise hineinzwängen, genausowenig lassen sich Methoden im Vorhinein bestimmen. Diese sind vom Projektkontext abhängig. Bill Buxton betrachtet den Designprozess selbst mehr als Sketch (Buxton, 2007, S. 77) als eine genau definierte Folge von Tätigkeiten, die einen Designprozess leiten.

Im Wesentlichen werden DesignerInnen zu einem Projekt hinzugezogen, wenn etwas neu entwickelt werden soll, nicht richtig funktioniert, weiterentwickelt wird oder sich nicht gut verkauft. Zu Beginn eines Projekts muss der Kontext, in welchem das Produkt entwickelt oder verbessert werden soll, geklärt werden. Dazu soll einE DesignerIn mit allen beteiligten Interessensgruppen sprechen, um die Problemstellung genau zu erfassen. Das heißt, die AuftraggeberInnen sollen in Gesprächen ihre Sichtweise betreffend des Designproblems und der Anforderung darlegen (Saffer, 2006, S. 24).
Bryan Lawson schreibt in seinem Werk "What Designers Know" dazu:

> *First, many people contribute to design problems (...). Obviously they include the client, but they also include users, legislators and the designers themselves. Even more confusingly clients are seldom able to comprehensively state their problem at the outset. In fact clients are often still unable to articulate the whole of the problem from their perspective even at the end of the process.* (Lawson, 2004, S. 13)

Der/die KlientIn spielt nur eine Rolle im Gesamtkontext des Produkts. Eine weitere wichtige Interessengruppe stellen die EndanwenderInnen dar, denn sie werden am Ende das Produkt benutzen. Schließlich soll sich ein Produkt an die Bedürfnisse der UserInnen anpassen, und nicht die UserInnen an das Produkt. Aus diesem Grund ist es unentbehrlich, auch von ökonomischen und effizienten Gesichtspunkten aus betrachtet, die Wünsche und Bedürfnisse der EndanwenderInnen und deren Umgebung zu erfahren, nicht nur die der AuftraggberInnen. Das Designteam soll die BenutzerInnen und die Umgebung der AnwenderInnen vor Ort erforschen, wenn der ganzheitliche Kontext erfahren werden soll.

Die Rolle der AnwenderInnen

DesignerInnen halten sich fälschlicherweise oft für typische UserInnen, da sie ihr eigenes Produkt immer und immer wieder anwenden und testen. Es ist klar ersichtlich, dass DesignerInnen ihre Produkte so gut kennen, dass sie Fehler im Design kaum bemerken. Norman stellt fest, dass Menschen nur Zugang zu deren Bewusstsein haben, aber nicht zum Unterbewussten. Aus diesem Grund tendieren Menschen dazu, deren eigene Rationalisierungen in Aktionen und Meinungen anderer zu projizieren (Norman, 2002, S. 157). Demzufolge beinhaltet professionell durchgeführtes Design sowohl EndanwenderInnen im Designprozess als auch beim Testen. Die Techniken und Methoden des professionellen Designs sind fundamental für den Designprozess und zufriedenstellende Produkte.

Es stehen Methoden zur Verfügung, um Informationen über UserInnen zu erheben, welche aus dem Blickwinkel des User Centered Designs stammen. User Center Design ist auf die Bedürfnisse und Ziele der AnwenderInnen fokussiert, d.h. UserInnen begleiten einen Designprozess, etwa durch regelmäßiges User Testing.

Andere Kategorien außer die des User Centered Designs sind (Saffer, 2006, S. 30) folgende:

- Im Activity- Centered Design liegt der Fokus bei den Aufgaben und Aktivitäten, die erledigt werden (etwa die Aktivität, ein Formular zu erstellen).
- Systems Design konzentriert sich auf die Komponenten eines Systems (z.B. Software Engineering).
- Genius Design verlässt sich auf das Wissen, die Geschicke und Fähigkeiten der DesignerInnen (beispielsweise die Zitronenpresse von Philippe Starck (Starck, 2008))

Design Research

Um zu wissen, wer und was erforscht werden soll, muss eine bestimmte Zielgruppe für das Produkt definiert werden. In diesem Zusammenhang wird auch von Zielgruppenanalyse gesprochen, damit bereits bei der Planung eines Projekts berücksichtigt wird, wer die EndanwenderInnen sein werden, und das Produkt für eine bestimmte Gruppe von UserInnen sinnvoll gestaltet werden kann. In der Literatur wird eine große Anzahl an Methoden beschrieben, mit Hilfe deren Informationen über EndanwenderInnen, Arbeitsweisen, Arbeitsabläufe, Entscheidungsprozesse, andere Produkte, also den Kontext, in den das Produkt eingebettet ist, erfahren werden können. Durch Design Research können sich die bisher bekannten Bedürfnisse und Vorstellungen zuvor definierter Zielgruppen ändern. Bedürfnisse verändern sich in unterschiedlichen Situations- und Produktbedingungen, aus diesem Grund ist es zu empfehlen, individuelle und der Projektabsicht entsprechende Design Researchs durchzuführen. Folgendes Zitat (Stapelkamp, 2007, S. 550) soll dies unterstreichen:

> *Wünsche und Vorstellungen einer Zielgruppe, aber auch die Vermutungen, die man in sie projiziert, bestimmen maßgeblich die Darreichungsform, den Stil, die Informationstiefe und viele andere Eigenschaften des Inhalts und der visuellen, auditiven und funktionalen Gestaltung eines Produkts. Nur wer seine Zielgruppe kennt, kann mit ihr kommunizieren und mit ihr gemeinsame Ziele erreichen.*

Qualitative Untersuchungen eignen sich besonders, um das Fachgebiet, den Kontext und die Einschränkungen eines Produkts zu erforschen. Quantitative Analysen hingegen können nur beantworten, wie oft eine Eigenschaft oder eine Qualität gezählt wird. Menschliche Verhaltensweisen auf Statistiken zu reduzieren, birgt die Gefahr in sich, Wesentliches zu übersehen, was sich wiederum signifikant auf die Merkmale des entstehenden Produkts auswirkt.

Am Anfang eines Projekts werden viele Fragen aufgeworfen: Wen möchte ich mit meinem Produkt erreichen? Was sind die Interessen, Vorlieben, Eigenarten und Besonderheiten meiner Zielgruppe? Welche soziodemographischen, psychologischen, Persönlichkeits- und gebrauchsspezifische Merkmale weisen sie auf (Stapelkamp, 2007, S. 551)?

Research Methods in qualitativer Form können mit Interviews, Beobachtungen, ethnographischen Interviews als Mixtur der beiden, Literaturüberblick, Konkurrenz- und Produktanalyse (Cooper & Reimann, 2003, S. 39ff), und Aktivitäten (Saffer, 2006, S. 70) durchgeführt werden. Im Besonderen helfen sie den DesignerInnen zu verstehen, wie existierende Produkte angewendet werden, und wie sich potentielle UserInnen an Aktivitäten und Probleme annähern, die das zu entwickelnde Produkt behandeln soll. Weiters werden durch qualitative Methoden der technische, geschäftliche und umgebende Kontext - das Fachgebiet - sowie dessen Vokabular und soziale Aspekte, erforscht (Cooper & Reimann, 2003, S. 40). Es wurde nun ausreichend erklärt, warum DesignerInnen möglichst viel über die Eigenschaften und Arbeitsweisen von EndanwenderInnen lernen wollen, um das zu entwickelnde Produkt auf deren Bedürfnisse abzustimmen, die Methoden dazu werden nun genauer beschrieben.

Interviews

Interviews sind eine regelmäßig verwendete Technik, um Design Research durchzuführen. Im weitesten Sinne ist ein Interview eine geführte Konversation mit dem Ziel, verschiedenste Informationen zu erhalten. Es existiert eine Vielfalt an verschiedenen Varianten, abhängig von den Beschränkungen und Anforderungen des Produktkontexts. Der große Vorteil von Interviews besteht darin, dass sie sehr flexibel sind, und sich während des Gesprächs unvermutete neue Aspekte ergeben können. Das Endresultat von Interviews besteht aus der Integration von mehreren individuellen Gesprächen mit unterschiedlichen

Interessensgruppen. Hiermit wird eine ganzheitliche Sichtweise auf ein Projekt, einen Prozess, eine Organisation ermöglicht (Courage & Baxter, 2004, S. 23f).

Abhängig von der Information und InterviewpartnerInnen müssen auch die Ziele eines Interviews unterschiedlich abgesteckt sein. Alan Cooper unterschiedet zwischen Stakeholder, Subject Matter Expert (SME), und User and Customer Interviews. Stakeholder Interviews sind wesentlich, um den geschäftlichen und technischen Kontext des Produkts zu erkennen. Wenn weiteres ExpertInnenwissen zum Verständnis benötigt wird, können überdies noch SME Interviews durchgeführt werde. Das Designteam erhält hierdurch mehr Fachwissen durch ExpertInnen über das Designgebiet. Interviews mit UserInnen bzw. KundInnen sind durch zweierlei Herangehensweisen bestimmt und dürfen nicht verwechselt werden, da die Gruppen unterschiedliche Ziele verfolgen. Unweigerlich muss der Hauptfokus des Design Researchs auf den UserInnen liegen, denn sie werden das Produkt am Ende verwenden. KundInnen hingegen vertreten als potentielle KäuferInnen andere Interessen, sie treffen die Kaufentscheidungen. Etwa wenn einE AbteilungsleiterIn Software erwirbt, die die MitarbeiterInnen, ohne die Möglichkeit bei der Kaufentscheidung mitbestimmen zu können, verwenden werden (Cooper & Reimann, 2003, S. 40f).

Beobachtungen

Die meisten Menschen sind nicht fähig, ihre eigenen Aktivitäten zu beschreiben, besonders außerhalb des Kontexts. Bei Interviews werden schlechtere Resultate erstellt, weil AnwenderInnen ihre Tätigkeiten subjektiv beschreiben und nicht den ganzheitlichen Zusammenhang erfassen können (Cooper & Reimann, 2003, S. 43). Infolgedessen ist es aufschlußreicher, Menschen, und im Besonderen die Aktivitäten, die sie ausführen, zu beobachten. EinE DesignerIn kann versuchen, "unsichtbar" zu werden und nur zu beobachten, sich etwa als normale KundIn in einem Geschäft zu bewegen um andere zu beobachten. Weiters kann viel Information gesammelt werden, wenn ein Subjekt (das aber davon wissen sollte!) einen Tag über wie ein Schatten verfolgt wird (Saffer, 2006, S. 78).

Ethnographische Interviews

Bei ethnographischen Interviews wird einE AnwenderIn sowohl interviewt als auch beobachtet. Kontextabhängige Untersuchungen basieren auf dem

Meister-Lehrling-Konzept. Dabei hat der/die Untersuchte die Rolle als Meister, und der Interviewer die Rolle als Lehrling inne. Beyer und Holtzblatt nennen vier grundlegende Prinzipien für ethnographische Interviews ((Cooper & Reimann, 2003, S. 44f), zitiert nach (Beyer & Holtzblatt, 1997, S. 42f)):

1. **Kontext:** Das ethnographische Interview muss in der Umgebung der zu untersuchenden Person und nicht an einem anderen Ort durchgeführt werden. Dies ist von Belang, da nur eine Untersuchung in der gewohnten Umgebung, mit allen dazugehörigen Aktivitäten und Artefakten, entscheidende und interessante Details hervorbringt.

2. **Partnerschaft:** Das Interview soll im Stil einer gemeinschaftlichen Erkundung stattfinden, abwechselnd zwischen Beobachtung der Aktivitäten und Diskussion bzw. Interview.

3. **Interpretation:** Die Aufgabe der DesignerInnen besteht auch darin, die gesammelten Daten und Beobachtungen zu interpretieren. Dabei müssen sie die Fähigkeit haben, zwischen den Zeilen zu lesen, und Aussagen und Handlungen zu interpretieren. Allerdings ist Vorsicht geboten, Annahmen ohne Bestätigung durch UserInnen zu machen.

4. **Fokus:** Der/die DesignerIn soll das Interview in eine Richtung lenken, die relevante Daten für den Designsachverhalt liefert. Die untersuchte Person soll weder ziellos zwischen Aktivitäten umherwandern können, noch soll eine vordefinierte Struktur das ethnographische Interview leiten, um den explorativen Charakter nicht durch feste Strukturen zu unterbinden

Ferner sollen DesignerInnen auf die Ziele der Untersuchung fokussieren, erst danach auf spezifische Aufgaben. Weiters ist es wichtig, die TeilnehmerInnen zu motivieren Geschichten zu erzählen und Diskussionen über Technologie zu vermeiden (Cooper & Reimann, 2003, S. 49f).

Literaturüberblick, Konkurrenz- und Produktanalyse

Das Designteam sollte neben Interviews mit Interessensgruppen Literatur zum betreffenden Produkt sammeln. Sei es nun Recherche durch das Web, Papers, Produktmarketingpläne, Marktforschung oder andere Quellen, die zum Verständnis des Fachgebiets beitragen. Zusätzlich zur Literatursammlung ist es

hilfreich, einen existierenden Prototypen des Produkts oder Konkurrenzprodukte zu begutachten, um einen Eindruck vom aktuellen Stand zu bekommen und Fragen für die Interviews aufzuwerfen (Cooper & Reimann, 2003, S. 44).

Aktivitäten

Untersuchte werden nicht nur beobachtet und ein Gespräch mit ihnen geführt, es werden ferner auch Aktivitäten mit den UserInnen unternommen, in denen ein Artefakt erzeugt wird. Dieser Prozess kann Emotionen hervorbringen und so zum Verständnis beitragen, wie UserInnen über eine Sache denken und an sie herangehen. Aktivitäten beflügeln die Kreativität der TeilnehmerInnen und erlauben es ihnen, sich anders zu artikulieren als dies in einem Interview möglich wäre. Anschließend ist es wesentlich, dass die TeilnehmerInnen das erzeugte Artefakt erklären, um deren Gedanken und Intentionen nachvollziehen zu können, da diese oft nicht selbsterklärend sind (Saffer, 2006, S. 81f).

Zu sehen ob die Erwartungen mit dem eigenen Produkt erfüllt werden, soll die anvisierte Zielgruppe an der Entwicklung des Projektes teilhaben. Entsprechende Methoden wie etwa User Testing sollen im Idealfall den Designprozess begleiten.

Sketching and reflection-in-action

Für DesignerInnen passiert der Akt der "reflection-in-action" (Schön, 1990) üblicherweise durch Zeichnen von Sketches in deren virtueller Welt, einer Repräsentation der realen praxisnahen Welt. Der Begriff der "reflection-in-action" soll durch folgendes Zitat näher erläutert werden (Smith, 2007):

> *We test out our 'theories' or, as John Dewey (Anm.: zitiert nach (Dewey, 2008)) might have put it, 'leading ideas' and this allows to develop further responses and moves. Significantly, to do this we do not closely follow established ideas and techniques - textbook schemes.*

In der virtuellen Welt des "Austestens" ist es möglich, die der realen Welt immanenten Beschränkungen zu überwinden, und somit frei zu experimentieren und Hypothesen zu testen. Das experimentieren und ausprobieren in einer virtuellen Welt, auch wenn es nur Stift auf Papier ist, ist unumgänglich für den

Designprozess. Denn Wissen und Tun sind untrennbar miteinander verbunden (Schön, 1990, S. 75f).

> *Seeing-as is not enough, however. When a practitioner sees a new situation as some element of his repertoire, he gets a new way of seeing it and a new possibility for action in it, but the adequacy and utility of his new view must still be discovered in action. Reflection-in-action necessarily involves experiment.* (Schön, 1990, S. 68)

Beim Sketching entstehen in kurzer Zeit visualisierte Ideen (meist skizziert mit Stift auf Papier), die später verwendet werden können. Es erlaubt den DesignerInnen viele Ideen auszuprobieren, bevor sie zu einer zeit- oder ressourcenintensiveren Methode übergehen. Grundsätzlich bestehen die essentiellsten Werkzeuge eines/einer Designers/in in Papier und Stift. Es werden meist keine digitalen Werkzeuge beim Sketching verwendet, da der Ideenfindungsprozess flexibel sein soll. Papierstücke, die zusammengeheftet oder collagenartig angeordnet werden, erlauben viel Freiheit und Flexibilität in der Gestaltung. Es scheint, dass das taktile Gefühl mit den Materialen eine ganz andere Form der Interaktion und Reflexion erlaubt, als dies digitale Werkzeuge ermöglichen. Eine andere Form bzw. Möglichkeit ist das Modellieren mit verschiedensten Materialien, um in kurzer Zeit ein Modell zu kreieren. Das Erzeugen von Sketches und Modellen soll während dem ganzen Designprozess permanent in die Designarbeit einfließen, um Ideen zu visualisieren, und mit anderen kommunizieren zu können (Saffer, 2006, S. 102).

Wiederum eine Art von Sketching, die gerade im Zusammenhang mit Interaktion sehr interessant scheint, ist das Interaction Sketching bzw. Video Sketching. Interaction Sketching erklärt ein Produkt in narrativer Weise: Es werden Fotos gemacht, in denen Mockups verwendet werden um ein Produkt zu repräsentieren, das im Sketch erklärt werden soll. Danach werden die Fotos in einer Art Präsentation hintereinander gereiht und als Video gezeigt. So ist es möglich, die wichtige Rolle des Zusammenspiels von Zeit und Verhalten im Erfahren eines Produkts sichtbar zu machen (Buxton, 2007, S. 299). Ein anderer Vorteil von Interaction Sketches besteht darin, dass sie schnell entworfen werden können, um eine Idee skizzenhaft zu repräsentieren. Sie verlangen durch das

detaillierte Arbeiten, dass sich die MacherInnen intensiv mit dem Konzept auseinandersetzen, und stellen somit ein effektives Werkzeug dar, Ideen und Reflexionen anzuregen.

Weitere Designmethoden

Der Vollständigkeit wegen sollen an dieser Stelle kurz andere Designmethoden beschrieben werden, die im Projekt nicht zur Anwendung kamen, aber durchaus relevant sind.

Storyboarding

Storyboarding ist eine Technik, die von FilmemacherInnen und der Werbung entstammt. Heutzutage wird die Methode des Storyboarding in der HCI und im Interaktionsdesign angewendet, um Designideen bzw. Konzepte kontextbezogen zu präsentieren. Ein Storyboard ist eine kurze visuelle Schilderung einer Erzählung, um verschiedenste Aktivitäten darzustellen (Saffer, 2006, S. 103).

