Universität Paderborn
Fakultät für Elektrotechnik, Informatik und Mathematik

SS 2004

Seminararbeit

# Usability im Software-Entwicklungszyklus

Sebastian Aland
sebaland@upb.de

Florian Schoppmann
fschopp@upb.de

2. August 2004

Teil des Seminars „Perspektiven der Software-Ergonomie“

Prof. Dr.-Ing. Reinhard Keil-Slawik
Arbeitsgruppe Informatik und Gesellschaft

Betreuerin: Joanna Slawik

# Inhaltsverzeichnis

**1 Einführung** **4**
1.1 Usability . . . 4
1.2 Usability-Engineering . . . 5

**2 Probleme und Vorurteile** **6**
2.1 „Usability ist nur nice-to-have, d.h. Luxus“ . . . 6
2.2 „Es reicht, wenn sich Entwickler an Gestaltungsrichtlinien halten“ . . . 6
2.3 „Usability ist Sache der Marktforschung“ . . . 7
2.4 „Usability ist subjektiv“ . . . 7
2.5 „Usability besteht nur aus Richtlinien“ . . . 8
2.6 „Bei Usability geht es nur um DAUs“ . . . 8
2.7 „Usability ist teuer“ . . . 8

**3 Motivation** **9**
3.1 Usability kann zu enormen Umsatzsteigerungen führen . . . 9
3.2 Usability erhöht die Erfolgssicherheit und Planbarkeit . . . 10
3.3 Neue Ideen fließen frühzeitig in den Entwicklungsprozess ein . . . 10
3.4 Usability kann Zeit und Ressourcen sparen . . . 11
3.5 Die Benutzeroberfläche macht einen großen Anteil des Quellcodes aus . . . 11
3.6 Usability erhöht Markenwert und Kundenloyalität, Joy of Use . . . 12
3.7 Usability spart Kosten . . . 12

**4 Wirtschaftliche Aspekte** **13**
4.1 Kosten-Nutzen-Analyse . . . 14
4.2 Total Cost of Ownership (TCO) . . . 15
4.3 Return on Investment (ROI) . . . 15

**5 Prozessmodelle** **16**
5.1 Norm DIN EN ISO 13407 . . . 16
5.2 User-Centered Design Process . . . 18

**6 Vorgeschlagene Prozessmodelle im Detail** **18**
6.1 Phase 1: Anforderungsanalyse . . . 19
6.2 Phase 2: Design/Test/Entwicklung . . . 22
6.3 Phase 3: Installation . . . 23
6.4 Prototyping . . . 24
6.5 Gestaltungsrichtlinien . . . 25
6.6 Usability-Engineering und Object Oriented Software Engineering . . . 26

**7 Resümee** **27**
7.1 „Todsünden“ und übliche Fehler beim Usability-Engineering . . . . . 27
7.2 Abschließende Bemerkungen . . . . . 30

**Abbildungsverzeichnis** **31**

**Tabellenverzeichnis** **31**

**Literatur** **32**

In dieser Seminararbeit betrachten wir ausgewählte Aspekte bei der Erzielung von Usability in der Software-Entwicklung. Dabei gehen wir anfangs auf typische Probleme und insbesondere Vorurteile ein, auf die man in der Praxis stößt. Zusätzlich beschreiben wir typische Argumentationsweisen für die Durchführung von Usability-Maßnahmen in Unternehmen und die für ihre wirtschaftliche Bewertung üblichen Metriken. Im Weiteren greifen wir aktuelle anerkannte Prozessmodelle – dabei insbesondere die Ideen von Deborah Mayhew und Jacob Nielsen – auf, die einen benutzerorientierten Entwicklungsprozess fordern. In diesem Zusammenhang werden auch einige Techniken kurz angerissen, die in den weiteren Arbeiten dieser Seminarreihe vertieft werden.

# 1 Einführung

Über 3 Millionen Ergebnisse lieferte die Suchmaschine Google Anfang Juli 2004 bei der Suche nach *Usability*. Zweifellos findet der Begriff mittlerweile häufige – fast meint man inflationäre – Verwendung. Er ist zu einem Mode- und Schlagwort geworden, dessen genaue Bedeutung oftmals jedoch nicht klar ist.

## 1.1 Usability

Wäre eine streng wörtliche Übersetzung eigentlich „Benutzbarkeit“, so hat sich im Deutschen zunehmend das viel treffendere Wort *Gebrauchstauglichkeit* durchgesetzt. Die insbesondere in der Umgangssprache übliche Beschreibung von Usability als „Benutzerfreundlichkeit“ ist unter Usability-Fachleuten nicht ganz unumstritten. So schreibt Nielsen in [Nie93, S. 23], es sei nicht nötig, dass Maschinen freundlich zu den Benutzern seien – sie sollten vielmehr nicht im Weg stehen. Benutzerfreundlichkeit sei daher ein „unnötig anthropomorphischer Begriff“. Außerdem, so Nielsen weiter, impliziere der Begriff eine eindimensionale Sicht auf die Bedürfnisse von Benutzern. Wäre ein System „freundlich“ zu einem Benutzer, so könnte es aufgrund anderer Anforderungen für einen anderen eher anstrengend und lästig sein.