Task Analysis

Eine Task Analysis ist eine Auflistung von Aktivitäten und Aufgaben, die das endgültige Produkt unterstützen und ausführen soll. Je nach Komplexität des Projektes können dies einfache oder schwierige und unübersichtliche Aufgaben sein. Eine Task Analysis ist hilfreich um nachzuprüfen, ob alle Aufgaben vom System in einer bestimmten Zeitspanne erledigt werden können (Saffer, 2006, S. 104).

Mood Board

Ein Mood Board ist ähnlich einem Poster gestaltet, welches aus Bildern, Text und Beispielen von Objekten collagenartig zusammengesetzt ist. Es werden so Emotionen und Gedanken des/der DesignerIn zum Produkt repräsentiert. Mood Boards sollen zum Entwurf und zur Kommunikation von Konzepten dienen. Das Arbeiten mit Mood Boards beinhaltet auch, dass bereits während des Designprozesses ein bestimmter graphischer Stil, ein Gefühl für das Produkt, gefunden werden kann (Saffer, 2006, S. 108).

Personas

1983 wurden Personas von Alan Cooper als Werkzeug eingeführt, um Archetypen von AnwenderInnen zu entwickeln, und DesignerInnen eine Vorstellung zu vermitteln, für wen sie ein Produkt gestalten. Sie sollten DesignerInnen als Hilfe dienen, um Bedürfnisse und Wünsche der zukünftigen AnwenderInnen betreffend des Produktes oder des Services zu erkennen (Cooper & Reimann, 2003, S.55). Im Wesentlichen ist eine Persona ein Portrait einer fiktionalen Person mit einem Gesicht, einem Namen, Familie, FreundInnen, Hobbys, Beruf, folglich demographischer Information und sozioökonomischem Status. Ferner hat eine Persona eine Lebensgeschichte und aufgabenspezifische Ziele und Bedürfnisse. Eine Persona muss für das Designteam als eigene Persönlichkeit realistisch vorstellbar sein.

Scenarios

Scenarios sind Geschichten, in diesem Sinn sind sie Prototypen, die mit Wörtern erzählt werden. Diese Geschichten haben als ProtagonistInnen Personas, welche bestimmte Ziele und Vorstellungen haben, sie spielen vor einer bestimmten Kulisse und haben Handlungen bzw. Sequenzen von Aktionen. Scenarios stellen einen effektiven Weg dar, Designkonzepte zu imaginieren. DesignerInnen welche Scenarios verwenden, können in Wörtern Sketche erstellen, was in manchen Fällen sinnvoller und dienlicher ist als z.B. Storyboarding, vor allem betreffend den zeitlichen Aufwand (Saffer, 2006, S. 101).

Prototyping

Der letzte Schritt vor der Fertigstellung eines Produktes besteht darin, es mit Hilfe von einem oder mehreren Prototypen an AnwenderInnen zu testen. Im Idealfall passiert dieser Schritt schon während des Designprozesses iterativ, bis das Ergebnis den Ansprüchen genügend ist. Wie andere schon besprochene Methoden ist auch das Prototyping eine Methode, damit zwischen den Teilnehmenden am Designprozess, seien es nun ExpertInnen oder zukünftige AnwenderInnen, kommuniziert werden kann. Prototypen vermitteln die Botschaft: „So könnte das fertige Produkt aussehen". Im Allgemeinen liegt in der Anwendung des Prototyping mehr potentielle Kreativität, weil mehrere TeilnehmerInnen im frühen Designprozess involviert sind (Saffer, 2006, S. 114).

Diese Liste an Methoden stellt keinen Anspruch auf Vollständigkeit, Designmethoden werden ständig weiterentwickelt und neu erforscht. Etwa wie der *Interactive Thread*, eine multidisziplinäre Designmethode, die eine Form von partizipatives Design mit vielen Involvierten darstellt. Es wurden TeilnehmerInnen auf einer Konferenz gefragt, gleichzeitig als AnwenderInnen und DesignerInnen zu agieren, und eine Auswahl von Aufgaben für Forschungszwecke zu erarbeiten. Aus diesen Aufgaben haben sich für das Produkt interessante Ergebnisse ergeben. Die Methode ist dazu gedacht, dass TeilnehmerIlnnen gleichzeitig als ExpertInnen (etwas auf einer Ärztekonferenz), agieren (Mackay, 2004).

Auch bei *Inspiration Card Workshops* können viele Ideen gmeinsam mit AnwenderInnen gesammelt werden, und so als Inspirationsquelle fungieren. Der Inspiration Card Workshop stellt eine Methode dar, Erkenntnisse von verschiedenen Domänen, repräsentiert durch Domain Cards, und Ideen aus technischen Anwendungen, repräsentiert durch Technology Cards, miteinander zu kombinieren. Die WorkshopteilnehmerInnen können dabei die Domain und Technology Cards beliebig anordnen, und ihrer Idee mit weiteren Kommentaren und Zeichnungen versehen (Halskov & Dalsgard, 2006).

Einen gänzlich andere Ansatz haben Höfnagels et al. in einem Designprojekt verwendet: sie gebrauchten *"Reibung" als Metapher* für das Design eines Produkts, mit dem Familien ihren Alltag planen können. Da in Familien durch den Alltag gemeinsame Zeit schwierig zu finden ist, wurde Reibung als Metapher verwendet, und durch abgeleitete Prinzipien das Design durch diese Metapher geleitet. Die Verwendung einer Metapher ermöglicht einen neuen Blick auf Lebensstile und unterstützt, Situationen anders und ungewohnt zu interpretieren (Hoefnagels, Geelhoed, Stappers, Hoeben & Lugt, 2004).

Der Erfolg von Designmethoden hängt zum größten Teil von den DesignerInnen selbst, den TeilnehmerInnen und deren Zusammenarbeit ab. Unterschiedliche Designkontexte erfordern unterschiedliche Methoden, auch abhängig von den Vorlieben, Erfahrungen und Fertigkeiten der DesignerInnen.

User Testing

Obwohl beim Projekt die Zeit und Ressourcen nicht ausreichten, um ein User Testing durchzuführen, soll dieser wesentliche Aspekt eines

Softwareentwicklungsprozesses hier dennoch behandelt werden. User Testing wird in der Literatur meist in einem Zug mit Usability Testing genannt (Stone, Jarrett, Woodroffe & Minocha, 2005, S. 423), (Galitz, 2002, S. 702); Usability versucht die Gebrauchstauglichkeit eines Produkts zu "messen". Jakob Nielsen definiert fünf Eigenschaften eines Produkts: Erlernbarkeit, Effizienz, Einprägsamkeit, wenige Fehler und Zufriedenheit der AnwenderInnen mit dem Produkt (Nielsen, 2000, S. 26). Es stellt sich nun die Frage, wann BenutzerInnen mit einem Produkt zufrieden sind. Die Betrachtungsweise von Nielsen hat sicher seine Berechtigung, und ein Produkt soll im besten Fall diese Eigenschaften besitzen, sie sind aber zu eindimensional. Es reicht es nicht aus, nur die Usability eines Produkts zufriedenstellend zu gestalten. Produkte können Menschen laut Donald A. Norman auf zwei weiteren Ebenen faszinieren (vgl. Kap. 2, S. 22). Abgesehen davon, dass die Zufriedenheit mit einem Produkt subjektiv ist, trägt nicht nur die Usability eines Produkts zur Faszination bei, sondern die ganze User Experience.

Bei einem User Test wird das zu testende System empirisch von den AnwenderInnen anhand realistischer Aufgaben erprobt und dabei von UsabilityexpertInnen, DesignerInnen, ProgrammiererInnen, u.a. beobachtet. Aus Beobachtungen und Äußerungen der AnwenderInnen während der Tests und in anschließenden Interviews können Schlussfolgerungen über Probleme und Verbesserungsmöglichkeiten eines Produktes analysiert werden (Sarodnick & Brau, 2006, S. 155).

Wie bei allen Projekten ist es wichtig, vor dem Testen den Zweck und die Zielsetzungen des Tests zu eruieren. Die meisten Tests setzen ferner voraus, dass schon Entwürfe auf Papier oder in digitaler Form existieren. Die Eigenschaften der verschiedenen Methoden beeinflussen auch, wann sie im Designprozess eingesetzt werden. Ein interessanter Aspekt dabei ist, dass Untersuchungen gezeigt haben, dass die Anzahl der gefundenen Usabilityprobleme nicht damit zusammenhängt, ob Papierprototypen oder weit entwickelte und realitätsnahe Prototypen verwendet wurden (Virzi, Sokolov & Karis, 1996). Es ist nicht immer notwendig, möglichst weit entwickelte Prototypen zu testen, da es neben einem großen Arbeitsaufwand dazu führen kann, dass wenig Diskussion stattfindet. Bei einfacheren Prototypen fällt es den Testpersonen einfacher, Kritik hinsichtlich der Funktionalität anzubringen und nicht auf einer stilistischen Designebene zu argumentieren. Bie Bandbreite an Prototypen im Allgemeinen und für User Tests

ist sehr groß: Es können Storyboards, Paper Mock-ups, Wizard of Ozing, Videoprototypen, Computeranimationen, Scenarios und andere Techniken eingesetzt werden (Sarodnick & Brau, 2006, S. 158).

Ein weiterer wichtiger Aspekt von User Tests ist die Auswahl geeigneter Testpersonen. Diese sollen die Zielgruppe des Systems und zukünftigen AnwenderInnen angemessen repräsentieren. Weiters sollten die Testpersonen das zu testende System nicht kennen, da sonst viele Probleme übersehen bzw. umgangen werden würden. Laut Untersuchungen reichen 5-6 Testpersonen aus, um 80% der Usabilityprobleme aufzudecken (Sarodnick & Brau, 2006, S. 159).

Für die Testpersonen sollen möglichst typische, realistische und/oder kritische Aufgaben ausgewählt werden (Dumas, 1989). Die auszuführenden Aufgaben bei einem Test sollen, wie die Testpersonen auch, so repräsentativ wie möglich sein. Sie sollten die wichtigsten Aspekte des User Interfaces der Software abdecken. Die Aufgaben sollen aber in einem vernünftigen Zeitintervall erledigbar sein, vor allem von Testpersonen, die nicht mit dem System vertraut sind. Andererseits sollen die Aufgaben aber auch nicht zu trivial sein, denn die Testperson soll sich nicht unterfordert und somit nicht ernst genommen fühlen (Nielsen, 2000, S. 158).

Es spielt überdies eine wichtige Rolle, ob Tests in einer realen (Field Studies) oder in einer künstlichen Umgebung (Controlled Studies) stattfinden. Auch wenn Field Studies mehr Organisationsaufwand mit sich bringen, haben sie einen entscheidenden Vorteil: Die Testpersonen können in ihrer eigenen Umgebung beobachtet und getestet werden (Stone et al., 2005, S. 468). Ein Usability Labor ist zwar effizienter, führt aber zu der Einschränkung, dass in einem Labor weniger Probleme sichtbar werden als bei Field Studies (Duh, Tan & Chen, 2006). AnwenderInnen können unvermutete Aspekte artikulieren, die vorher nie betrachtet wurden. Denn schließlich werden Produkte am Ende auch in wirklichen, und nicht künstlichen Umgebungen verwendet; Field Studies sind näher an der Realität angesiedelt. Allerdings haben Laboratory Studies in manchen Fällen Vorteile, beispielsweise wenn zwei Systeme unter kontrollierten Bedingungen verglichen werden sollen (Dix, Finlay, Abowd & Beale, 2003, S. 328).

Es muss natürlich auch während des Tests aufgezeichnet werden, wie die Testpersonen die Aufgaben absolvieren. So besteht im Nachhinein noch die Möglichkeit der Analyse. Obschon können während des Tests Beobachtungen,

z.B. durch lautes Denken, angestellt werden. Die Tests sollen mit Video aufgezeichnet, und die Testpersonen zum lauten Mitdenken motiviert werden. Außerdem können auch noch Log Files und Eyetracking angewendet werden (Sarodnick & Brau, 2006, S. 158).

Kritik

Usability Tests werfen die Frage auf, ob diese wirklich aussagekräftig sind. Dicks argumentiert, dass Usability Tests inhärente Grenzen haben (Dicks, 2002):

- Das Testen geschieht immer in einer künstlichen Situation, besonders in einem Usability Labor.
- Gute Testergebnisse beweisen nicht, dass ein Produkt gebrauchstauglich ist.
- Testpersonen sind selten wirklich repräsentativ für die Zielgruppe.
- Testen ist nicht immer die beste anzuwendende Methode.

Trotzdem soll der Stellenwert eines Usability Tests nicht gänzlich vernachlässigt werden, wie auch Rubin in kurzen Worten beschreibt:

> *It's better to test than not to test* - (Rubin, 1994)

Weiters stellt sich auch die Frage der Rolle der Testpersonen im kreativen Designprozess. Bei User Tests nur Produkte zu testen, anstatt mit den Testpersonen einen Dialog über das zu entwickelnde Produkt zu führen, ist nicht immer der beste Ansatz, um Kreativität zu fördern. Testpersonen können als Inspirationsquelle dienen und sehr wohl brauchbare Ideen für ein Design liefern (Buur & Bagger, 1999). Ergänzend soll noch folgendes Zitat von Bill Atkinson das Unterkapitel abschließen.

> *I learned from MacPaint, that in order to get a piece of Software to be smooth, you must start over a number of times. You need to test it on a lot of different people and have them use the program. A lot of what i would do was just watch Susan Kare using it. "What is that like you'd like to be able to do?" I would ask her, "What's the most frustrating thing about this?"*
> *Then I would go back and see what I could do about that. I think that the more User Testing a piece of Software has, the smoother it can become.* - Bill Atkinson (Moggridge, 2006, S. 102)

Zusammenfassung

Ein Designprozess unter dem Blickpunkt des User Centered Designs involviert neben anderen Interessensgruppen wie ManagerInnen oder AuftraggeberInnen, vor allem die EndanwenderInnen. Analog zu Buxton soll einem Entwicklungsprozess eine preproduction Phase vorangehen, in welcher Bedürfnisse und Interessen der AnwenderInnen und der Projektkontext erforscht werden, bevor mit der Implementierung eines Systems begonnen wird. Unterschiedliche qualitative Methoden, die Bedürfnisse der AnwenderInnen zu erforschen, wurden dargelegt. Es ist zu beachten, dass sowohl Design Research Methoden als auch User Tests mit wenigen Ausnahmen in der gewohnten Umgebung der UserInnen am besten aufgebhoben sind, da sie näher an die Realität und den Projektkontext herankommen. Die Wahl der Designmethoden ist sowohl von den Eigenschaften des Projekts, als auch der Erfahrung und Vorlieben der DesignerInnen abhängig. Es werden ständig neue Methoden erforscht, die die Bandbreite erweitern; Auch ungewöhnliche Ansätze, etwa einer Metapher, die einen Designprozess leitet, finden sich in der Literatur. Im Idealfall wird ein Designprozess von User Tests begleitet, die in der Literatur oft in einem Zug mit Usability Tests genannt werden. Es wurde versucht, diese beiden Begriffe zu unterscheiden bzw. voneinander abzugrenzen. Usability bezieht sich einzig auf die Gebrauchstauglichkeit eines Produkts, und stellt einen Faktor der ganzheitlichen User Experience dar. User Tests müssen nicht zwingend nur die Usability eines Produkts messen, es können durch ein Gespräch mit den Testpersonen neue Lösungvorschläge und Ideen auftauchen.
In diesem Kapitel war der Hauptaugenmerk auf den Designprozess von Produkten gerichtet. Im nächsten Kapitel wird auf die speziellen Aspekte des User Interface Designs bei Softwareapplikationen eingegangen.

4 User Interface Design

> *"Das Prinzip der gotischen Architektur (...) ist die vorstellbar gemachte Wirklichkeit." Das gleiche ließe sich über das moderne Interface sagen. Wo die schwebenden Stützpfeiler der Kathedrale von Chatres das Himmelreich in Stein gehauen wiedergeben, verkörpert der Informationsraum auf dem Monitor den sonst unsichtbaren Tanz von Nullen und Einsen, die durch unsere Mikrochips wirbeln - "er macht ihn vorstellbar".* (Johnson, 1999, S. 54)

In diesem Kapitel werden Aspekte, die beim Design von Interfaces zu beachten sind, und Richtlinien, etwa die acht goldenen Regeln des User Interface Designs (verstanden als Orientierungshilfe, weniger als Direktive), erläutert. Im Besonderen soll das Design von Editoren berücksichtigt werden, da im Projekt (vgl. Kap. 5) ein Editor bzw. Dokument Designer entwickelt wurde. Zunächst aber soll der Begriff des User Interface Designs im Kontext anderer Disziplinen betrachtet werden.

Interface Design im Kontext der User Experience

In Abbildung 4.1 ist illustriert, dass User Interface Design einen Teil der gesamten User Experience darstellt. Demzufolge beinhaltet User Interface Design nicht nur das Designen der Benutzeroberfläche, es handelt auch davon, die ganzheitliche Erfahrung und das Erleben mit dem Produkt zu gestalten. Zur User Experience tragen auch das Interaktionsdesign, die Informationsarchitektur, Kommunikationsdesign, Usability und HCI bei.

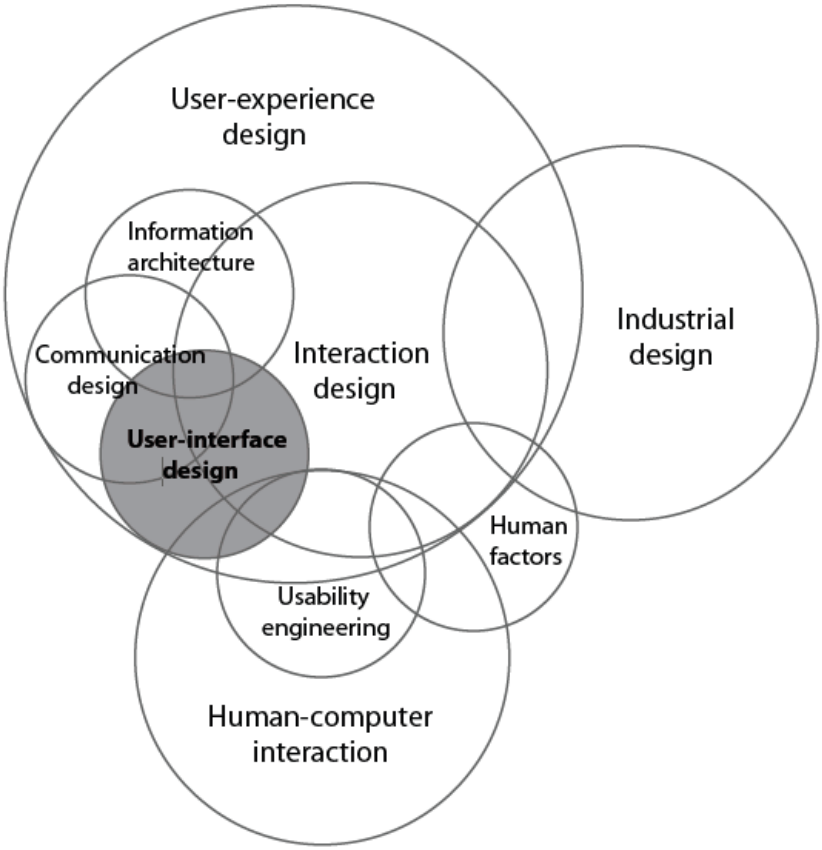

Abbildung 4.1: Die Zusammenhänge zwischen unterschiedlichen Disziplinen unter besonderer Berücksichtigung des User Interface Designs. Alle Disziplinen sind Teil der ganzheitlichen User Experience und des Interaktionsdesigns. (Quelle: Saffer, 2006, S. 17)

Die Auffassung des User Experience Designs ist dadurch gekennzeichnet, dass sie nicht nur einen einzigen Aspekt behandelt, sondern viele Faktoren ein Produkt beeinflussen. User Experience Design kann in fünf Ebenen eingeteilt (Garrett, 2002, S. 33) werden (vgl. Abbildung 4.2):
Auf der untersten und abstraktesten Ebene, der strategischen, liegen die Anforderungen der AnwenderInnen, und von außen bestimmte Ziele, etwa geschäftliche Bedingungen. Auf der darüberliegenden Ebene, die des Anwendungs- und Handlungsbereichs, wird die funktionelle Spezifikation definiert, d.h. welche Anforderungen das Produkt erfüllen soll, welche Aspekte nicht implementiert werden sollen (wichtig, um nicht zuviele Features und Funktionen zu implementieren). Eine weitere Ebene darüber, die Struktur der

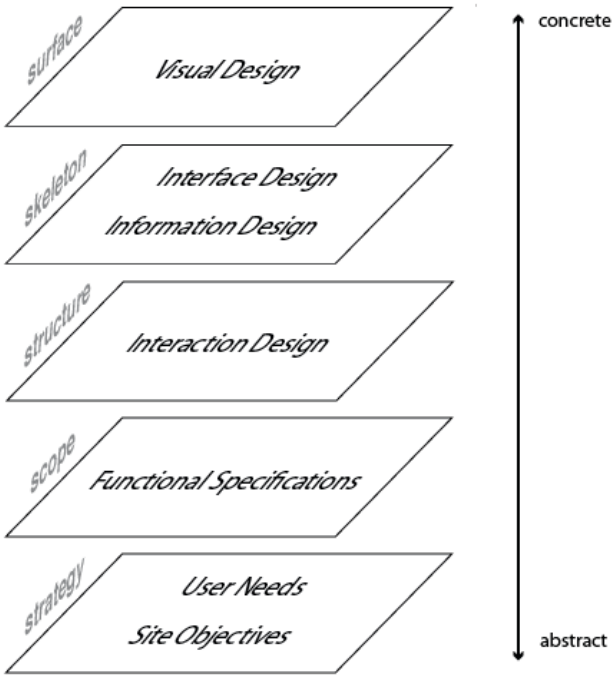

Abbildung 4.2: User Experience Design kann in fünf Ebenen strukturiert werden. (Quelle: Garrett, 2002, S. 33)

Applikation, beinhaltet das Interaktionsdesign, dass sich mit der Beschreibung möglicher Verhaltensweisen von UserInnen und Feedback des Systems beschäftigt. Die vierte und vorletzte Ebene wird als Skelett bezeichnet. Diese Ebene beinhaltet sowohl das Design des Interfaces, das den AnwenderInnen als "Fenster" dient Aktionen zu setzen, als auch verständliche Informationsdarstellung um eine effektive Kommunikation mit dem Interface zu gewährleisten; Das Interaktionsdesign ist bedingt durch das Informationsdesign. Die oberste und somit konkreteste Ebene ist die, die sich mit der Oberfläche beschäftigt: In dieser wird das visuelle Design bestimmt. Hier spielen nicht nur ästhetische Faktoren eine Rolle, es soll auch auf den Stil und die Effektivität (z.B.: sind die Icons eindeutig?) des Designs geachtet werden (Garrett, 2002, S. 40ff).

Gutes User Interface Design

Gute Software soll im besten Fall AnwenderInnen dabei unterstützen, eine Tätigkeit in kurzer Zeit bestmöglich zu erledigen. Gute Software soll UserInnen ihre Ziele verfolgen lassen, sodass die Arbeitsschritte am Computer selbst in den Hintergrund rücken, und sie sich vollkommen auf ihre Arbeit konzentrieren können. Bei gutem Softwaredesign stehen die Menschen im Mittelpunkt, die die Software letztendlich verwenden und mit dieser interagieren, nicht technische Spezifikationen, Softwareentwicklungsprozesse oder Managemententscheidungen, die an den Bedürfnissen der AnwenderInnen vorbeigehen.

Demzufolge stehen am Anfang einer Produktentwicklung zukünftige AnwenderInnen und deren sozialer Kontext, Werte, Vorstellungen und Bedürfnisse (vgl. Kap. 3). Speziell bei Softwareprodukten muss es für DesignerInnen von Interesse sein, welche Aufgaben damit erledigt werden sollen, und in welchem größeren Zusammenhang. Tidwell beschreibt folgende Aspekte, die DesignerInnen über AnwenderInnen lernen sollen (Tidwell, 2005, S. 6):

- deren Ziele bei der Verwendung der Software
- die spezifischen Aufgaben, die unter Berücksichtigung der Ziele bewerkstelligt werden
- die Sprache und Worte, die AnwenderInnen verwenden, wenn sie beschreiben, was sie tun
- die Fähigkeiten der AnwenderInnen bei ähnlicher Software wie der Entstehenden
- die Einstellung zur entstehenden Software, und wie unterschiedliche Designs die Haltung beeinflussen können

Ein Interface kann in gewissem Sinne als Erweiterung einer Person betrachtet werden das es ermöglicht, Handlungen zu setzen. Geeignete Software reflektiert die Fähigkeiten einer Person und entspricht ihren Anforderungen (Galitz, 2002, S. 40). Um den Anforderungen bestens genüge zu tun, können die acht goldenen Regeln des Interface Designs herangezogen werden.

Die acht goldenen Regeln der User Interface Designs

Richtlinien stellen nicht den Anspruch, vollständig bzw. perfekt auf ein System abgestimmt zu sein. Sie dienen als Anhaltspunkte, weniger als allumfassende Prinzipien, die gemeinhin und ohne Ausnahme gelten. Dennoch sind die im folgenden aufgelisteten Regeln des Interface Designs für fast jedes interaktive System anwendbar, und haben sich daher als eine sehr wertvolle Richtlinie erwiesen (Shneiderman & Plaisant, 2005, S.74 f):

1. **Bemühen um Konsistenz:** Es existieren viele Formen von Konsistenz, daher wird diese Regel fortwährend missachtet. Konsistenz soll beim Ablauf von Aktionen in ähnlichen Situationen gegeben sein (etwa wie ein Dialog geführt wird). Konsistenz heißt auch, dass bei Prompts, Menüs, und Hilfen identische Terminologie verwendet werden soll. Dasselbe Prinzip gilt für verwendete Farben, Schriften, Layouts, usw., diese sollen durchgehend angewendet werden.

2. **Auf allgemeine Usability eingehen:** Die Bedürfnisse der diversen AnwenderInnen sollen erkannt und im Design berücksichtigt werden. BeginnerInnen und ExpertInnen haben unterschiedliche Anforderungen die das Design leiten sollen. Durch das Hinzufügen von relevanten Features wie Hilfen für AnfängerInnen und Shortcuts für ExpertInnen, kann das Interface Design bereichert und verbessert, als auch die wahrgenommene Produktqualität gesteigert werden. Desweiteren haben Alter, Behinderungen, technische Vielfältigkeit usw. einen großen Einfluss auf die Anforderungen eines Systems.

3. **Aufschlussreiches Feedback anbieten:** Es soll für jede durchgeführte Aktion hilfreiches Feedback für die AnwenderInnen stattfinden. Für regelmäßige und unwesentliche Aktionen kann die Reaktion weniger Aufmerksamkeit erregen als bei unregelmäßigen und wichtigen. Die Änderungen sollen direkt bei den Objekten angezeigt werden (direkte Manipulation), da dadurch die Systemumgebung angenehmer und leichter verständlich wird.

4. **Dialoge, die einen Abschluss ergeben:** Die Abläufe von Aktionen sollen in Gruppen organisiert sein, die einen Anfang, Zwischenschritte und ein Ende haben. Aufschlussreiches Feedback nach Beendigung einer Gruppe von Aktionen stellt die AnwenderInnen zufrieden, da sie unmittelbar sehen, die

Aufgabe bewältigt zu haben. Es lässt sie Erleichterung empfinden und gibt das Signal, sich auf die nächsten Aktionen vorzubereiten.

5. **Fehler vermeiden:** So weit es geht, sollte ein System so geplant sein, dass AnwenderInnen keine Fehler machen können. Wenn einE UserIn einen Fehler macht, soll das Interface den Fehler erkennen und einfache, konstruktive und präzise Anleitungen zur Recovery anbieten. Ferner sollen fehlerhafte Aktionen den Systemstatus unverändert lassen.

6. **Einfache Umkehraktionen erlauben:** Das Wissen, jede Aktion rückgängig machen zu können, baut Ängste der AnwenderInnen ab, Fehler zu machen. Mehr noch werden sie so dazu ermutigt, unbekannte Möglichkeiten und Optionen auszuprobieren und kennenzulernen. Reversible Aktionen können einfache Aktionen sein, aber auch Gruppen von Aktionen.

7. **Interne Kontrollüberzeugung[1] unterstützen:** AnwenderInnen verlangen danach für ein System verantwortlich zu sein und ihm vertrauen zu können, denn dies verleiht ihnen das Gefühl der Kontrolle. Das impliziert, dass das Interface auf Aktionen richtig reagieren muss. Unvermutete Reaktionen des Systems, ermüdende Abläufe von Dateneinträgen, Schwierigkeiten, richtige Informationen zu finden, und Unvermögen eine gewünschte Aktion auszuführen, lösen Ängste und Unzufriedenheit mit dem System aus.

8. **Reduktion der Kurzzeitgedächtnislast:** Dadurch, dass das menschliche Kurzzeitgedächtnis begrenzt ist, soll die Interaktion mit dem System einfach und das Display übersichtlich sein. Das System soll ermöglichen, dass AnwenderInnen Informationen wiedererkennen, sie also nicht aus dem Kurzzeitgedächtnis abrufen müssen. Durch einen Rechtsklick auf ein Objekt kann etwa angezeigt werden, welche Aktionen darauf ausgeführt werden können (Mandel, 1997, S. 5f).

Die acht goldenen Regeln gelten allgemeinhin für Softwareprodukte. Bei Editoren können weitere Eigenschaften in Bezug auf User Interface Design spezifiziert werden.

[1] Kontrollüberzeugung meint die Überzeugung eines Individuums, dass ein Zusammenhang zwischen Handlungen oder Vorkommnissen und darauf folgenden Ereignissen besteht. Eine interne Kontrollüberzeugung besteht dann, wenn die Aktionen von einem selbst, z.B. durch Drücken einer Taste, ausgelöst werden.

Interface Design unter besonderer Berücksichtigung von Editoren

Über User Interface Design und Gestaltungsprinzipien zu schreiben würde hier die Grenzen sprengen, an dieser Stelle wird ausschließlich User Interface Design speziell bei Editoren betrachtet. Ein Editor stellt ein metaphorisches System dar, das die verwendeten Objekte idealerweise auf natürliche und erwartungskonforme Art und Weise visualisiert. Editoren bieten Methoden an, mit denen Objekte einfach manipuliert werden können (Herczeg, 2006, S. 86). Die Qualität direkter Manipulation kann zu drei Prinzipien zusammengefasst werden (Shneiderman & Plaisant, 2005, S. 234):

- Stetige Visualisierung der Objekte, Werkzeuge und Aktionen durch sinnvolle Metaphern
- Physische Aktionen (mit Mausklicks, Joystickbewegungen, ...) oder Drücken beschrifteter Buttons anstelle von komplexer Syntax
- Schnelle, inkrementelle und reversible Aktionen, deren Auswirkung auf die Objekte sofort sichtbar ist

Der Vorteil der Beachtung dieser Prinzipien liegt darin, dass AnwenderInnen unmittelbar bewerten können, ob die ausgeführten Aktionen zum Ziel führen. Kontraproduktive Aktionen können rückgängig gemacht werden. Fehlermeldungen werden außerdem seltener gebraucht, da Feedback vom Zustand des Objekts sichtbar ist. UserInnen fühlen sich im Umgang mit der Applikation sicher und beruhigt, sofern das Interface begreiflich und Aktionen leicht rückgängig gemacht werden können. Zusätzlich erhalten sie Vertrauen in und Beherrschung über die Applikation, weil sie die Initiatoren der Aktionen sind, die Reaktionen der Objekte im Interface vorhersagen können und dadurch das Gefühl der Kontrolle erlangen (Shneiderman & Plaisant, 2005, S. 234).

Editoren verlangen eine besondere Berücksichtigung im User Interface Design, um auf die inkrementelle[2] Arbeitsweise der AnwenderInnen Rücksicht zu nehmen. Ein Dokument, eine Webpage, oder ein Bild werden nicht in einem linearen Vorgehen erzeugt, es sind viel mehr kleine Schritte, die zum Ziel eines

[2] Eine kontinuierliche Vorgehensweise, welche häufig in kleinen oder sogar kleinsten Schritten vollzogen wird. Der Endzustand steht in der Regel nicht fest.

fertigen Arbeitsstücks führen. Es wird eine Kleinigkeit kreiert, dann verändert, etwas hinzugefügt, ein Teil gelöscht, und am Ende überprüft ob es zufrieden stellt. AnwenderInnen gehen in dieser Tätigkeit auf, wenn die Interaktion und die Verwendung der Werkzeuge dabei so in den Hintergrund rücken, dass der Erfüllung der Aufgabe im Vordergrund steht und die gesamte Aufmerksamkeit geschenkt werden kann. Dabei muss Unmittelbarkeit von Aktionen gegeben sein und der Arbeitsablauf soll dabei möglichst wenig durch plötzlich auftauchende Dialoge, unnötige Mausbewegungen oder Wartezeiten unterbrochen werden. Tidwell beschreibt vier Elemente, die für die meisten Editoren essentiell sind (Tidwell, 2005, S. 243f):

- **WYSIWYG editing:** "What You See Is What You Get" bedeutet, dass ein Dokument im Editorfenster entsprechend aussieht wie das Endprodukt. Demzufolge ist die direkte Manipulation im Editorfenster ohne Befehlseingaben oder Zwischenrendern möglich.
- **Direct Manipulation:** UserInnen können Objekte nehmen, drücken, ziehen und loslassen, in der Größe verändern, sammeln, anmalen, öffnen, schließen, etc. Diese Aktionen sollen unmittelbar und ohne unnötige Wartezeiten ausgeführt werden.
- **Modes**: Ein Interface Mode bestimmt, welches Ergebnis eine doppeldeutige Geste bewirkt. Es macht beispielsweise einen großen Unterschied, ob nur eine Buchstabentaste in einem Texteditor gedrückt wird oder gleichzeitig mit der Steuerungs-Taste. Dasselbe gilt auch für manche Mausgesten: Drag-and-Drop ist normalerweise so definiert, dass ein Objekt verschoben wird. Wenn allerdings ein Linienzeichenwerkzeug aktiviert ist, wird mit dieser Mausgeste eine Linie gezeichnet.
- **Selection:** Oberflächlich betrachtet sind Selektionsoperationen einfach zu designen, da sich mittlerweile einige Konventionen durchgesetzt haben. Ein einzelner Klick in einen Text setzt einen Textcursor, ein Doppelklick selektiert das ganze Wort, ein Dreifachklick wählt eine ganze Zeile oder den ganzen Paragraphen. In schwierigeren Fällen, etwa wenn mehrere Objekte ausgewählt und an der linken Kante ausgerichtet werden, stellt sich das Selektionsproblem als komplizierter heraus.

Im Detail werden einige Techniken, die in die Struktur der vorhin beschriebenen Muster passen, genauer erklärt: Durch *Edit-in-Place* können UserInnen Text an

der Stelle editieren, an welcher sich das Textobjekt befindet. Dabei wird eine Editor direkt über dem Text platziert anstatt den Text in einem separaten Panel oder einem Dialog zu ändern.

Smart Selection bedeutet, dass die Software automatisch eine kohärente Gruppe erkennt und diese auswählt. Nicht alle BenutzerInnen können präzise Mausbewegungen durchführen, in manchen Fällen kann es notwendig sein, etwa wenn nur ein unpräzises Trackpad zur Verfügung steht.

Bei *Composite Selection* werden unterschiedliche Mausgesten, etwa Gesten in verschiedenen Bereichen eines Interfaces, anders gedeutet, je nachdem ob der Zusammenschluß von Objekten oder die darin enthaltenen Objekte ausgewählt werden sollen.

Normalerweise wollen AnwenderInnen eine Operation (etwa das Erzeugen eines Objekts) genau einmal durchführen, genannt *One-off Mode*, um dann sofort zur nächsten Aktion (etwa der Manipulation des Objekts) überzugehen. Aus diesem Grund lässt man manche Operationen nur einmal durchführen.

In einem *Spring-loaded Mode* kann einer Geste eine zusätzliche Bedeutung verliehen werden, wenn begleitend eine Tastatur- oder Maustaste gedrückt wird. Der Modus wird verlassen sobald die Taste losgelassen wird, danach wird wieder zum ursprünglichen Modus gewechselt.

Constrained resize bedeutet, dass unterschiedliche Möglichkeiten zur Verfügung stehen sollen, wenn ein Objekt in der Größe verändert wird, beispielsweise das Erhalten der Seitenverhältnisse.

Magnetism ist eine Möglichkeit, BenutzerInnen unter die Arme zu greifen, wenn sie Objekte im Verhältnis zu anderen anordnen wollen. Das Anordnen und automatische "Andocken" von Toolbars wäre ein solcher Anwendungsfall. Eine direkt mit Magnetismus in Zusammenhang stehende Anwendung sind *Guides*, d.h. horizontale und vertikale Linien, an denen Objekte angeordnet werden können (Tidwell, 2005, S. 249f).