Eine umfassendere Definition von Usability im Zusammenhang mit Software-Entwicklung findet sich in der Norm DIN EN ISO 9241, Teil 11 (Richtlinien zur Gebrauchstauglichkeit), die Gebrauchstauglichkeit und damit Usability beschreibt als das

> Ausmaß, in dem ein Produkt durch bestimmte Benutzer in einem bestimmten Nutzungskontext dazu genutzt werden kann, bestimmte Ziele effektiv, effizient und zufrieden stellend zu erreichen.

## 1.2 Usability-Engineering

Eine Beschäftigung mit Usability im Software-Entwicklungszyklus setzt zwangsläufig auch eine Klärung von *Usability-Engineering* voraus, ist doch insbesondere im Angelsächsischen die ingenieurmäßige Vorgehensweise sehr beliebt. In [May99, S. 2f.] beschreibt Mayhew:

> Usability Engineering is a discipline that provides structured methods for achieving usability in user interface design during product development. It is a discipline with roots in several other basic disciplines, including cognitive psychology, experimental psychology, ethnography, and software engineering.

und weiter:

> Usability adapts these general components of software engineering [gemeint sind: Anforderungsdefinitionen, Zielsetzung, iteratives Design und Testen bis zur Zielerfüllung] to provide an engineering-like process for the design and development of usable product user interfaces.

Nielsen stellt darüber insbesondere heraus, dass Usability-Engineering keine einmalige Einzelangelegenheit sei, die die Benutzeroberfläche schon vor der Freigabe eines Produkts festlegt. Vielmehr finden die mit Usability-Engineering verbundenen Aktivitäten während des gesamten Lebenszyklus eines Produkts statt, mit einem signifikanten Anteil bereits während der Anfangsphasen, noch bevor überhaupt die Benutzeroberfläche ausgestaltet ist.

Die Grundprinzipien des Usability-Engineering wurden schon relativ früh (1985) von Gould and Lewis erkannt und beschrieben: „Early focus on users and tasks“, „Empirical Measurement“ und „Iterative Design“ [GoL85, S. 300f.]. Bemerkenswerterweise kam man damals jedoch zu dem Ergebnis, dass diese Prinzipien seinerzeit wenig in der Praxis verbreitet und offenbar weit weniger intuitiv als zuvor angenommen waren.

Zusammenfassend kann also festgestellt werden, dass Usability-Engineering noch eine relativ junge Disziplin ist. Sowohl als Konsequenz hieraus als auch der Natur des Ingenieurmäßigen geschuldet, sind viele in dieser Seminararbeit zitierte Quellen

keine rein wissenschaftlich gewonnenen Ergebnisse, sondern beruhen oftmals auf Erfahrungswerten aus der Praxis.

# 2 Probleme und Vorurteile

Mayhew und Kalbach stellen in [May99, S. 405] bzw. [HeV03, S. 14f.] einige pointierte Aussagen vor, die den Kern der oft in der Praxis vorherrschenden Meinung bezüglich der Verankerung von Usability und Usability-Engineering im Software-Entwicklungszyklus widerspiegeln.

## 2.1 „Usability ist nur nice-to-have, d.h. Luxus“

Mayhew schreibt:

> Back in the late 1970s and 1980s, it was common to hear managers and developers make this statement [„The quality of the user interface doesn't really matter“]. And, to a certain extend, it was true.

Nachvollziehbar ist diese Aussage für die damalige Zeit, da die Ära der PCs – also *Personal* Computer – gerade erst begann. Entsprechend hatte ein Großteil der seinerzeit entwickelten Systeme einen stark eingeschränkten Nutzerkreis, oftmals beschränkte er sich sogar auf geschulte Experten. Zumindest „in a bottom-line sense“, so Mayhew weiter, war die Benutzeroberfläche also unerheblich.

Eine Begründung für die auch heute weiterhin vorhandene Verbreitung dieser Denkweise lässt sich am ehesten in dem Paradigmenwechsel finden, der in vielen Firmen offenbar eine längere Zeit benötigt.

## 2.2 „Es reicht, wenn sich Entwickler an Gestaltungsrichtlinien halten“

Oftmals, so Kalbach, ist in der Praxis die Meinung zu hören, es genüge, „wenn Entwickler auf eine hohe Benutzerfreundlichkeit achten“. Auch Mayhew geht auf diese Ansicht ein:

> As long as designers are familiar with available user interface guidelines and principles, they will design good user interfaces.