Zusammenfassung

User Interface Design soll als Disziplin im Zusammenhang anderer gesehen werden, da es sich beim Design einer Benutzeroberfläche nicht alleinig um die

Anordnung von Buttons handelt. Interface Design passiert immer im größeren Kontext des Erlebens eines Produkts, dieser Zusammenhang muss beachtet und im Design berücksichtigt werden. Vor allem die Interessen der AnwenderInnen müssen in das Design miteinbezogen werden, um das Arbeiten mit dem Produkt optimal zu gestalten. Allgemein dienen in die acht goldenen Regeln des User Interface Designs als Anhaltspunkte. Desweiteren wurde speziell auf das Interfacedesign bei Editoren eingegangen, da dies für das Projekt relevant erscheint. Das Projekt wird im folgenden Kapitel mit einem Rückblick und nachbetrachtender Reflexion beschrieben.

5 (Re)Design des Infinica Document Designers

Das Projekt Infinica begann für unser Designteam, bestehend aus den Betreuern Peter Purgathofer, Wilfried Reinthaler und mir als Diplomandin, Ende Juni 2007. Es wurde vier Monate vorher begonnen, eine Softwareapplikation zu entwickeln, die das Verwalten und Akquirieren von Kunden-, Geschäfts- und anderen Daten in einem Unternehmen ermöglicht. Die Applikation entstand im Zuge eines Forschungsprojekts der Firma Qualysoft in Zusammenarbeit mit dem Institut für Verteilte Systeme an der TU Wien (Information & Software Engineering Group, 2007). Die Serverlösung des Infinica Packages bestand schon seit längerem, als Neuerung wurde ein Document Designer entwickelt, mit dem XSLT und XForms Dokumente erstellt werden können. Da der Document Designer viele Funktionen erfüllt und folglich ein komplexes Interface zu designen war, wurden wir zu dem Projekt hinzugezogen.

Das erste Treffen

Das erste Treffen mit den Entwicklern bei Qualysoft fand Ende Juni 2007 statt. Das Projekt wurde uns erstmals beim Treffen vorgestellt und der Infinica Designer als Teil des gesamten Softwarepackets präsentiert. Neben dem Infinica Document Designer gehören Infinica Server, Infinica Process Designer und Infinica Administrator zum Softwarepacket, das zum prozessorientierten Verwalten der Dokumentgenerierung dienen soll. Beim Treffen fand ein gegenseitiges erstes Kennenlernen statt: Wir erhielten einen ersten Eindruck von den Eckpunkten des Projekts, der Applikation, dem Programmierteam, und dem Kontext, in dem das Projekt stattfand.

Die Ziele bei Qualysoft sehen laut Spezifikation folgendermaßen aus:

> *Ziel des Projektes Infinica Interact ist die Erstellung eines dokumentenorientierten Formularmanagement Systems, welches mit Enterprise Application Integration (EAI) und Workflowsystemen interagiert. Als Besonderheit werden dabei die Formulare im Zusammenspiel mit einer dynamischen Dokumenterzeugung betrachtet. Ebenso ist eine Integration in bestehende Portalsysteme über die standardisierten Portletmechanismen vorgesehen. Die speziellen technischen Ziele und Besonderheiten sind dabei:*
>
> - *Vollständiges Formular- und Dokumentenerstellungssystem*
> - *Formular- und Dokumentendesign innerhalb eines einzigen Designers*
> - *Konformität zum XForms Standard*
> - *Konformität zum XSL-FO Standard*
> - *Einbindung in Enterprise Portal Systeme mittels JSR168 Standard*
> - *Enge Anbindung an EAI Systeme mittels BPEL und BPEL4People Standard*
> - *Einbindung von Sicherheitsaspekten zu den erwähnten Standards*
> - *Skalierbarkeit und Performance*
> - *Benutzerfreundlichkeit*

Was uns zu diesem Zeitpunkt ins Auge stach war eine technisch einwandfreie Spezifikation mit einer klaren Zielsetzung, aber ohne klar definierte Zielgruppe. Es sollte ein Document Designer entwickelt werden, mit welchem es möglich ist, XSLT und XForms Dokumente und Formulare zu erstellen. Beim Treffen erfuhren wir, dass kein Editor existiere, in dem beide Standards bearbeitet werden können, dass auf dem Markt befindliche Editoren in keinster Weise zufriedenstellend funktionieren und für unerfahrene AnwenderInnen zu komplex in der Erstellung von Layouts seien. Ein weiter definiertes Ziel bei Qualysoft, das vor allem uns betraf, war die Benutzerfreundlichkeit des Document Designers.

Technische Beschreibung

Für das weitere Verständnis werden an dieser Stelle technische und funktionelle Hintergrundinformationen zum Infinica Designer erläutert: Der Infinica Designer

soll als Teil einer vollständigen Datenbanklösung für Firmen das Editieren von XSLT- (= Extensible Stylesheet Language Transformation) und XForms-Dokumenten und Formularen ermöglichen. XSLT ist im Zusammenhang mit dem Infinica Designer zur Transformation von XML-Dokumenten in andere XML-Formate und durch XSL-FO (W3schools, 2007) in "lesbare" Formate wie *.pdf, *.ps, *.doc, *.html zuständig. Die Transformation in andere Formate funktioniert wie mit einem Stylesheet (W3C, 2007b). XForms wird zur Erstellung von interaktiven Webformularen angewendet, in die Daten eingegeben, und automatisch an den Infinica Server weitergeleitet werden. Technisch handelt es sich hierbei um eine XML-Applikation, welche die Präsentation vom Inhalt trennt (W3C, 2007a).

Bei der Erstellung eines Formulars stehen die Felder aus der Datenbank als XML-Datei zur Verfügung. Ein Feld heißt z.B. "*$auszug/kunde/nachname*". Dieses Feld kann durch Drag-and-Drop oder Copy-and-Paste in das Formular übertragen werden. Es können durch XPath auch Abfragen auf den Datenbankfeldern ausgeführt werden, z.B. können alle Kunden einer Firma, deren Rechnung mehr als € 9.99 beträgt, ausgegeben werden. Ein Dokument kann nach seiner Fertigstellung für beliebig viele Datensätze seriell erzeugt werden (etwa wie Telefonrechnungen, die monatlich vom Handyanbieter an alle KundInnen verschickt werden). Diese serielle Erzeugung passiert am Infinica Server, z.B als PS-Dateien. Für weitere Informationen zum Infinica Packet ist die Webadresse des Produkts zu empfehlen (Qualysoft, 2008).

Da unterschiedliche Datensätze ungleich groß sind (etwa in der Buchstabenlänge einer Adresse), und ein Dokument für viele Datensätze seriell erzeugt wird, kann das Aussehen eines Dokuments nicht vorherbestimmt werden. Es kann passieren, dass etwa bei einer ungewöhnlich langen Adresse eines Kunden eine weitere Zeile begonnen wird, und sich das Aussehen des Dokuments für diesen einen Datensatz verändert. Außerdem werden Datensätze oft durch eine Bedingung, über welche iteriert wird, ausgewählt (z.B.: nur Vertragspartner seit 2001), somit kann nicht vorweggenommen werden, wie lange ein Datensatz sein wird. Im Falle des Infinica Document Designers bedeutet dies, dass es kein WYSIWYG gibt, und folglich die Anzeige im Editorfenster nicht der endgültigen Länge und dem Aussehen des Dokuments gleicht. Als Vorschau dient ein PDF-Viewer, diese benötigt ein paar Sekunden zur Generierung eines PDFs, abhängig von der Größe des Dokuments. Dies

entspricht infolgedessen keiner richtigen Vorschau, da eine solche in Echtzeit generiert werden muss (vgl. Kap. 4, S. 47). Aus der Tatsache, dass im Designer kein WYSIWYG möglich ist, ergaben sich im Laufe des Projekts sehr komplexe und schwierige aber nichtsdestoweniger spannende und interessante Problemstellungen.

Projektkontext bei Qualysoft

Es gibt beim Projekt Infinica neben dem Projektleiter zwei Vollzeit angestellte Programmierer und fünf weitere Diplomanden, die das Projekt Infinica auf systemischer Ebene programmieren, wobei einer halbtags angestellt ist. Der Projektleiter, der für die Aufgabenverteilung, den Projektplan und Entscheidungen zuständig ist, ließ uns bei den Treffen freie Hand und Gestaltungsmöglichkeiten. Der Entwicklungsprozess und die Arbeitsweise des Teams war für uns nicht ganz transparent, was auch daran lag, dass wir am wöchentlich Jour Fixe nicht teilnahmen.

Die ersten (Re)Designschritte

Die Entwicklung des Infinica Document Designers begann schon vier Monate vor unserer Mitarbeit. Das Projekt war zu dieser Zeit schon sehr weit fortgeschritten (vgl. Abbildung 5.1, S. 55). Der Grundaufbau des Interfaces war bereits feststehend, überdies war dem Document Designer durch Eclipse (Eclipse, 2008) als grundlegende Systemumgebung eine fixe Struktur immanent.
In Abbildung 5.1 ist ersichtlich, dass der Infinica Designer viele Fenster mit Tabs für weitere Views hat. Die XSD-Schema View, die die XML-Felder aus der Datenbank anzeigt, ist in diesem Screenshot nicht zu sehen. Somit war einer unserer ersten Schritte im Designprozess, das Aufräumen und sie Simplifizierung des Infinica Designers. Eine Simplifizierung bzw. dass Komplexitäten, die nicht ständig benötigt werden, nicht sichtbar sind, soll den Fokus auf relevante Komponenten lenken. Der Hauptaugenmerk sollte auf dem Editorfenster liegen, in dem sich direkt manipulierbare Objekte befinden. Eine der ersten Ideen war außerdem, die Linien zu reduzieren und bei einem Mouseover den betreffenden Block hervorzuheben, um die Sichtbarkeit zu erhöhen. Weiters wollten wir die Werkzeuge und Eigenschaften (konsistent mit anderen Windowsapplikationen) in

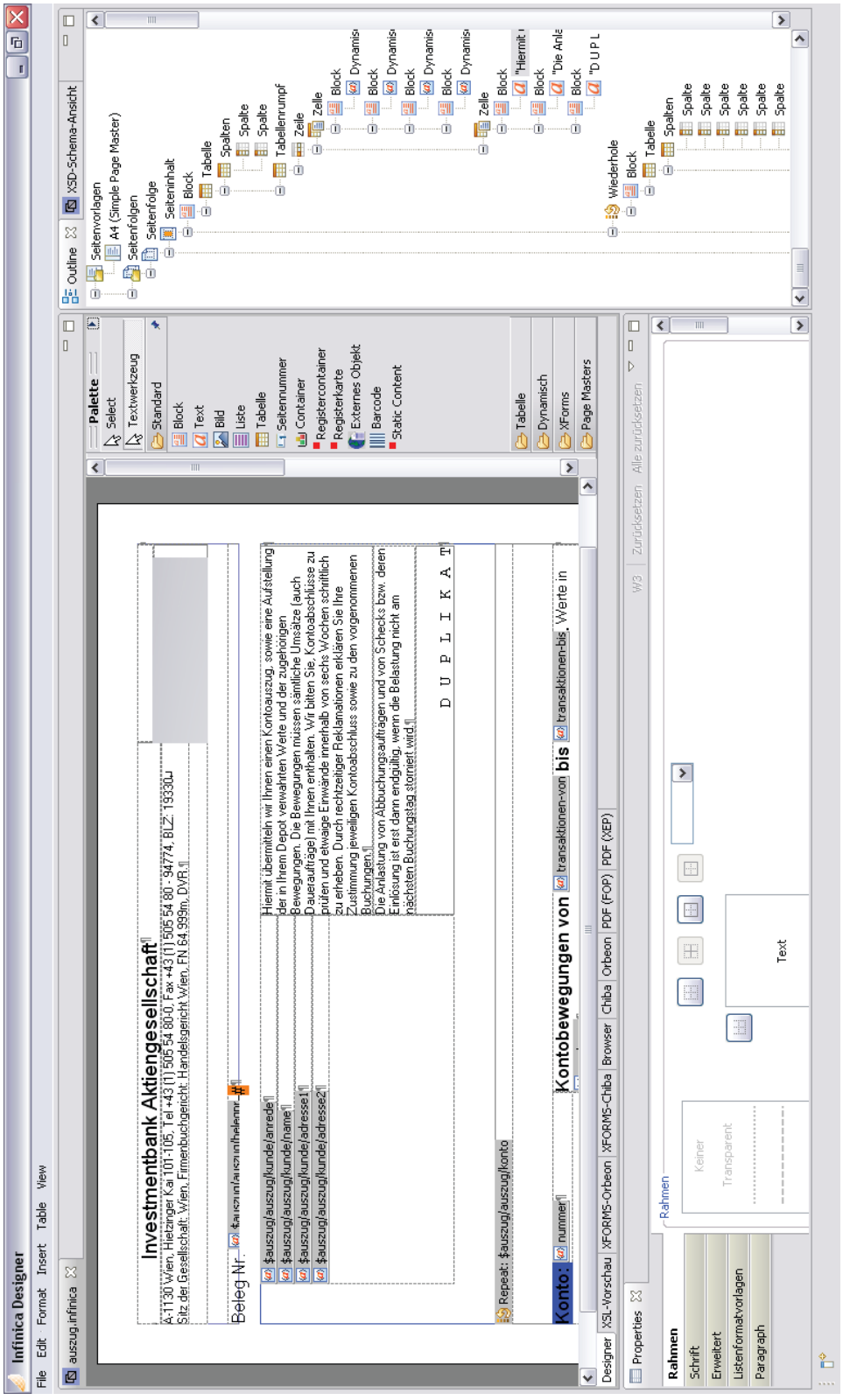

Abbildung 5.1: Der Infinica Document Designer zu Beginn unserer Mitarbeit. Die Applikation wurde zu diesem Zeitpunkt bereits vier Monate entwickelt.

horizontalen Toolbars anbieten. Unser Vorschlag war, die Werkzeug- bzw. Property-View zu entfernen, damit das Interface strukturierter wird und Klarheit kommuniziert.

In der ersten Orientierungs- und Explorationsphase analysierte ich andere Softwareapplikationen, was mir einen Einblick verschaffte, wie Anforderungen am Editordesign bei anderen Produkten gelöst werden. (vgl. Kap. 3, S. 31). Überdies lieh ich mir vom Institut für die Masterarbeit ein MacBook, um andere User Interface und Interaktionskonzepte kennenzulernen als sie unter Windows zu finden sind. Das Konzept der direkten Manipulation, eine essentielle Eigenschaft des Ediordesigns, ist unter Mac OS X besser verwirklicht.

Während der Designphase, die über Simplifizierungen hinausging, hatten wir mehr und mehr Schwierigkeiten damit, dass wir keine konkrete Vorstellung von den wesentlichen Eigenschaften und Bedürfnissen der EndanwenderInnen im Kontext des Document Designers hatten. Aus diesem Grund stellten sich viele Fragen: Wer sind die AnwenderInnen? Wie würden sie die Software benutzen? Was ist der genaue Workflow? Was sind deren Ziele? In welchem Umfeld und unter welchen Bedingungen wird das geschehen? Die Beantwortung dieser Fragen war essentiell, um das Design weiterzuentwickeln und erstmals einen Eindruck davon zu bekommen, für wen wir designten. Denn ohne eine Vorstellung darüber, welche Qualifikationen, Bedürfnisse, Erwartungen und Wünsche die EndanwenderInnen haben, war es sinnlos weiterzuarbeiten.

Die EndanwenderInnen

Um mehr über die Menschen zu erfahren, die am Ende des ganzen Entwicklungsprozesses den Infinica Document Designer anwenden werden, führten wir Interviews (vgl. Kap. 3, S. 29) in einem großen Telekommunikationsunternehmen in Wien durch, die Kunden der Qualysoft sind und eng mit ihr zusammenarbeiten.

Für die qualitativen Interviews hatten wir uns im Vorhinein einige Punkte überlegt, über die wir mehr in Erfahrung bringen wollten, etwa welche Software zu diesem Zeitpunkt für die Aufgaben verwendet wurde, was die MitarbeiterInnen grundsätzlich an Software schätzten, den Workflow und die Arbeitsweise.

Außerdem wollten wir mehr über die generelle Verantwortlichkeitsverteilung bzw. die Zuständigkeit für die Überprüfung und Freigabe zur seriellen Erzeugung der Formulare wissen. Der Hintergrund bzw. die Eigenschaften der gewünschten AnwenderInnen war zudem ein wichtiger Aspekt: Sind es jüngere, ältere, computerversierte AnwenderInnen? Wird sich an der Aufgabenverteilung in Zukunft etwas ändern?

Interviews

Durch den ersten uns vermittelten Kontakt kam es zu einem Interview mit zwei technischen Mitarbeitern der IT-Abteilung des Unternehmens. Im Interview wurden einige spannende Aspekte im Zusammenhang mit der Erstellung von XSLT- und XForms-Dokumenten ersichtlich: Die IT-Abteilung erhielt Aufgaben in Form von Microsoft Word-Dokumenten aus der Prozessabteilung, die durch XSLT und XForms nachgebaut werden, d.h., technisch weniger Qualifizierte können keine Layoutierungsaufgaben und textuelle Änderungen mit derzeit erhältlichen XSLT-Editoren durchführen. Diese Aufgaben waren den Mitarbeitern der IT-Abteilung nicht nur lästig, sondern entsprachen auch nicht deren Kompetenzen, daher sähen sie Aufgaben wie Layoutierung und textuelle Gestaltung gerne bei den dafür verantwortlichen Abteilungen wie Kommunikation und Marketing. Damit einher ging aber die Angst, dass andere im Workflow Beteiligte die Logik, die von technischen MitarbeiterInnen programmiert wurde, versehentlich veränderten oder im schlimmsten Fall löschten. Aus diesem Grund artikulierten sie eindeutig den Wunsch nach einem Rollenkonzept mit bestimmter Rechtevergabe (einschließlich ihrem Recht, alles verändern zu können). Ein weiterer Grund für den Wunsch, die Dokumenterstellung auch in andere Abteilungen zu verlegen, waren ressourcentechnischer Natur. Denn die Erstellung von Formularen und Dokumenten wurde in der IT- Abteilung teilweise als lästiger Zeit- und Ressourcenfresser gesehen, vor allem wenn es simpler textueller und layoutierungstechnischer Veränderungen bedurfte. Am Ende eines Dokumenterstellungsprozess stand ein Team oder eine Abteilung, das die Dokumente auf seine Richtigkeit überprüfte und danach zur seriellen Erzeugung freigab.

Nachdem viele Abteilungen und Interessensgruppen an der Erstellung dieser Formulare beteiligt waren, stellten sich gänzlich unvorhergesehene und interessante Anforderungen an den Infinica Designer. Denn dieser sollte im besten Fall diese Arbeitsabläufe unterstützen.

Wir gingen noch ein zweites Mal zum Kundenunternehmen von Qualysoft, um auch mit den MitarbeiterInnen der Prozess-, Marketing- und Kommunikationsabteilung zu sprechen, und so andere Sichtweisen zu erfahren. Während des Gesprächs haben sich viele überraschende Aspekte aufgetan: Der Wunsch nach Kontrolle über die Dokumente wurde nun von dieser Seite artikuliert. Überdies stellte sich im Interview heraus, dass MitarbeiterInnen dieser Abteilungen die Kontrolle über das Corporate Design (Layout) überhaben wollten, weil die technischen MitarbeiterInnen darauf weniger Wert legten. Außerdem äußerte eine Mitarbeiterin den Wunsch, ohne Hilfe der IT-Abteilung befüllbare Formulare (Daten werden in ein Formular eingegeben und an die Datenbank gesendet) erstellen zu können.

Nachdem bei der Dokumenterstellung viele Interessensgruppen Entscheidungen treffen, hinter denen ein komplexer interner Workflow steht, soll der Infinica Designer diese Entscheidungsprozesse unterstützen. Die Rechtsabteilung möchte etwa gesetzlich die Inhalte absichern, wohingegen die MitarbeiterInnen der Kommunikationsabteilung andere Prioritäten haben. Die MitarbeiterInnen der Marketingabteilung wollen das Layout bestimmen. Es kommt vor, dass die IT-Abteilung erklärt, ein Formular sei technisch nicht realisierbar, und dahingehend das Design auf die Möglichkeiten in XSLT abgestimmt werden muss. Zusammengefasst spielen bei der Dokumenterstellung viele Faktoren mit: Abteilungen und ihre MitarbeiterInnen, Diskussionen um den Inhalt, Entscheidungsprozesse und Verantwortlichkeiten.

Es wurden schon während des Interviews Lösungen angedacht und einige Ideen aufgeworfen, einen transparenten Entscheidungsprozess zu unterstützen und ein gemeinsames Arbeiten an einem Dokument zu erleichtern. Die Versionierung eines Dokuments sollte transparent und klar sein. Die Elemente eines Dokuments sollten mit Zusatzinformationen, TODOs und Kommentaren annotiert werden können. Ein Dokument selbst sollte auch mit Metainformationen bzw. Tags versehen werden können, um eine Suche nach Schlagworten zu ermöglichen,

beispielsweise nach den Dokumenten, die ein Kündigungsschreiben sind oder in denen eine Weihnachtsaktion beworben wird.

Die Idee von Bausteinen bzw. Subtemplates erschien allen als essentiell, das Konzept soll durch ein Beispiel erklärt werden: Eine Adresse wird als Subtemplate einmal erstellt, und kann dann in allen betreffenden Dokumenten referenziert werden. So muss die Adresse nicht jedes mal neu in ein Dokument eingegeben werden und eine Adressänderung muss nur im Subtemplate durchgeführt werden. Um bestehende Templates zu aktualisieren, wie beispielsweise bei einer Aktionsänderung, sollte eine Suche- und Ersetzen-Funktion über alle Templates hinweg zur Verfügung stehen, ähnlich Programmierumgebungen. Kurz tauchte die Idee auf, Bausteine und Dokumente hierarchisch und mit Rechten zu verwalten, um dem Kontrollbedürfnis über die Dokumente nachzukommen. Bald aber stellte es sich heraus, dass dies keine einfache Lösung war: Wer vergab die Rechte, wie sollte alles verwaltet werden? Ein Mitarbeiter meinte nach dem Verwerfen der Idee:

> *"Ja, wahrscheinlich ist das besser, es gibt eh schon genügend Elefanten"*

Generell lagen die Wünsche der EndanwenderInnen noch nicht im Fokus des Designs des Programmierteams. Durch die Interviews bekamen wir eine völlig neue Sichtweise darauf, was der Infinica Designer unterstützen soll und in welchem Kontext er verwendet wird. Welches Potential er in sich birgt und wie er die Abläufe in einem Unternehmen erleichtern könnte. Die Erkenntnisse durch die Interviews waren richtungsweisend für das Design des Infinica Document Designers. Grundsätzlich waren die Design- und Explorationsphase keine getrennten Prozesse, die unabhängig voneinander stattfanden, sondern sie waren eng miteinander verbunden und bedingten sich gegenseitig.

Das (Re)Design

Zur Ideenfindung trafen wir uns regelmäßig und kommunizierten Ideen durch Sketches (vgl. Kap. 3, S. 32) auf Papier. Ich hatte verschiedenste Ideen, Screenshots von anderen Programmen auf Papier an die Wand neben meinem Schreibtisch geheftet, kommentiert, und in Verbindung miteinander gebracht.

Wenn ein Konzept fertig durchdacht war, fertigte ich Screenshots mit Adobe Photoshop ® und Illustrator ® an. Dabei ergaben sich meist noch mehr Details, die Fragen und Probleme aufwarfen und die es zu beantworten galt. Bis es zu einem fertigen Designvorschlag kam, wurden meist mehrere Versionen angefertigt und weitere Überlegungen angestellt. Wir trafen uns regelmäßig, um an bestimmten Problemstellungen zu arbeiten. Dazwischen ging ich oftmals zu Qualysoft mit dem Ziel, ihre Arbeitsabläufe besser zu verstehen und mich über den neuesten Stand des Document Designers zu informieren. Ich brachte weiters in Erfahrung, dass das Wiener Telekommunikationsunternehmen die XSLT- und XForms-Dokumenterstellung an das Entwicklerteam bei Qualysoft outsourcte, wenn die Ressourcen knapp wurden. Somit waren sie Power User und Programmierer der Software zugleich. Power UserInnen haben andere Anforderungen an einen XSLT- und XForms-Designer als MitarbeiterInnen, die das Layout eines Dokuments gestalten.

Die erste Präsentation

Am 24. August präsentierten wir unsere Designvorschläge bei Qualysoft vor dem gesamten Team. Die Präsentation wurde als Arbeitssitzung abgehalten, die Redesignvorschläge wurden diskutiert. Grundsätzlich stand wenig Zeit zur Verfügung, unsere Ideen und Lösungsvorschläge zum Redesign auszuarbeiten, so konnten wir leider nicht alle Möglichkeiten und das ganze Potential der Software ausschöpfen. Wir versuchten, ein paar essentielle Redesignvorschläge zu machen, die das Produkt verständlicher und benutzerfreundlicher machen (vgl. Abbildung 5.2, S. 61).
Die dynamischen Felder aus der Datenbank werden durch die blaue Umrandung als Objekte sichtbar. Ebenso sind logischen Objekte, etwa eine repeat- oder if- Schleife, durch ein oranges abgerundetes Rechteck erkennbar (vgl. Abbildung 5.3, S. 62) Die Farben der dynamischen und logischen Elemente aus 5.3 sind aus der Windows XP visual Guideline entnommen (Corporation, 2001).
In Abbildung 5.3 sind ferner auch graue Objekte zu sehen, die Subtemplates. Diese waren ein Wunsch der MitarbeiterInnen des Telekommunikationsunternehmens. Das Anliegen wurde in unseren Redesignvorschlägen übernommen, da sie ein Organisieren und Verwalten von Teilstücken eines Dokuments erleichtern, etwa Aktion, die in allen Footern stehen

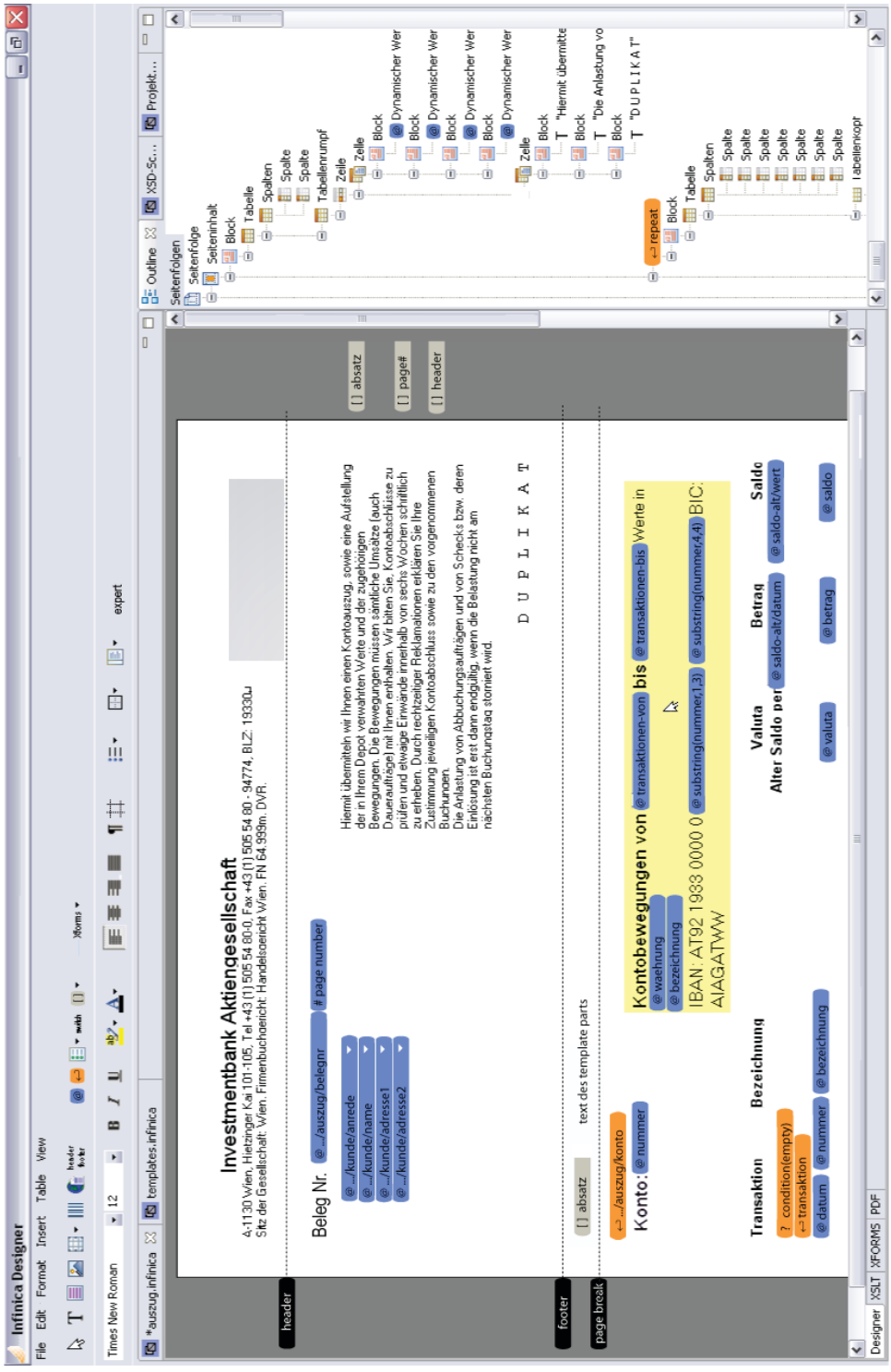

Abbildung 5.2: Gesammelte Designvorschläge für den Infinica Designer. Einer der ersten Schritte war es, die Linien im Editor zu reduzieren und das Interface aufzuräumen. Dies beinhaltete, dass die untere Property View und die Werkzeugpalette in zwei Leisten nach oben wanderten, wie es bei Windows-Applikationen üblich ist.

Abbildung 5.3: Das Aussehen von dynamischen (blau) und logischen (orange) Elementen. Das graue Element stellt ein Subtemplate dar: ein Dokumentteil, der einmal definiert und beliebig oft referenziert werden kann.

soll. Durch die Objekhaftigkeit der abgerundeten Rechtecke und die farbliche Kodierung ist sofort klar, um welche Entitäten es sich handelt. Somit kann auf den ersten Blick das Layout selbst von technischen Aspekten wie Datenbankanbindungen und logischen Aspekten unterschieden werden. Die Implementierung eines Rollenkonzepts, in welchem unterschiedliche Rechte bestimmten Arbeitsmöglichkeiten mit dem Document Designer einräumen, ist dadurch nicht unbedingt notwendig. In den Interviews hatten MitarbeiterInnen Befürchtungen artikuliert, andere am Entwicklungsprozess eines Dokuments Beteiligte könnten Teile des Dokuments durch Unwissenheit löschen oder falsch bearbeiten. Aus einem technischen Blickwinkel wäre ein Rollenkonzept mit vielen Schwierigkeiten verbunden.

Ziel der Designvorschläge war es, dass Power User nicht mehr in die Codeansicht wechseln müssen, um die Logik eines XSLT-/XForms-Dokuments zu programmieren. Ein logisches oder dynamisches Objekt kann durch einen Doppelklick geöffnet werden sodass Code editiert oder ausgewählte Elemente betrachtet werden können. Ferner schlugen wir statt der Verwendung von leeren Tabellen vertikale Grids vor, die magnetisch sind (Snap-to-Grid). Objekte können auf diese Weise einfacher angeordnet werden. Die Grids können überdies ein- und ausgeschalten, und der Detailgrad verändert werden.

Abschließend wurde noch über das Projekt selbst und die Herangehensweise diskutiert. Peter Purgathofer wiederholte noch einmal, dass wir erst sehr spät zum Projekt hinzugezogen wurden, was auch für Erstaunen sorgte.

> *"Wir sind erst sehr spät zum Projekt hinzugekommen..."*
> *"Wieso, ab wann arbeiten denn die Designer normalerweise mit?"*
> *"Von Anfang an."*
> *"Aha ..."*

In den meisten IT Firmen ist der Gedanke, vor der Implementierung eine Preproduction-Phase zu durchgehen, neu (vgl. Kap. 2, S. 10). Desweiteren sprach Peter Purgathofer darüber, dass das Team am Weg ist einen XSLT-/XForms-Editor zu entwickeln, weniger einen Document Designer. Wenn es deren Ziel ist, einen Designer zu erstellen, der auch von technisch weniger versierten Bürokräften angewendet werden soll, muss Funktionalität, die XSLT und XForms zur Verfügung stellt, von der technischen Ebene abstrahiert und vereinfacht werden. Der Document Designer muss näher zum mentalen Modell der AnwenderInnen gebracht werden (vgl. Abbildung 2.2, S. 13).

Nach der ersten Präsentation wurde eine Dokumentation mit den Ergebnissen der Präsentation erstellt. Die weiteren Ziele waren, sowohl eine Lösung für den Pagemaster zur Definition der Seitenbereiche, als auch mit dem damit zusammenhängende Problem der Static Contents (Kopf- und Fußzeile, rechter und linker Bereich) zu finden. Ich ging in der Zeit nach der Präsentation oftmals zu Qualysoft um weitere Designprobleme und Fragen der Interaktion ("*Wie soll ein Block bei einem Mouseover bzw. wenn man ihn anklickt aussehen?*") zu klären: Diese Situation war einerseits verständlich für die Programmierer, die zugleich Designvorschläge für ihre Probleme in Erfahrung bringen wollten, um die Arbeit fortsetzen zu können. Andererseits durfte ich keine Auskunft geben, da Vorschläge ausschließlich zu Ende gereift weitergegeben werden sollen. Schließlich sollen keine überhasteten Entscheidungen implementiert werden, die dann wieder revidiert werden und die Programmierarbeit somit überflüssig wäre. Bei der ersten Präsentation schlugen wir vor, das Icondesign einem professionellen Grafikbüro zu überlassen, die Aufgabe wurde von einem Programmierer des Entwicklerteams übernommen, welcher sich auch mit Graphikdesign beschäftigte.

Pagemaster

Ein Pagemaster dient zur

- Bestimmung des Seitenformats, also der Größe der Seitenränder (Header, Footer, rechter und linker Bereich)
- Definition der Reihenfolge der Seitenformate (etwa dass ein Dokument auf der ersten Seite ein anderes Format haben soll wie bei den darauffolgenden)

Dadurch dass der Infinica Designer WYSIWYG nicht unterstützt und absolute Angaben nicht möglich sind, wurden wir vor ein schwieriges Designproblem gestellt. Der Pagemaster trat nach einigen Überlegungen und Lösungsversuchen mehr und mehr als wicked problem (vgl. Kap. 2, S. 15) in Erscheinung. Es waren viele Treffen und Entwürfe nötig, um zu einer zufriedenstellenden Lösung zu kommen. Missverständnisse in der Kommunikation erschwerten die Definition des Problems, da sie erst im Nachhinein aufgeklärt wurden. Die vollständige Spezifikation des Problems war auch nicht möglich, da es sich um ein wicked problem handelte. Der Entwurf musste am Ende leicht abgeändert werden. Dieser Umstand erklärt auch, dass eine Problemstellung genauso wie eine Lösung nicht vollständig sein kann, und Lösung und Problem einander bedingen (vgl. Kap. 2, S. 15).

Am Ende wurden die Static Contents (Header, Footer, Left and Right) als graue Objekte eingebettet in eine strichlierte Linie repräsentiert (vgl. Abbildung 5.4). Ein Seitenformat wurde als größeres graues Objekt zentriert in einer durchgehende Linie eingefügt, und galt ab dem relativen Punkt, wo sie definiert war (kein WYSIWYG und somit keine absoluten Angaben).

Im oberen Dialog in Abbildung 5.5 ist unser Design zum Pagemaster Dialog illustriert, darunter die tatsächliche Implementierung. Auf der linken Seite des Dialogs werden neue Seitenformate erstellt, darunter befindet sich eine Preview um abschätzen zu können, wie groß die Seitenbereiche sind. In der Mitte werden die genauen Einstellungen vorgenommen. Rechts können Folgen von Seitenformaten, also Seitensequenzen (z.B. eine Rechnung, in der die erste Seite anders formatiert ist als die folgenden) angelegt werden. Ein Seitenformat kann auch per Drag-and-Drop in eine Seitensequenz eingefügt werden. Der rechte Bereich der Seitensequenzen muss explizit durch einen Klick geöffnet werden.

Workshops bei Qualysoft

Wir präsentierten Designvorschläge und schickten diese schriftlich dokumentiert, angereichert mit Erklärungen, Sinn und Zweck an Qualysoft. Nichtsdestoweniger waren wir ungeduldig und wollten erfahren, ob unsere Vorschläge angenommen wurden. Waren die Programmierer selbst davon überzeugt? Waren die Vorschläge zu schwierig in der Implementierung umzusetzen?