Kern beider Aussagen ist die Reduzierung von Usability auf die Einhaltung von Oberflächen-Gestaltungsrichtlinien. Überspitzt formuliert, sieht man hier die Meinung vertreten, dass die korrekte Verwendung und Platzierung von GUI-Elementen wie Schaltflächen, Textfeldern etc. für gute Usability bereits ausreichend ist. Neben der Einsicht, dass *Gebrauchstauglichkeit* von mehr Faktoren bestimmt wird, wird hierbei ebenfalls übersehen, dass Entwickler

> in der Regel kaum in der Lage sind, sich in Nutzer hinein zu versetzen, die weder ihre umfangreichen IT-Kenntnisse haben noch sich mit dem jeweiligen Produkt schon auskennen. [HeV03, S. 15]

## 2.3 „Usability ist Sache der Marktforschung“

Kalbach sagt hierzu:

> Da sich sowohl Usability als auch Marktforschung mit den Käufern bzw. Benutzern eines Produktes beschäftigen, kursiert das Missverständnis, sie seien eigentlich identisch.

Problematisch hierbei ist jedoch, dass die Perspektiven unterschiedlich sind: Kalbach gibt als Beispiel eine (hypothetische) Befragung: Stellt sich hierbei heraus, dass

> 80% der Besucher einer Website die Marke als „kommunikativ“ und „innovativ“ und damit positiv empfinden, heißt das noch lange nicht, dass sie die Website auch bedienen können.

## 2.4 „Usability ist subjektiv“

Sowohl Mayhew als auch Kalbach führen diesen Punkt auf. Oftmals wird in der Praxis die Meinung vertreten, dass Usability vor allem Geschmackssache und der Rest mit gesundem Menschenverstand von Software-Entwicklern selbst herleitbar sei. Kalbach nennt ein Beispiel („Banner Blinkness“), warum jedoch Usability nicht unbedingt intuitiv sein müsse:

> Es wird eigentlich erwartet, dass größere, in der Regel blinkende Elemente auf einer Webpage bevorzugt die Aufmerksamkeit des Users gewinnen. Dies ist jedoch nicht immer so: Internet-Nutzer haben gelernt, Elemente zu ignorieren, die visuell vom Rest der Seite getrennt sind – egal, wie groß oder animiert sie sind.

Abgesehen davon ist offensichtlich, dass Usability sehr wohl messbar ist und objektiv diskutiert werden kann: Durch Benutzertests, Umfragen, Interviews etc. lassen sich bspw. Größen wie Fehler pro Stunde bei Benutzung einer Software oder u. a. auch die Anwenderzufriedenheit ermitteln. Natürlich ist insbesondere bei letzterem ein Einzelergebnis eine subjektive Meinung, jedoch ist bei einer großen heterogenen Benutzermenge das empirisch gewonnene Ergebnis sicherlich hinreichend zuverlässig.

## 2.5 „Usability besteht nur aus Richtlinien“

Kalbach greift dieses Vorurteil auf und beschreibt, dass oftmals der Eindruck besteht, Usability fördere aufgrund von starker Regel-Lastigkeit die Gestaltung langweiliger Oberflächen. Diese Meinung lässt jedoch außer Acht, dass ein „guter benutzerorientierter Prozess [...] weit über Richtlinien hinausgeht“. Im Gegenteil – folgt man den Überlegungen von Norman in [Nor02], so ist eine ansprechende Oberfläche integraler Bestandteil von Usability:

> Use a pleasing design, one that looks good and feels, well, sexy, and the behavior seems to go along more smoothly, more easily, and better. Attractive things work better.

## 2.6 „Bei Usability geht es nur um DAUs“

Dieser ebenfalls in der Praxis zu hörenden Meinung liegt die Vorstellung zugrunde, Usability diene insbesondere dazu, Software idiotensicher zu machen. Diese Sichtweise ignoriert jedoch, dass für die Bewertung von Usability typische Nutzer herangezogen werden. – Bei komplexen Systemen sind dies gerade keine DAUs, sondern u. U. sogar sehr erfahrene Benutzer.

Abgesehen davon, ist es natürlich Ziel, eine Software robust gegen Falscheingaben zu machen – diese Anforderung ist aber nicht auf das Usability-Engineering beschränkt, sondern wird gleichermaßen auch beim Software Engineering gefordert.

## 2.7 „Usability ist teuer“

Schließlich scheitern die Planung und Umsetzung von Usability-Maßnahmen nicht selten auch an der Überlegung, sie seien zu teuer. Statt dessen wird das gesparte Geld lieber für weitere Funktionalität investiert.

Hier zeigt sich jedoch, dass in der Praxis oftmals überhaupt kein genauer Geschäftsplan aufgestellt wird, der diese These belegen könnte. Im Gegenteil: Eine Argumentation über einen umfassenden Geschäftsplan führt in der Regel vor, dass Usability-Maßnahmen sogar zu Kostensenkungen führen. Im Kapitel 4 soll auf die ökonomischen Aspekte von Usability genauer eingegangen werden.