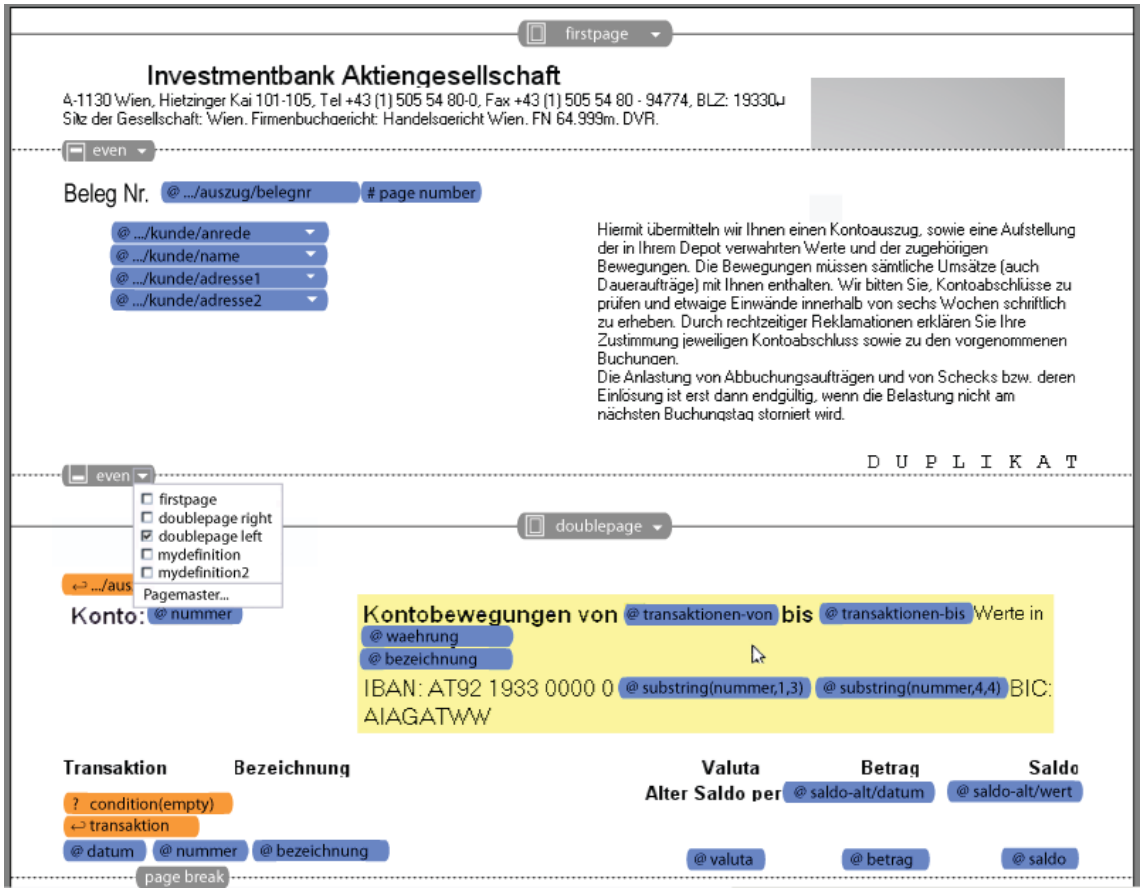

Abbildung 5.4: Die Static Contents sind (Header, Footer, Left, Right) als graue Objekte visualisiert. Durch einen Klick öffnet sich ein Pop-up Menü, um dem Static Content ein oder mehrere Seitenformate zuzuordnen.

Ein Hindernis in der Kommunikation war sicherlich, dass wir als Designteam nicht anwesend waren, beziehungsweise nur dann, wenn es etwas Spezifisches zu besprechen galt. Somit gab es unsere Designvorschläge, aber keinen regelmäßig physisch präsenten Ansprechpartner. Für die Kommunikation zwischen dem Designteam und den Entwicklern wäre es von Vorteil gewesen, einen regelmäßigen Jour Fixe abzuhalten. Eine gemeinsame Arbeitsbasis hätte das Projekt besser unterstützt. Als Consultants war es unsere Aufgabe Designvorschläge zu präsentieren und diese zu argumentieren, allerdings war es nicht unsere Aufgabe dafür Sorge zu tragen, dass alle Designvorschläge implementiert werden. Aus diesem Grund schlugen wir drei Workshops vor, in denen grundsätzliche Dinge noch einmal diskutiert wurden, um Raum für die ReDesignvorschläge zu schaffen und wieder präsenter zu sein.

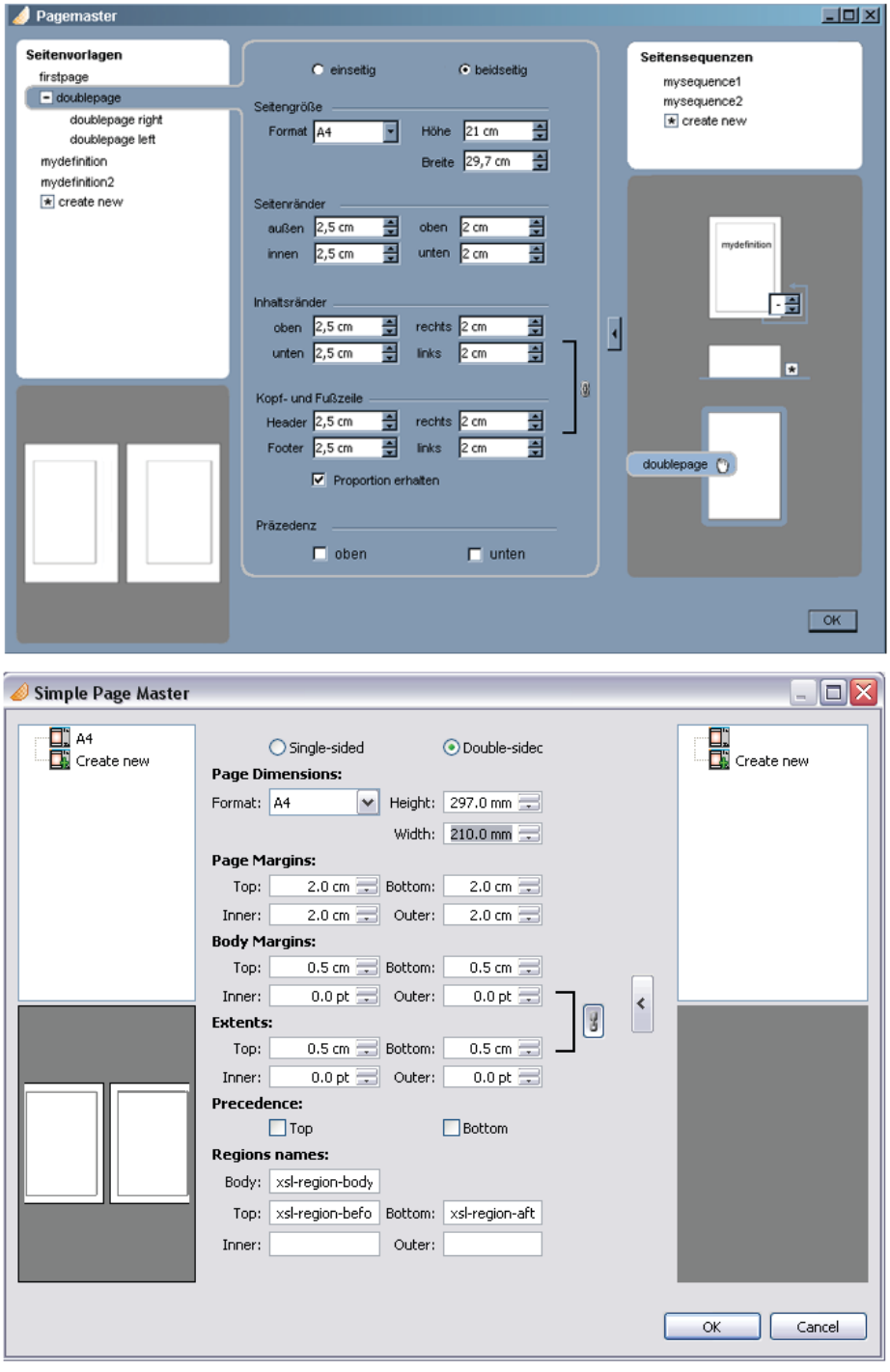

Abbildung 5.5: Complex Pagemaster: Der Dialog oben war unser Designvorschlag, der untere Dialog stellt den Complex Pagemaster, wie er wirklich implementiert wurde, dar.

Die Workshops waren sehr produktiv und wir versuchten gegenseitiges Verständnis zu schaffen. Wenn sich bisweilen auch die unterschiedlichen Modelle und Vorstellungen aneinander rieben, und nicht alles einfach zu kommunizieren war.

> *Designer: "Ich glaube nicht dass dieses Gespräch gerade konstruktiv ist. Es geht hier um gestalterische Grundprinzipien, was will ich erzählen, was will ich denn mitteilen, und nicht dass man es ja eh so machen kann weil es immer schon so gemacht wurde."*
> *Programmierer: "Aber wenn ich etwas editiere, kann das nicht gleich auf der Platte übernommen werden, es muss ein Textfeld darüber gelegt werden!"*

Es gab bei den Workshops auch Zeit, vonseiten des Entwicklerteams neu auftauchende Designfragen zu besprechen, wie etwa das Design des Xpath-Editors.

XPath Editor

Einer der Entwickler stellte bei einem Workshop den XPath Editor, an dem er gerade arbeitete, vor. Wiederum wollten die Entwickler beim Workshop selbst schon eine Lösung von unserer Seite, und teilweise ließen wir uns überreden, uns auf eine Struktur zu einigen.

Der Entwurf zum XPath Editor (vgl. Abbildung 5.6) war unter anderem auch dadurch geprägt, dass wir viel Platz zur Verfügung hatten. Der XPath-Editor soll nur, wenn er benötigt wird, sichtbar sein. Er bleibt solange sichtbar, bis er bewusst wieder geschlossen wird. Nach der Präsentation des Entwurfs stellte sich heraus, dass das Eingabefeld zu klein war, da oft mehrzeilige Eingaben vorkommen. Wir haben auf Papier hastig einen anderen Vorschlag gesketched, der dem Programmierer als Anhaltspunkt dienen sollte.

Der XPath Editor ist ein weiteres Beispiel dafür, dass Missverständnisse in der Kommunikation vorkamen. Wir mussten durch das Kommunikationsmissverständnis mit dem Textfeld den Entwurf anpassen.

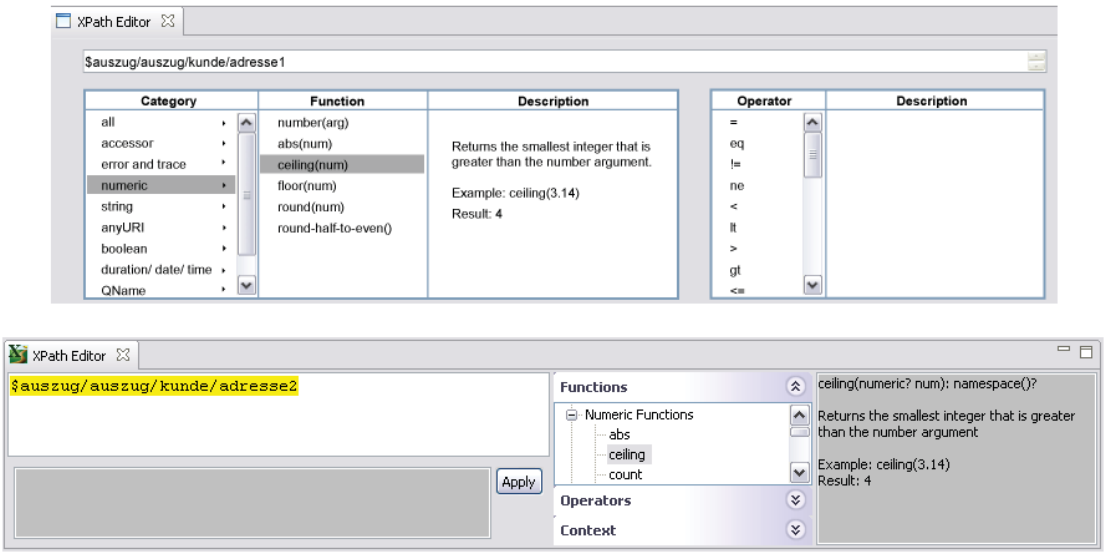

Abbildung 5.6: Entwurf zum XPath Editor im oberen Dialog, die Implementierung unterhalb: Während eines Workshops stellte sich heraus, dass im Entwurf das Eingabefenster viel zu klein war, da Eingaben oft mehrzeilig waren.

Abschlusspräsentation

Am 26.Februar fand die Abschlusspräsentation statt. Wir bekräftigten den von unserer Seite erfolgreichen Verlauf und unsere Zufriedenheit mit dem Ergebnis. Ebenso war es die Ansicht des Projektleiters, dass unsere Designarbeit das Projekt in eine positive Richtung gelenkt hatte. Die Entwicklung des Infinica Packages in Zusammenarbeit mit dem Institut für Gestaltungs- und Wirkungsforschung soll im Rahmen einer weiteren Masterarbeit fortgeführt werden. Auf meine Frage, ob noch User Tests durchgeführt werden, hat der Projektleiter aus einem Missverstehen von Software Tests gesprochen. Wir hatten nur an qualitatives User Testing (vgl. Kap. 3, S. 36) gedacht, wenn von Tests die Rede war.

Nachbetrachtungen zum Projekt

Alle am Projekt Beteiligten haben großartige Arbeit geleistet, denn der Infinica Designer ist nun ein Produkt, das andere am Markt erhältliche XSLT-/XForms-Editoren übertrifft. Wie vorhin erwähnt wurden einige der Designvorschläge umgesetzt (vgl. Abbildung 5.7), manche (noch) nicht. Die

Arbeit am Document Designer ist noch nicht abgeschlossen, er wird künftig weiterentwickelt und verbessert.

Bei den Gesprächen bzw. Interviews mit den MitarbeiterInnen erkannten wir die Bedürfnisse an die Software, die bei der anfänglichen Entwicklung nicht beachtet worden waren, da keine Zielgruppe definiert worden war. Die Entwicklung des Document Designers begann mit dem (technischen) Ziel, einen XForms- als auch einen XSLT-Editor zur Dokumenterstellung zu implementieren. Die Frage nach einer bestimmten Zielgruppe respektive den Bedürfnissen dieser stand dabei im Hintergrund. Als wir später zum Projekt stießen, war es nicht mehr möglich, ein Bewusstsein dafür zu schaffen.

Schwierigkeiten im Projekt stellten jene Situationen dar, in denen die Programmierer bei Qualysoft während der Treffen schnelle Designentscheidungen wollten, ohne Zeit für Überlegungen zu lassen, um sofort mit der Implementierung beginnen zu können. In solchen Momenten ist man als DesignerIn auch versucht, eine Lösung anzubieten. Überhastete Designentscheidungen sind allerdings zu vermeiden, da die erste Lösung meist nicht die beste ist. Designs befinden sich in einem stetigen Fluss von Vorschlägen, Ideen und Entscheidungen, bis sie in einem präsentierbaren Designvorschlag ausgearbeitet werden.

Manche unserer Vorschläge stießen von Anfang an auf Ablehnung, was wir als Consultants auch akzeptieren mussten. Gründe dafür waren eine ressourcenintensive Implementierung oder dass die Entwickler auf manche Features, die wir als überflüssig ansahen, nicht verzichten wollten: Der Vorschlag, die Dokumentrepository als eigenen Tab zu realisieren wurde verworfen, da diese zu aufwändig zu implementieren gewesen wäre. Die sehr lange Liste an Properties wollten die Programmierer nicht an einer anderen Stelle anbieten bzw. gänzlich entfernen, um Power User, die vermutlich mit dieser Liste arbeiten, nicht zu verärgern.

Bei einem Treffen warf ich die Idee von User Tests auf, die für die Entwicklung der Software und als Feedback wesentlich sind. Auch die Programmierer wollten User Tests machen, allerdings waren dafür keine Ressourcen zur Verfügung bzw. wurde die Idee nicht weiter verfolgt. Das Management hätte stärker von der Sinnhaftigkeit qualitativer User Tests überzeugt werden müssen.

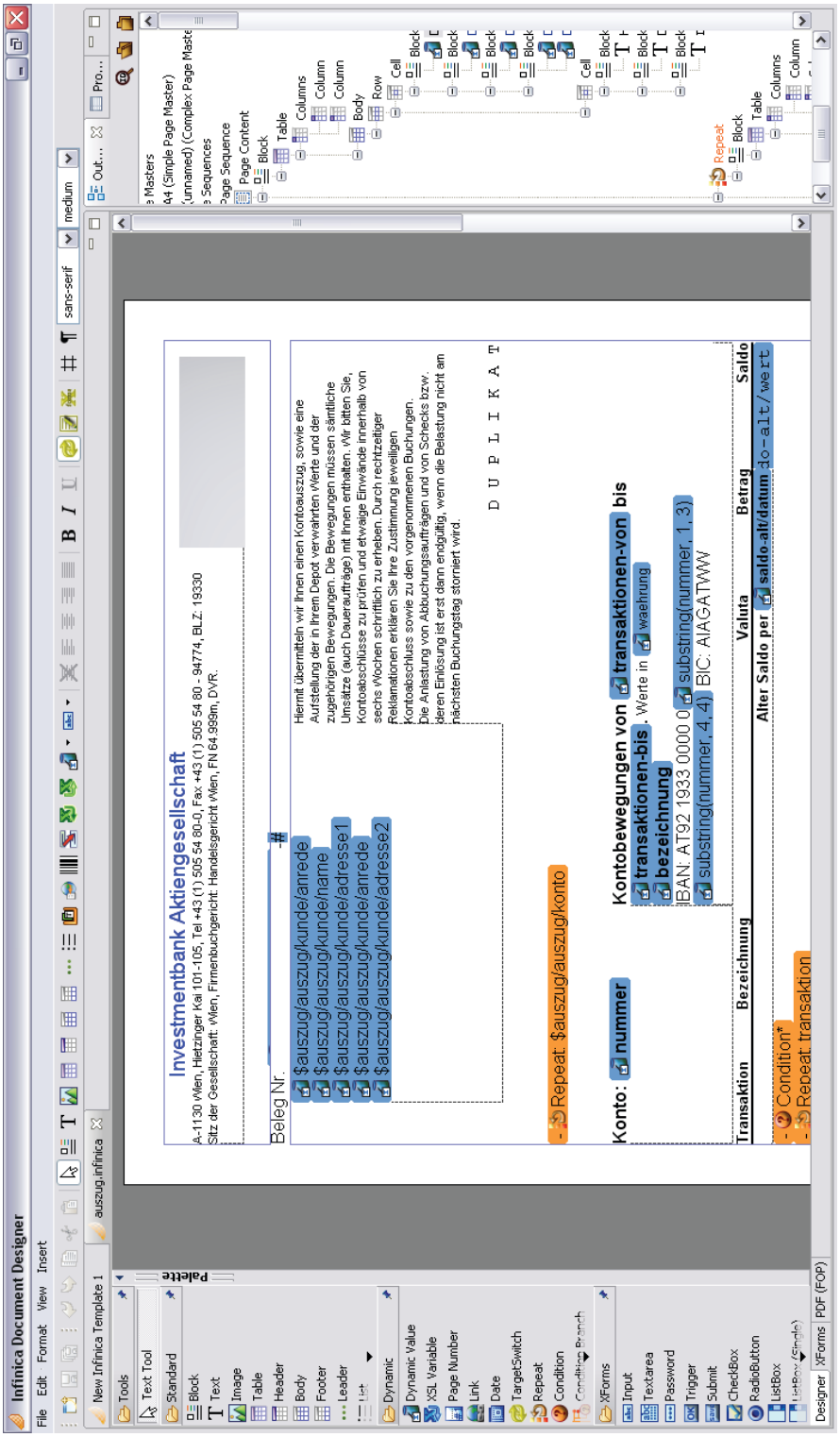

Abbildung 5.7: Der Infinica Document Designers nach dem ersten offiziellen Release. Viele unserer Designvorschläge wurden implementiert. Die Applikation wird künftig weiterentwickelt und verbessert.

Die Entwicklungsumgebung des Document Designers mit Eclipse war durch eine vorgegebene Struktur im Interfaceaufbau charakterisiert. Einerseits erleichterte das viele Entscheidungen und nahm den Entwicklern Arbeit ab, andererseits ist die Freiheit in der Gestaltung eingeschränkt und die Struktur strikt definiert. Etwa passiert kein Highlight von Toolicons bei einem Rollover, die ist ein Bug der schon lange bei Eclipse besteht. Weiters war es nicht möglich unseren Vorschlag, die Werkzeuge in Toolbars zu verpacken effektiv umzusetzen, weil Eclipse alle Icons ohne Abstand nebeneinanderreihte und somit keine Unterscheidung zwischen einzelnen Toolbars, kein Hinzu- und Wegschalten dieser, ermöglichte.

Einsichten und Vorgehensweise für weitere Projekte

Der Infinica Designer ist bereits jetzt ein gutes Produkt, wird aber, wie vorhin erwähnt, weiter entwickelt werden und verbessert. Grundsätzlich existiert kein Projekt, dessen Abwicklung einwandfrei, perfekt und ohne Probleme durchgeführt wird. Fehler zu machen ist natürlich, ohne Irrtümer würde kein Lernprozess stattfinden. Es gibt dennoch Punkte, die in nachträglicher Reflexion anders stattfinden hätten sollen.

Die Zeit, die uns zur Verfügung stand, war durch den Projektkontext bedauerlicherweise begrenzt, dennoch bin ich der Meinung, wir hätten mehr User Research durchführen sollen. Es wäre relevant gewesen, eine ethnographische Studie (vgl. Kap. 3 auf S. 30) durchzuführen, um AnwenderInnen bei der Dokumenterstellung über die Schulter zu schauen und Fragen zu stellen, und so mehr über die Dokumenterstellung zu erfahren. Dies wäre mit Mitarbeitern bei Qualysoft ohne großen organisatorischen Aufwand zu bewerkstelligen gewesen. Sicher konnten wir selbst auch Dokumente erstellen, aber den gesamten Prozess konnten wir wegen fehlender Infrastruktur (keine Datenbankanbindung und Serialisierung am Server) nicht nachvollziehen. Ferner haben AnwenderInnen eine andere Arbeitsweise, die viele interessante Aspekte aufgeworfen, und das Design mitunter verändert hätte.

Ferner stelle ich mir die Frage, ob wir nicht präsenter hätten sein sollen. Wir waren zwar in der Rolle als Consultants im Projekt involviert, aber durch regelmäßige Präsenz hätten wir sicher mehr verändern können. Der Zusammenarbeit mit den Entwicklern wäre ein regelmäßiger Jour Fixe dienlich gewesen.

Ein weiterer Aspekt den ich aus dem Projekt gelernt habe, sind formulierte Designprobleme und neue Anforderungen schon während der Treffen mitzuschreiben oder Gesprächsmitschnitte aufzunehmen, da meist nicht alles erfasst werden konnte. Durch Missverständnisse wurde viel Energie in eine Lösung gesteckt, die dann doch nicht zufriedenstellend war, weil die Anforderung anders kommuniziert wurde. Dies hätte wahrscheinlich auch durch regelmäßige Treffen, in denen Designvorschläge diskutiert werden, verhindert werden können. Ferner hätten wir bei regelmäßigen Treffen die Möglichkeit erhalten, die Relevanz von Design in IT-Projekten hervorzuheben.

Zusammenfassend war das Projekt für mich persönlich sehr wertvoll, weil ich die Möglichkeit bekam, selbstständig Wissen umzusetzen und meinen praktischen Erfahrungsschatz zu erweitern. Es war spannend zu sehen, wie ein Projekt durchgeführt, und ein Produkt entwickelt wird, und meinen Teil dazu beizutragen. Mein praktisches Wissen wurde durch das Projekt erweitert, und diese Erfahrung wird in weiteren Projekten von Nutzen sein.

Zusammenfassung

Das Projekt war durch viele Möglichkeiten, die uns offen standen, und viel Handlungsfreiheit charakterisiert. Wir haben trotz der späten Einstiegsphase in das Projekt versucht, die Bedürfnisse und Wünsche der AnwenderInnen zu berücksichtigen, und in das Produkt einzubringen. Selbstverständlich hing auch vieles von den Entwicklern ab, d.h. inwieweit sie bereit waren, die Implementierung zu verändern bzw. Code wegzuwerfen (vgl. Kap. 6, S. 75).

Die Probleme mit der Rolle und dem Zeitpunkt von Design im Softwareentwicklungsprozess würde sich durch Verwendung agiler Methoden (vgl. Kap. 6, S. 76) einfacher lösen lassen, da diesen Methoden eine adaptive Vorgangsweise immanent ist. Dadurch, dass wir bei Qualysoft nur redesignen konnten, obwohl das Produkt neu entwickelt wurde, waren die Möglichkeiten doch eingeschränkt. Das Produkt wäre wahrscheinlich besser durchdacht und für AnwenderInnen handhabbarer gewesen, hätten wir von Anfang an mitarbeiten dürfen. Einschränkungen bedeuten einerseits weniger Gestaltungsmöglichkeiten, andererseits werden Entscheidungen abgenommen, was den Designprozess erleichtern kann (vgl. Kap 6, S. 74). Trotz einiger

Einschränkungen ist der Infinica Document Designer ein gutes Produkt, und die Entwickler waren für unsere Designvorschläge offen.

Wahrscheinlich war es eine der ersten Male bei Qualysoft, dass in einem Projekt mit einem Designteam gearbeitet wurde. Um Design in einem Unternehmen als festen Bestandteil zu integrieren, bräuchte es mehr Zeit und vor allem ständige Präsenz eines Designteams (vgl. Kap. 6, S. 74). In jedem Fall wurde bei Qualysoft ein Schritt in die richtige Richtung gesetzt, und der Wunsch nach Weiterführung des Projekts mit dem Institut spricht für sich.

6 (Re)Designs in späten Projektphasen

In den vorigen Kapiteln wurde beschrieben, welche Faktoren gutes Design bestimmen und wie ein Designprozess theoretisch funktionieren soll. Dass Design den ganzen Entwicklungsprozesses begleiten soll, auch schon vor der Implementierung, stellt einen Konsens in der wissenschaftlichen Literatur dar. Gute Produkte zeichnet ein Prozess aus, dem kreatives Design der Implementierung vorangeht und das Projekt hindurch begleitet (vgl. Abbildung 2.3, S. 14). Ein Projekt, das dadurch gekennzeichnet ist, dass es keine Preproduction Phase gibt, ist gewissen Beschränkungen unterworfen. Diese Beschränkungen und der daraus entstehende Projektkontext, sollen in diesem Kapitel, einhergehend mit den Möglichkeiten, die Situation zu meistern, diskutiert werden.

Beschränkungen in Designprozessen

In gewisser Weise ist Design ohne Beschränkungen ein schwieriger Prozess. Gedenryd geht so weit zu behaupten, dass Constraints (Beschränkungen) auch einen praktischen Charakter haben. In der traditionellen Ansichtsweise sind Einschränkungen Teil der Problemdefinition bzw. eine Anforderung an das Design. Beschränkungen müssen sich nicht per se negativ auf ein Design auswirken, sie können das Design zwar in gewisser Weise beschneiden, nehmen andererseits aber Entscheidungen ab. Gedenryd fasst Constraints folgendermaßen zusammen (Gedenryd, 1998, S. 71f):

> *Constraints are restrictions on an acceptable solution that are specified in the instructions given to the designer. They are non-optional (but indeed required) and thus beyond the designers control.*

Beschränkungen sind voller Widersprüche: Sie können in manchen Situationen ein Hindernis darstellen oder aber hilfreich sein, optional oder unumstößlich, von Vornherein gegeben oder durch DesignerInnen selbst auferlegt sein. Gedenryd beschreibt drei unterschiedliche Arten von Einschränkungen, je nachdem von welcher Interessengruppe sie geleitet sind. Gesetzliche Einschränkungen sind komplett starr und unabänderlich. Von Klienten auferlegte Beschränkungen sind in gewisser Weise flexibel, weil unter Umständen verhandelbar. Gegebenenfalls sind eine oder mehrere Einschränkungen für den/die DesignerIn inakzeptabel, weil sie zu einer schlechten Lösung führen. Es ist allerdings jederzeit die Möglichkeit gegeben, die Umstände zu verhandeln. Beschränkungen, die von DesignerInnen selbst auferlegt wurden, sind vollends flexibel, da sie sich völlig unter deren Kontrolle befinden und jederzeit geändert oder verworfen werden können (Gedenryd, 1998, S. 73f).

Code Ownership

Eine andere Art der Einschränkung in Softwareprojekten ist durch Code Ownership gegeben. In einer späten Projektphase, in der schon mit der Implementierung begonnen wurde, hängen viele Möglichkeiten von den ProgrammiererInnen ab und inwieweit sich diese bereit erklären, Code zu ändern bzw. wegzuwerfen und neu zu schreiben. Es ist ein psychologisches Phänomen, dass wir beginnen uns mit Dingen, die wir kreieren, zu identifizieren bzw. sie als unseren Besitz zu betrachten. Der Begriff des psychologischen Besitztums gilt sowohl für physische als auch für nichtphysische Objekte ((Pierce, Kostova & Kirks, 2001), zitiert nach (Wang et al., 2006)). Einerseits können in diesem Zusammenhang zwei positive Phänomene beobachtet werden: Menschen fühlen sich vermehrt für etwas verantwortlich, und auch der Wert des Objekts wird als gesteigert wahrgenommen, wenn sie sich als BesitzerInnen betrachten. Andererseits hat Besitztum über eine Sache auch schlechte Eigenschaften, die die Kooperation erschweren. MitarbeiterInnen können persönlichen Besitzverlust, Frustration und Stress empfinden, wenn sie beobachten, dass ihr "Besitz" einer Änderung unterworfen wird. Diese Effekte liegen im Kontrollverlust über das Objekt, das einst ihr eigenstes war, begründet ((Bartunek, 1993), zitiert nach (Wang et al., 2006)). Diese psychologischen Vorgänge haben selbstverständlich Auswirkungen auf die Realisierung von Designvorschlägen bzw. Kritik an bestehenden Lösungen. ProgrammiererInnen

(und auch DesignerInnen) identifizieren sich so stark mit ihrem geschaffenen Artefakt, dass sie Kritik daran persönlich nehmen.

> *After Code is written, it is very difficult to to throw it out. Like writers in love with their prose, programmers tend to have emotional attachments to their algorithms. Altering programs in midstride upsets the development process and wounds the code, too. It's hard on the manager to discard code because she is the one who paid dearly for it, and she knows she will have to spend even more to replace it.* (Cooper, 2004, S. 53)

Code Ownership kann somit, abhängig von den EntwicklerInnen, der verwendeten Methoden und der Projektleitung, zu einer weiteren Einschränkung in einem Projekt werden.

Agile Softwareentwicklung

Die agile Softwarebewegung wurde aus der Softwarekrise geboren (vgl. Kap. 2, S. 10). Wie beschrieben, entstand die Softwarekrise durch komplexer und größer werdende Programme, die mit großen Entwicklerteams scheiterten.

> *The immeasurability and intangibility of software conspires to make it nearly impossible to estimate its size and assess its state of completion. Add in the programmer's joy in her craft, and you can see that software development always grows in scope and time and never shrinks. We will always be surprised during its construction, unless we can accurately establish milestones and reliable measure our progress against them.* (Cooper, 2004, S. 53)

Der Versuch, sich an anderen Ingeneursdisziplinen zu orientieren, scheiterte auch, da diese Vorgehensweisen nicht ohne weiteres auf das Software Engineering übertragen werden können (Dogs & Klimmer, 2005, S. 19). Der permanente Bedarf zur Anpassung in Softwareprojekten ist in den agilen Methoden, die eine adaptive Vorgangsweise erlauben, berücksichtigt. Im agilen Manifest werden vier Werte formuliert, die einen Ausweg aus der Softwarekrise anbieten sollen (Dogs & Klimmer, 2005, S. 32):

1. *Individuen und Interaktionen* sind wichtiger als Prozesse und Werkzeuge.
2. *Funktionierende Software* ist wichtiger als umfassende Dokumentation.
3. Die *Zusammenarbeit mit KundInnen* ist wichtiger als Vertragsverhandlungen.
4. *Sich auf unbekannte Änderungen einzustellen*, ist wichtiger als einem Plan zu folgen.

Die bekannteste der agilen Methodiken ist das populäre eXtreme Programming, das die EntwicklerInnen und KundInnen stark in den Vordergrund stellt und konkrete Praktiken rund um die Programmierung anbietet (Dogs & Klimmer, 2005, S. 85).

eXtreme Programming

Die XP-Werte sind Einfachheit, Kommunikation, Mut, Feedback und Respekt (Fowler, 2007). Das besondere bei eXtreme Programming ist die Praxis des Collective Ownership, d.h. der gesamte Code und alle Dokumente gehören dem gesamten Team. JedeR im Projekt kann zu jeder Zeit den Quellcode ändern. Gemeinsame Verantwortlichkeit bedeutet, dass bei Änderungen und Problemen das gesamte Team verantwortlich ist (Wolf, Roock & Lippert, 2005, S. 107). Mit Hilfe der gemeinsamen Verantwortlichkeit kann das Problem des vorhin beschriebenen Code Ownerships umgangen werden, denn unter diesen Umständen ist nicht nur einE EntwicklerIn von Änderungsvorschlägen betroffen. Ein weiterer Vorteil des eXtreme Programmings besteht in der Tatsache, dass KundInnen und auch EndanwenderInnen immer vor Ort für Fragen zu Verfügung stehen (sollen), und das Produkt so besser auf die Bedürfnisse der BenutzerInnen abgestimmt werden kann.

Notwendigkeit einer Preproduction-Phase

Die Methoden, die letztendlich zur Implementierung einer Produktkonzepts verwendet werden, ob dies nun agile Methoden, die flexibler auf Änderungen reagieren können, oder althergebrachte sind, haben alle gemeinsam, dass kreatives Design nicht als Bestandteil eines Designprozesses verstanden wird, also vor der Implementierung verwirklicht werden soll.

Normalerweise beginnen alle Unternehmen, die keine Softwarefirmen sind, ihren Entwicklungsprozess mit Recherche und enden mit der Produktion ihres Services oder Produkts. Sie planen sorgfältig zwischen den Phasen und sind sich der Gefahr bewusst, dass ein noch schlecht durchdachtes Produkt, dass frühzeitig produziert wird, finanzielle Einbußen bringen und den Ruf des Unternehmens schädigen kann. Diese Unternehmen wissen, dass sich die Zeit, das Geld und die Überlegungen, die in die Planung investiert werden, positiv auf die Verkaufszahlen, die Popularität, die Eleganz und die Schnelligkeit in der Erzeugung des Produkts auswirken werden (Cooper, 2004, S. XXV).

> *If design isn't done before programming starts, it will never have much effect.* (Cooper, 2004, S. 53)

Die vorhin beschriebene ambige Rolle von Einschränkungen in der Designarbeit findet in der Literatur eine eindeutige Sprache, wenn es sich um Fälle handelt, in welchen Design nach der Implementierung eingesetzt wird. Folgendes Zitat stammt aus einem Buch für Visuelle Gestaltung bereits aus dem Jahr 1994:

> *Design is not something that can be applied after the fact, when the fundamental organization of the product has already been determined - though this is indeed a common misconception. To be effective, design must be an integral part of the product development lifecycle.* (Mullet & Sano, 1994, S.7)

Dass Theorie und Praxis unterschiedlich aussehen bzw. Erkenntnisse nicht angewendet werden, wird durch mehrere Faktoren beeinflusst. Eine große Rolle spielt, dass Design, wie schon beschrieben, in vielen Softwareunternehmen keine gewichtige Rolle zugeschrieben wird, da Software Engineering als Ingeneursdisziplin betrachtet wird.

Dadurch, dass im beschriebenen Projekt (vgl. Kap. 5) ohne Design von unserer Seite mit der Implementierung begonnen wurde, konnten wir auch nicht darauf hinweisen, dass eine Zielgruppendefinition und Erforschung der Bedürfnisse wesentlich ist. Auch aus ökonomischen Gesichtspunkten ist es essentiell, Informationen über EndanwenderInnen einzuholen, und Design von Anfang an als fixen Bestandteil zu betrachten.

> *Incorporating ease of use into your products actually saves money. Reports have shown it is far more economical to consider user needs in the early stages of design, than it is to solve them later. For example, in Software enginnering: A practitioners Approach author Robert Pressman shows that for every dollar spent to resolve a problem during product design, $10 would be spent on the same problem during development, and multiply to $100 dollar or more if the problem had to be solved after the product's release.* ((Courage & Baxter, 2004, S. 20), zitiert nach (Pressman, 2005, S.13 f))

Dieses Beispiel erläutert, wie wichtig und entscheidend Informationen über UserInnen sind, um richtige Designentscheidungen treffen zu können. Wenn sich ein Problem erst nach der Auslieferung herausstellt, ist eine Änderung schwieriger zu bewerkstelligen und auch teurer. Aus diesem Grund soll es doch gewissermaßen auch im Interesse der Produktfirmen liegen, gute Produkte zu gestalten, da es erfolgversprechend und günstiger ist. Daraus folgend wirkt sich ein Prozess, in dem Design vor der Implementierung eingesetzt wird, am Ende gewinnbringend für das Unternehmen aus.