# 3 Motivation

Es wurde bereits gezeigt, dass viele verbreitete Einwände gegenüber Usability auf einseitigen oder veralteten Sichtweisen beruhen, zu deren Entkräftung sich einfach Argumente finden lassen. Um in der Praxis Usability fest zu verankern und Entscheidungsträger in Unternehmen zu überzeugen, reicht dies oft jedoch noch nicht aus. Hier muss insbesondere gezeigt werden, dass Usability nicht nur keine *Nachteile* hat, sondern sogar große *Vorteile* – idealerweise illustriert durch Beispiele (*Erfolgsgeschichten*).

Die Argumentation im Folgenden orientiert sich an der Reihenfolge, in der die Vorurteile gegenüber Usability genannt wurden.

## 3.1 Usability kann zu enormen Umsatzsteigerungen führen

Web-Seiten bzw. Web-Entwicklungen stellen hierzu ein Paradebeispiel dar, da Usability sofortige Auswirkungen auf das Kundenverhalten hat. Fällt die Bedienung bspw. eines Online-Shops schwer, so kann (unter der Prämisse eines funktionierenden Marktes: Nachfrage bestimmt Angebot, Konkurrenz ist vorhanden) davon ausgegangen werden, dass viele Kunden ihre Einkaufsvorgänge frühzeitig abbrechen und zur Konkurrenz wechseln.

Ein von Battey in [Bat99] dokumentiertes Beispiel (1999) ist die Umgestaltung der Web-Seite von IBM im Frühjahr 1999:

> The company says in the month after the February re-launch, traffic to the Shop IBM online store increased 120 percent, and sales went up a whopping 400 percent.

In [Usa04-1] ist zu selbigem Fall zu lesen:

> On IBM's website, the most popular feature was the search function, because the site was difficult to navigate. The second most popular feature was the 'help' button, because the search technology was so ineffective.

> IBM's solution was a 10-week effort to redesign the site, which involved more than 100 employees at a cost estimated 'in the millions.' The result: In the first week after the redesign, use of the 'help' button decreased 84 per cent, while sales increased 400 per cent.

Es zeigt sich, dass Usability keineswegs Luxus, sondern im Gegenteil wenn nicht gar Bedingung, so doch immens wichtig für den Geschäftserfolg ist.

## 3.2 Usability erhöht die Erfolgssicherheit und Planbarkeit

Kalbach schreibt in [HeV03, S. 8]:

> Letztlich wird jedes Produkt einem Usability Test unterzogen – spätestens bei der ersten Nutzung durch den Endverbraucher. Wenn dieser Test negativ ausfällt, ist der Aufwand für den Herstellenden erheblich höher, dem Produkt doch noch zum Erfolg zu verhelfen.

Folglich lassen sich durch einen Usability-orientierten Entwicklungsprozess Nutzungsprobleme frühzeitig entdecken und eine aufwändige Korrektur erst nach der Veröffentlichung der Software vermeiden.

Da ausschließlich das Einhalten von Gestaltungsrichtlinien durch die Entwickler eben nicht einem Usability-Test entspricht (wie im Abschnitt 2.2 aufgezeigt), ist das dort beschriebene Vorurteil umso mehr widerlegt.

## 3.3 Neue Ideen fließen frühzeitig in den Entwicklungsprozess ein

Bloomer, Croft und Wright beschreiben in [BCW97, S. 38] unter der Bezeichnung *Collaborative Design* (auch *Participatory Design* genannt), dass Benutzer bei Einsatz entsprechender Techniken wertvolle Beiträge zum Entwicklungsprozess leisten können:

> The team [which employed collaborative design] believes that the result will be achieved more quickly and that the interface will operate better because it has captured a better set of user requirements and designed a more usable user interface. The team also believes that the interface will require fewer enhancements and maintenance after implementation. Furthermore, the value added to the process is already evident in user feedback and management support.

Mayhew unterstützt diese Position in [May99, S. 200]: Sie schreibt, dass die Beteiligung von Anwendern den Entwicklungszyklus effizienter machen kann, indem frühzeitig die Sinnhaftigkeit von Design-Ideen sichergestellt wird, bevor sie einer strukturierten Usability-Evaluation unterzogen werden.

## 3.4 Usability kann Zeit und Ressourcen sparen

Führt man den letzten Gedanken fort, so ist schnell einzusehen, dass mit Usability-Engineering, d. h. insbesondere der frühen Einbeziehung von Nutzern, Zeit gespart werden kann: Kalbach schreibt hierzu in [HeV03, S. 9]:

> Usability-Engineering kann hier Zeit sparen, denn es hilft, zwischen möglichen Features eines Produktes oder einer Website zu priorisieren. Funktionalitäten, die den Nutzern weniger wichtig sind, müssen erst gar nicht entwickelt werden oder lassen sich auf später verschieben.

Das frühe Erkennen von Problemen, auf der anderen Seite, erlaubt zumeist eine wesentlich einfachere und schnellere Behebung, als wenn die Korrektur erst nach der Veröffentlichung erfolgen würde.

Dies entspricht der Aussage aus [MaB94], dass Designänderungen während der Entwicklungsphase 10 mal und nach dem Produktrelease 100 mal so viel kosten wie eine Änderung während der benutzerbezogenen Design-Phase.

## 3.5 Die Benutzeroberfläche macht einen großen Anteil des Quellcodes aus

Myers und Rosson zeigen in [MyR92, S. 195] auf, dass durchschnittlich ca. 48% des Quellcodes damaliger (1992) Anwendungsprogramme für die Steuerung der Benutzeroberfläche zuständig waren. Selbst wenn man unterstellt, dass sich seitdem die Qualität der Werkzeuge verbessert und der Anteil der Projekte vergrößert hat, die graphische GUI-Werkzeuge auch verwenden, bliebe man auch heute noch bei einer bemerkenswerten Zahl, denn selbst

> the projects using UIMSs [User Interface Management Systems] or interface builders spent the least percent of time and code on the user interface (around 41%) suggesting that these tools are effective.

Web-Anwendungen, auf der anderen Seite, dürften oftmals noch viel mehr Aufwand für die Oberfläche betreiben. Man kann also auch heute noch berechtigt von einem sehr hohen durchschnittlichen Anteil des Codes ausgehen, der sich mit der Benutzeroberfläche befasst.

Allein aus diesem Grund, so schreibt Nielsen im Vorwort von [Nie93], sollte es gerechtfertig sein, einen angemessenen Anteil des Software-Entwicklungsaufwands für das Sicherstellen der Gebrauchstauglichkeit von Software aufzuwenden.

## 3.6 Usability erhöht Markenwert und Kundenloyalität, Joy of Use

Nielsen zeigt in [Nie97-1] auf (1997), dass neue Besucher auf einer untersuchten E-Commerce Seite im Durchschnitt $127 pro Kopf ausgegeben haben, wiederkehrende aber $251. Solche Zahlen unterstreichen, dass es wesentlich leichter und kostengünstiger ist, einen bestehenden Kunden zu halten als einen neuen zu gewinnen.

Wie schon in Abschnitt 3.1 gesehen, ist jedoch gerade bei Web-Anwendungen und Web-Shops Usability ein entscheidender Faktor, um Loyalität zu erzielen. Kalin schreibt bspw. in [Kal99]:

> Convoluted e-commerce sites can lose up to half of their potential sales if customers can't find merchandise, according to Forrester. Hard-to-use content sites can turn off up to 40 percent of return visitors, and those losses add up quickly.

## 3.7 Usability spart Kosten

Haunold und Kuhn beschreiben in [HaK94] die Erfassung der österreichischen Katastralpläne (260.000 Pläne): Durch die Erhöhung der Durchsatzgeschwindigkeit einer einzelnen Aufgabe, die 11% des Gesamtaufwands pro Plan darstellte und die durch Usability-Maßnahmen auf 73% des zeitlichen Aufwands gesenkt wurde, konnten ca. $730.000 (= 73% von $1 Mio.) eingespart werden.

In [MaT88, S. 437] gehen Mantei und Teorey auf den erhöhten Nutzen von Usability-Maßnahmen bei größerer Benutzerzahl ein. Trägt man die Anzahl der Benutzer gegenüber den Kosten und Einsparungen von Usability-Maßnahmen ein, so ergibt sich ein Bild wie in Abbildung 1 gezeigt. Dabei soll ein Benutzer für jedes Jahr der Benutzung erneut gezählt werden. Dass auch die Kosten bei größerer Benutzerbasis steigen, wird wie folgt erklärt:

> The cost of running user studies rises with the complexity of the interface. Since an interface built for many users is somewhat more complex than one built for a few users, the cost of running the user studies increases slightly with the number of users.

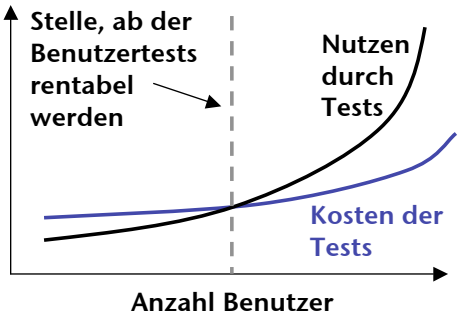


**Abbildung 1:** Kosten/Nutzen von Usability-Maßnahmen nach Mantei und Teorey

Da der Nutzen erheblich schneller steigt, lässt sich (theoretisch) ein Punkt errechnen, ab dem Usability-Maßnahmen in jedem Fall kosteneffizient sind. Auch wenn schon aus dem Jahre 1988 stammend, ist die Aussage heute nicht weniger relevant: Durch das Internet und dort insbesondere bei Web-Anwendungen ist von einer potentiell viel größeren Benutzermenge auszugehen, als dies sicherlich bei „klassischen“ Anwendungen der Fall ist.