Das Widersprüchliche an der Situation scheint, dass viele Softwareunternehmen nicht bzw. weniger in eine Anfangsphase investieren wollen, in welcher die Anforderungen an ein Produkt, Ideen und Konzepte ausprobiert werden können. Stattdessen investieren sie (viel mehr) am Ende des Produktprozesses, wenn ein Produkt zu spät ausgeliefert wurde oder sein Potential nicht entfalten konnte (Buxton, 2007, S. 141).

Wenn nun einE DesignerIn oder ein Designteam für ein Projekt angestellt wird, gehen die Erwartungen an ein Produkt und den Prozess sehr weit. Dass aber einE DesignerIn alleine nur ein kleines Rad im Getriebe des Entwicklungsprozesses darstellt, und viele andere Faktoren eine gewichtige Rolle spielen, wird dabei oft vergessen. Auch die anderen Mitarbeiter, das Management, Marketing- und Businessabteilung müssen eine designzentrierte Arbeitsweise mittragen. Folgendes Zitat von Kim Goodwin unterstreicht dies:

> *"First place to start is to realize that, it's not plug and play. You know, you can't just plug a designer into a culture and expect that all the sudden you're gonna do design and live design in that company. I*

> *think it's important to realize that significant cultural change takes years ...* " Kim Goodwin (Keirnan, 2008)

Der Weg zu einem Unternehmen, in welchem Design bei Produkten seinen fixen Platz besitzt, ist ein langer und steiniger, dieser Prozess geschieht nicht von heute auf morgen. Die Frage lautet nun, wie Design in einem Unternehmen etabliert werden kann. Ein guter Ansatz liegt in eXtreme Programming, hier arbeiten KundInnen bzw. EndanwenderInnen eng mit den EntwicklerInnen zusammen (Wolf et al., 2005, S. 25). Somit wird auch sichergestellt, dass sich das Entwicklerteam am richtigen Weg befindet, und die KundInnen zufrieden sind, was aber nicht impliziert, dass die EndanwenderInnen ebenso fühlen. Diese Vorgangsweise beinhaltet allerdings noch kein kreatives Design in der Anfangsphase, bevor grundlegend mit der Implementierung begonnen wird, sondern stellt nur sicher, dass während der Entwicklung noch Raum für Gestaltung und Änderungen ist.

Zusammenfassung

Projekte, die in der Anfangsphase kein gestalterisches Design einsetzen, sind neben ökonomischen und kreativen Nachteilen vielerlei Erschwernissen unterworfen. Die Rolle von Einschränkungen spielen im Zusammenhang solcher Projekte eine wichtige Rolle. Nicht nur dass Änderungen im Nachhinein bei den EntwicklerInnen durch Code Ownership behindert werden, über Umgestaltungen im Design muss erst mit den anderen InteressensvertreterInnen argumentiert werden. Außerdem sind Änderungen im Nachhinein, wenn einmal durchgesetzt, aufwändiger und kostspieliger in der Umsetzung.

Agile Methoden und im Besonderen eXtreme Programming wurden als Mittel diskutiert, Änderungen leichter durchführen zu können. Denn der Erfolg bei der Umsetzung von Redesignvorschlägen hängt mit der Bereitschaft der Verantwortlichen ab, diese auch zu implementieren.

7 Konklusion

Design bedeutet immer auch Kompromisse einzugehen, denn viele Interessensgruppen wollen ihre Ziele in einer Produktentwicklung durchsetzen bzw. ihre Prioritäten realisieren. Auch im Projekt brachten wir Vorschläge ein, die wir als wesentlich für das Redesign betrachteten aber von den Programmierern aus Gründen der ressourcenintensiven Implementierung nicht übernommen wurden. Der Vorschlag eines User Testings wurde nicht realisiert, weil diese Notwendigkeit nicht erkannt wurde. Grundsätzlich stoßen in einem interdisziplinären Projekt die unterschiedlichsten Perspektiven und Erfahrungen aufeinander, und es ist eine Frage der Bereitschaft der ProjektmitarbeiterInnen, eine gemeinsame Sprache zu finden und Verständnis für die gegenseitigen Anliegen zu schaffen. Die Zusammenarbeit wird deutlich begünstigt, wenn Teammitglieder vor Ort und jederzeit zugegen sind, und infolgedessen sofort auf auftauchende Fragestellungen eingegangen werden kann. Wie beschrieben sind Wissen und Tun eng miteinander verbunden und bedingen sich gegenseitig. Viele Fragen ergeben sich erst während der Implementierung eines Produkts, es kann nicht vollends im Vorhinein geplant und durchdacht werden.

Auch während eines Projekts existiert noch Raum für Kreativität und Innovation, allerdings ist dieser Raum im Vergleich zum Projektstart, bei dem noch nicht implementiert wurde, sehr beengt. Die für das nicht technische Design wichtigen Freiheiten sind durch bereits getroffene Entscheidungen und der Schwierigkeit, Änderungen im Code im Nachhinein zu verwirklichen, beschränkt. Falls die Projektumstände so gegeben sind, dass die ProgrammiererInnen das Produkt entwickelt haben, reflektiert das implementierte Design vielfach die dahinter liegenden technologischen Konzepte anstelle der Interessen der BenutzerInnen. Diese Entscheidungen wirken sich negativ auf das Produkt aus, denn die ProgrammiererInnen verwenden die Software nicht, sie entwickeln diese, und

haben aus diesem Grund andere Vorstellungen und Fähigkeiten. Aufgabe des kreativen Designverständnisses bei Softwareprodukten ist es, verständliche, ästhetische Produkte für BenutzerInnen zu schaffen, die die zu absolvierenden Aufgaben in den Vordergrund und die Software selbst in den Hintergrund rücken lassen.

Ein erster Schritt in diese Richtung ist, die Softwareentwicklung nicht als rationellen Prozess zu betrachten, und von den Wurzeln der Ingeneursdisziplin loszulösen. Großes Potential liegt in agilen Methoden, die adaptiv auf Veränderungen eingehen können und eine neue Herangehensweise in der Softwareentwicklung verkörpern. Wenn diese Methoden noch soweit gedeihen, dass die User Experience miteinbezogen wird, ist ein wichtiger Wendepunkt erreicht. Es wurden einige vielversprechende Projekte durchgeführt, die eXtreme Programming gepaart mit User Interaction Design in Iterationen realisiert haben. Die Ergebnisse zeigten, dass sowohl die ProgrammiererInnen als auch die DesignerInnen mit dem Prozess, den Projekten und der Zusammenarbeit sehr zufrieden und erfolgreich waren. Einer Implementierungs- ging jedes mal eine Designphase voraus, wenn Probleme auftraten konnte Rücksprache mit den DesignerInnen gehalten werden (Ferreira, Noble & Biddle, 2007).

Der Aspekt der Präsenz des Designteams wurde in den Nachbetrachtungen zum Projekt bereits angesprochen. Es ist schwer durchführbar, einen Designvorschlag im Vorhinein so auszuarbeiten, dass sich keine Fragen vonseiten der ProgrammiererInnen ergeben. Wenn (gegenwärtig) keine Ansprechperson vor Ort ist, entscheiden die ProgrammiererInnen selbst. Daraus folgt die Feststellung, dass eine engere Zusammenarbeit die Erreichung des gemeinsamen Ziels, ein gutes Produkt zu entwickeln, begünstigt.

In Anbetracht des Projektes und den theoretischen Prämissen kann festgehalten werden, dass ein Projektanfang durch eine explizite Design- und Planungsphase charakterisiert sein soll. Projekte, die direkt mit der Implementierung beginnen, lassen weniger Raum für Kreativität, der Beobachtung der AnwenderInnen und der Erkundung des Verwendungskontexts. Um die Metapher des dreibeinigen Stuhls (vgl. Kap. 2, S. 19) heranzuziehen, wird in solchen Projekten nur das technische Verständnis gefördert und die beiden anderen Aspekte kommen zu kurz. Einzig wenn alle drei Beine im Entwicklungsprozess einbezogen werden,

kann der Suhl stehen, und nur dann können innovative und überzeugende Produkte entstehen.

Literatur

Ackoff, R. L. (1974). *Redesigning the future*. Wiley.

Alben, L. (1996). Quality of experience: defining the criteria for effective interaction design. *interactions*, *3*(3), 11–15.

Bartunek, J. M. (1993). Rummaging behind the scenes of organizational change and finding role transitions, illness, and physical space. In R. W. Woodman & W. A. Pasmore (Hg.), *Research in organizational change and development* (Bd. 7, S. 41–76). JAI Press.

Beyer, H. & Holtzblatt, K. (1997). *Contextual design : A customer-centered approach to systems designs (morgan kaufmann series in interactive technologies)*. Morgan Kaufmann.

Buur, J. & Bagger, K. (1999). Replacing usability testing with user dialogue. *Commun. ACM*, *42*(5), 63–66.

Buxton, B. (2007). *Sketching user experiences: Getting the design right and the right design*. Morgan Kaufmann.

Conklin, J. (2006). *Dialogue mapping: Building shared understanding of wicked problems* (Bd. Chapter 1). CogNexus Institute: Wiley.

Cooper, A. (2004). *The inmates are running the asylum : Why high tech products drive us crazy and how to restore the sanity (2nd edition)*. Sams.

Cooper, A. & Reimann, R. M. (2003). *About face 2.0: The essentials of interaction design*. Wiley.

Corporation, M. (2001). *Windows xp visual guidelines.* http://interface.free.fr/Archives/ GUI_Xp.pdf.

Courage, C. & Baxter, K. (2004). *Understanding your users: A practical guide to user requirements methods, tools, and techniques*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Dewey, J. (2008). *How we think*. Cosimo Classics.

Dicks, R. S. (2002). Mis-usability: on the uses and misuses of usability testing. In *Sigdoc '02: Proceedings of the 20th annual international conference on computer documentation* (S. 26–30). New York, NY, USA: ACM Press.

Dix, A., Finlay, J. E., Abowd, G. D. & Beale, R. (2003). *Human-computer interaction (3rd edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

Dogs, C. & Klimmer, T. (2005). *Agile software-entwicklung kompakt*. mitp-Verlag.

Duh, H. B.-L., Tan, G. C. B. & Chen, V. H. hua. (2006). Usability evaluation for mobile device: a comparison of laboratory and field tests. In *Mobilehci '06: Proceedings of the 8th conference on human-computer interaction with mobile devices and services* (S. 181–186). New York, NY, USA: ACM.

Dumas, J. S. (1989). Stimulating change through usability testing. *SIGCHI Bull., 21*(1), 37–44.

Eclipse, F. (2008). *Eclipse - an open development platform.* http://www.eclipse.org/.

Ehn, P. (2002). Neither bauhäusler nor nerd educating the interaction designer. In *Dis '02: Proceedings of the conference on designing interactive systems* (S. 19–23). New York, NY, USA: ACM Press.

Ferreira, J., Noble, J. & Biddle, R. (2007). Agile development iterations and ui design. In *Agile '07: Proceedings of the agile 2007* (S. 50–58). Washington, DC, USA: IEEE Computer Society.

Forlizzi, J. & Battarbee, K. (2004). Understanding experience in interactive systems. In *Dis '04: Proceedings of the 2004 conference on designing interactive systems* (S. 261–268). New York, NY, USA: ACM Press.

Fowler, M. (2007). *The new methodology.* http://martinfowler.com/articles/newMethodology.html.

Galitz, W. O. (2002). *The essential guide to user interface design: an introduction to gui design principles and techniques* (Second Edition Aufl.). New York, NY, USA: John Wiley & Sons, Inc.

Garrett, J. J. (2002). *The elements of user experience: User-centered design for the web*. New Riders Press.

Gasson, S. (2007). *Emergence in organizational 'problem-solving': Theories of social cognition.* http://www.cis.drexel.edu/faculty/gasson/papers/ProbSolv.html.

Gedenryd, H. (1998). *How designers think*. Unveröffentlichte Dissertation, University of Lund, Lund Sweden.

Halskov, K. & Dalsgard, P. (2006). Inspiration card workshops. In *Dis '06: Proceedings of the 6th acm conference on designing interactive systems* (S. 2–11). New York, NY, USA: ACM Press.

Herczeg, M. (2006). *Interaktionsdesign*. München: Oldenbourg Verlag.

Hoefnagels, S., Geelhoed, E., Stappers, P. J., Hoeben, A. & Lugt, R. van der. (2004). Friction in scheduling and coordinating lives of families: designing from an

interaction metaphor. In *Dis '04: Proceedings of the 2004 conference on designing interactive systems* (S. 321–324). New York, NY, USA: ACM Press.

Information & Software Engineering Group, A. S. (2007). *ïnfinica", development of a document-oriented form-management system.* http://www.ifs.tuwien.ac.at/node/4855.

Johnson, S. (1999). *Interface culture. wie neue technologien kreativität und kommunikation verändern.* Klett-Cotta.

Jones, L. & Greene, S. L. (2000). Moma and the three-legged stool: fostering creative insight in interactive system design. In *Dis '00: Proceedings of the conference on designing interactive systems* (S. 39–47). New York, NY, USA: ACM Press.

Keirnan, T. (2008, January). *Design critique 40: Interview with kim goodwin at user interface 12.* http://www.designcritique.net/index.php?post_id=301487.

Kruchten, P. (2005, March-April). Editor's introduction: Software design in a postmodern era. *Software, IEEE, 22*(2), 16-18.

Kushalani, A., Smith, R. & Howard, S. (1994). What happens when designers don't play by the rules: Towards a model of opportunistic behaviour in design. *Australian Journal of Information Systems*, 13 - 31.

Lanzara, G. F. (1983). The design process: Frames, metaphors and games. *Systems Design For, With and By The Users*, 29-40.

Lawson, B. (1997). *How designers think, third edition: The design process demystified.* Architectural Press.

Lawson, B. (2004). *What designers know.* Architectural Press.

Löwgren, J. (1995). Applying design methodology to software development. In *Dis '95: Proceedings of the 1st conference on designing interactive systems* (S. 87–95). New York, NY, USA: ACM.

Löwgren, J. & Stolterman, E. (2007). *Thoughtful interaction design: A design perspective on information technology.* The MIT Press.

Mackay, W. E. (2004). The interactive thread: exploring methods for multi-disciplinary design. In *Dis '04: Proceedings of the 2004 conference on designing interactive systems* (S. 103–112). New York, NY, USA: ACM Press.

Maeda, J. (2006). *The laws of simplicity (simplicity: Design, technology, business, life).* The MIT Press.

Mandel, T. (1997). *Elements of user interface design.* New York, NY, USA: John Wiley & Sons.

Moggridge, B. (2006). *Designing interactions*. The MIT Press.

Mullet, K. & Sano, D. (1994). *Designing visual interfaces: Communication oriented techniques*. Prentice Hall PTR.

Nielsen, J. (2000). *Usability engineering*. New York: AP Professional.

Norman, D. A. (2002). *The design of everyday things*. Basic Books.

Norman, D. A. (2005). *Emotional design: Why we love (or hate) everyday things*. Basic Books.

Pierce, J. L., Kostova, T. & Kirks, K. T. (2001). Toward a theory of psychological ownership in organizations. *Academy of Management Review*, *26*, 289–310.

Pressman, R. S. (2005). *Software engineering: A practitioner's approach* (Sixth Edition Aufl.). New York, NY: McGraw-Hill.

Qualysoft, G. (2008). *Infinica document solutions.* http://www.infinica.com/infinica/at/en/1Home/.

Rittel, H. W. & Webber, M. M. (1973, June). Dilemmas in a general theory of planning. *Policy Sciences*, *4*(2), 155–169.

Rubin, J. (1994). *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests*. New York: Wiley.

Saffer, D. (2006). *Designing for interaction: Creating smart applications and clever devices (voices that matter)*. Peachpit Press.

Sarodnick, F. & Brau, H. (2006). *Methoden der usability evaluation. wissenschaftliche grundlagen und praktische anwendung*. Bern: Huber.

Schön, D. A. (1990). *Educating the reflective practitioner : Toward a new design for teaching and learning in the professions (higher education series)*. Jossey-Bass.

Shneiderman, B. & Plaisant, C. (2005). *Designing the user interface : Strategies for effective human-computer interaction (4th edition)*. Addison Wesley.

Smith, M. K. (2007). *donald schon (schön): learning, reflection and change.* http://www.infed.org/thinkers/et-schon.htm#_The_reflective_practitioner.

Stapelkamp, T. (2007). *Screen- und interfacedesign: Gestaltung und usability für hard- und software (x.media.press)*. Springer.

Starck, P. (2008). *S+arck.* http://www.philippe-starck.com/.

Stone, D., Jarrett, C., Woodroffe, M. & Minocha, S. (2005). *User interface design and evaluation (the morgan kaufmann series in interactive technologies)*. Morgan Kaufmann.

Taylor, P. (2001). Interpreting mayall's 'principles in design'. *Software Engineering Conference, 2001. Proceedings. 2001 Australian*, 297-305.

Tidwell, J. (2005). *Designing interfaces : Patterns for effective interaction design*. O'Reilly Media, Inc.

Virzi, R. A., Sokolov, J. L. & Karis, D. (1996). Usability problem identification using both low- and high-fidelity prototypes. In *Chi '96: Proceedings of the sigchi conference on human factors in computing systems* (S. 236–243). New York, NY, USA: ACM Press.

W3C. (2007a). *Xforms 1.1.* http://www.w3.org/TR/xforms11/#concepts.

W3C. (2007b). *Xsl transformations (xslt) version 2.0.* http://www.w3.org/TR/xslt20/#what-is-xslt.

W3schools. (2007). *Introduction to xsl-fo.* http://www.w3schools.com/xslfo/xslfo/intro.asp.

Wang, Q., Battocchi, A., Graziola, I., Pianesi, F., Tomasini, D., Zancanaro, M. et al. (2006). The role of psychological ownership and ownership markers in collaborative working environment. In *Icmi '06: Proceedings of the 8th international conference on multimodal interfaces* (S. 225–232). New York, NY, USA: ACM.

Wolf, H., Roock, S. & Lippert, M. (2005). *extreme programming: Eine einführung mit empfehlungen und erfahrungen aus der praxis*. it-agile.