# 4 Wirtschaftliche Aspekte

Im vorangegangenen Kapitel wurde insbesondere anhand von Beispielen aufgezeigt, dass Usability-Engineering auch aus wirtschaftlicher Sicht positive Auswirkungen hat. In diesem Abschnitt soll nun ergänzt werden, was als Grundlage für eine genauere Kosten-Nutzen-Analyse zur Rechtfertigung einzelner Maßnahmen dienen kann.

Dabei kann natürlich nur ein kleiner Ausschnitt präsentiert werden, da eine umfassende Beschreibung mit Hinweisen für die Praxis den Rahmen dieser Arbeit stark sprengen würde. Für weitere Informationen sei daher an dieser Stelle auf die zitierten Quellen verwiesen.

## 4.1 Kosten-Nutzen-Analyse

In der Praxis, so stellt Hinderberger in [HeV03, S. 24f.] fest, wird die Kosten-Nutzen-Argumentation oftmals gar nicht durchgeführt. Hierbei führt er eine Reihe von Gründen auf: Usability-Professionals haben vielfach, da von anderen Bereichen kommend, keine oder sehr geringe BWL-Kenntnisse. Statt dessen sind viele Usability-Experten es eher gewöhnt, ihre Dienstleistungen durch Erfolgsgeschichten zu verkaufen als durch individuelle Kosten-Nutzen-Berechnungen. Bei den Entscheidungsträgern in Unternehmen ist man vielfach mit dieser Argumentationsweise zufrieden, da sich die Usability-Kosten gewöhnlich in überschaubarem Rahmen halten. Einen Business-Case durchzurechnen, wäre u. U. teurer als einen Usability-Experten extern zuzukaufen. Schließlich jedoch sind Usability-Erträge in komplexeren Software-Produkten nicht einfach messbar. So werden Verbesserungen oftmals im Zuge ansonsten ebenfalls erneuerter Versionen erzielt, und eine isolierte Betrachtung der Einzelerträge verschiedener Maßnahmen ist kaum möglich.

Dennoch bleibt im Regelfall zur Durchsetzung von Usability in einem Unternehmen zu zeigen, dass Usability eine lohnende Investition ist: D. h. Usability-Maßnahmen erzielen:

- Senkung von Kosten
- Steigerung von Umsätzen und Gewinnen
- Sonstige positive Auswirkungen, wie z. B. Festigung der Kundenbindung, Verbesserung des Firmenimage etc.

Hinderberger empfiehlt, die Argumentation dabei mit dieser Priorisierung zu führen, da sie so die Erwartungshaltung vieler Manager und Controller trifft.

Ein Beispiel, bei dem die Argumentation über sonstige positive Auswirkungen zu erfolgen hätte, liefert Nielsen in [Nie93, S. 3]. Er beschreibt die Situation australischer Finanzämter, die im Durchschnitt A$2.25 rückerstatten. Trotz dieses durchschnittlich sehr niedrigen Betrages beträgt die durchschnittliche Zeit, die ein australischer Steuerzahler für die Bearbeitung seiner Steuererklärung braucht, 11 Stunden – 62% nehmen gar die Hilfe eines Steuerberaters in Anspruch. Würden die Formulare vereinfacht (d. h. die „Usability“ erhöht), könnte für die australischen Bürger zwar ein enormer Zeitvorteil erreicht werden, für die Behörden selbst wären die Einsparungen bei der (vermutlich maschinellen) Auswertung der Formulare aber eher gering, d. h. im Bereich weniger Cent.

Insbesondere bei längeren Budget-Planungen ist es aber auch möglich, die positiven betriebswirtschaftlichen Auswirkungen von Usability-Maßnahmen darzustellen. Hinderberger [HeV03, S. 29] schreibt:

> Wenn eine Argumentation mit den einzelnen Faktoren, die Usability-Maßnahmen positiv beeinflussen, nicht ausreicht, muss eine komplexe Gesamtbetrachtung durchgeführt werden. Dies ist eine so genannte Business Case-Berechnung bzw. im Falle einer Usability-Maßnahme ein Usability-Business-Case.

An dieser Stelle soll jedoch nur auf die beiden in diesem Zusammenhang verbreitetsten Metriken *Total Cost of Ownership* (TCO) und *Return on Investment* (ROI) eingegangen werden.

## 4.2 Total Cost of Ownership (TCO)

Der TCO betrachtet die Gesamtkosten einer Investition während ihres gesamten Lebenszyklus. Geht es bspw. um die Beschaffung einer komplexen IT-Anlage, so wären auch alle zusammenhängenden Kosten einzurechnen wie sie etwa für die Einarbeitung, Wartung/Reparatur, Ausfallzeiten etc. bis hin zur Entsorgung anfallen. Es ist offensichtlich, dass die Qualität einer TCO-Analyse maßgeblich von der korrekten Identifikation versteckter Kosten abhängig ist. Insbesondere Systeme mit niedrigem Usability-Grad weisen oft sehr hohe versteckte Kosten auf, die nur durch eine saubere Analyse aufgedeckt werden.

Nicht eingerechnet in den TCO wird der Nutzen, der aus einer Investition entsteht. Soll ein Vergleich von Investitionsalternativen auf Basis des TCO stattfinden, so ist dementsprechend auch das Unterlassen von Usability verbessernden Maßnahmen mit Kosten zu bewerten.

## 4.3 Return on Investment (ROI)

Der ROI, vereinfacht definiert als $\frac{\sum \text{Einkünfte}}{\sum \text{Kosten}}$, zeigt auf, welchen prozentualen Rückgewinn man nach einem gewissen Zeitraum für das eingesetzte Kapital erwarten darf. Eine Aussage über den absoluten Gewinn wird nicht getroffen. Aus diesem Grund ist der ROI ein beliebtes Maß für Usability-Entwicklungen, da sich u. a. abschätzen lässt, ob bzw. ab wann sich eine Investition rentiert. Natürlich lässt sich der ROI beliebig verfeinern – oftmals wird bspw. ein beschränkter Zeitrahmen gefordert, ab dem sich eine Investition rentieren soll. Darauf soll an dieser Stelle jedoch nicht eingegangen werden, sondern es sei auf einschlägige Literatur zu diesem Thema verwiesen.

# 5 Prozessmodelle

Nachdem in den vorherigen Kapiteln typische Probleme, Vorurteile, anschließend Vorzüge sowie Argumentationsansätze für die Verankerung von Usability-Maßnahmen aufgezeigt wurden, soll nun auf die praktische Umsetzung von Usability-Engineering eingegangen werden.

## 5.1 Norm DIN EN ISO 13407

In den letzten 15–20 Jahren wurde eine Reihe von Vorschlägen für die Ausgestaltung von Usability-Engineering gemacht (siehe Literatur-Verzeichnis), die sich seit 1999 auch in der ISO-Norm 13407 (DIN EN ISO seit November 2000) widerspiegeln.

In [JIMK03, S. 54] stellen Jokela, Iivari, Matero und Karukka einen groben Überblick dar, der hier kurz wiedergegeben werden soll:

> ISO 13407 describes user-centered design [UCD im Folgenden] from four different aspects:
>
> - Rationale for UCD
> - Planning UCD
> - Principles of UCD
> - Activities of UCD

Dabei findet sich im *Rationale*-Teil insbesondere eine Motivation für gebrauchstaugliche Systeme. Im *Principles*-Teil werden vier generelle Prinzipien herausgestellt (hier etwas zusammengefasst), die die benutzerorientierte Gestaltung charakterisieren:

- Miteinbeziehung von Benutzern
- Sinnvolle Aufteilung der Funktionen zwischen Benutzern und Technologie
- Iteration der Gestaltungslösungen
- Multi-disziplinäre Gestaltung

Unter dem Punkt *Planning* werden Hinweise gegeben, wie UCD Aktivitäten in den Gesamtprozess zu integrieren sind und unter *Activities* schließlich werden die Hauptaktivitäten für UCD herausgestellt (siehe Abbildung 2). Sie lassen sich wie folgt zusammenfassen:

**Benutzerzentrierte Gestaltung planen** Bedarf für benutzerzentrierte Gestaltung erkennen, Teamwork und Kommunikation vorbereiten

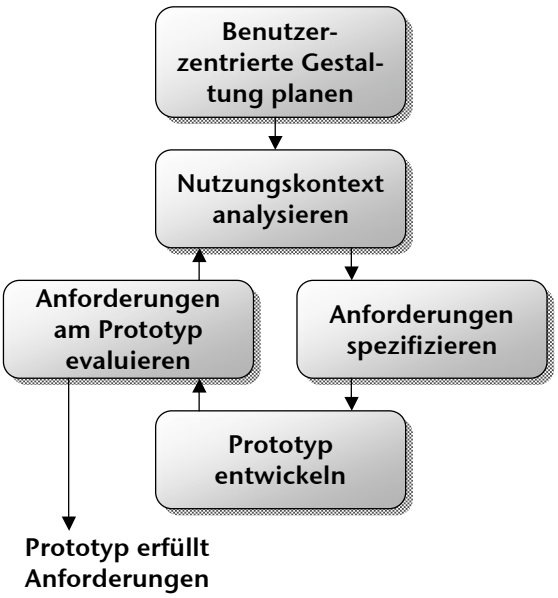


**Abbildung 2:** Prozessmodell nach ISO 13407

**Nutzungskontext analysieren** Den Benutzer, sein Arbeitsumfeld und die Aufgaben kennen lernen

**Anforderungen spezifizieren** Erfolgskriterien für die Usability festlegen in Form von Benutzeraufgaben, z. B. wie schnell ein Benutzer eine bestimmte Aufgabe erledigen kann

**Prototyp erstellen**

**Anforderungen am Prototyp evaluieren**

Wie in [JIMK03, S. 54] betont, gibt ISO 13407 keine detaillierten Methoden und Techniken an. Statt dessen trifft auf sie zu, was bspw. Baggen in [HeV03, S. 80-81] zu Normen schreibt: Sie

> liefern einen begrifflichen Rahmen und vielfältige Anregungen, die gewinnbringend in professioneller Usability-Arbeit angewendet werden können.

Eine Anwendung wäre bspw. als Bestandteil von Verträgen denkbar.

## 5.2 User-Centered Design Process

Nicht zuletzt die Existenz der Norm 13407 ist ein Indikator dafür, dass sich die Idee des User-Centered Design Process allgemein durchgesetzt hat, wie es bspw. Janson in [Jan01, S. 74] behauptet.

Eine sehr wichtige Anforderung beim Ablauf des UCD Prozesses ist die Integration mit dem Software Engineering (SE). Auch dort hat es in den vergangenen Jahren die Entwicklung weg vom konventionellen Wasserfallmodell und hin zu iterativen Abläufen gegeben. Sehr populär sind der Ansatz des *Object-Oriented Software Engineering* (OOSE) bzw. Modelle, die aus ihm hervor gegangen sind (wie z. B. Rational Unified Process, RUP). Ein detailliertes Eingehen auf diese Methodologien würde den Rahmen dieser Seminararbeit sprengen – als Charakterisierung sollen folgende Merkmale genügen, wie sie in der freien Online-Enzyklopädie WIKIPEDIA (Ende Juli 2004) zum RUP genannt werden: Iterativ, Anforderungen managen, Komponentenbasierte Architektur, Modellierungssoftware (UML), Software-Qualitäten überprüfen, Änderungen kontrollieren.

Man kann leicht nachvollziehen,

> [the philosophy and approach of OOSE] is very compatible with the Usability Engineering Lifecycle,

wie Mayhew in [May99, S. 27] schreibt. Da das iterative Vorgehen und Erstrecken von Aufgaben über mehrere Entwicklungsphasen sowohl im UCE und SE eine leichte Modifizierbarkeit erfordert, schreibt Janson in [Jan01, S. 76]:

> Usability-Engineering ist daher auch an den software-technischen Entwurfszielen Portierbarkeit, Wiederverwendbarkeit und Änderbarkeit orientiert.

In Abschnitt 6.6 wird auf die von Mayhew vorgeschlagene Verknüpfung von OOSE und ihrem Prozessmodell detaillierter eingegangen.

# 6 Vorgeschlagene Prozessmodelle im Detail

Im Folgenden werden die von Deborah Mayhew in [May99] und Jacob Nielsen in [Nie93] vorgeschlagenen Prozessmodelle genau unter die Lupe genommen. Jacob Nielsen stellte 1993 ein Modell vor, das in elf Schritte gegliedert ist. Deborah Mayhew hingegen unterteilte ihr 1999 vorgestelltes Modell, das sie „The Usability Engineering Lifecycle“ nannte, in drei Phasen und beschreibt diese in sehr detaillierter Weise. Sie schlägt vor, dass die einzelnen Phasen für jedes Software-Projekt durchschritten

werden sollen. Allerdings können bei wenig komplexen Projekten abkürzende Techniken für einzelne Schritte verwendet werden (wie sie in ihrer Grafik – siehe Abbildung 3 – mit gestrichelten Pfeilen darstellt).

Auch wenn die vorgeschlagenen Prozessmodelle unterschiedlich strukturiert sind, lassen sie sich gegenüberstellend vergleichen. Die Schritte aus Nielsens Modell können im Wesentlichen den einzelnen Phasen Mayhews Modells zugeordnet werden. Diese Phasen sind:

1. Anforderungsanaylse
2. Design/Test/Entwicklung
3. Installation

## 6.1 Phase 1: Anforderungsanalyse

**Benutzerprofil erstellen** Um eine gebrauchstaugliche Software entwickeln zu können, ist es notwendig, den zukünftigen Benutzer der Software und seine Eigenschaften zu kennen und zu analysieren. Aufgrund dieser Informationen ist es ggf. zwingend, besondere Aspekte bei der Gestaltung der Oberfläche stärker zu berücksichtigen oder gar bestimmte Gestaltungselemente ganz außen vorzulassen. Die folgenden Informationen sollten also erhoben werden.

- Benutzereigenschaften: Hat die Benutzergruppe ein durchschnittlich besonders hohes oder niedriges Alter? Sind die Benutzer körperlich eingeschränkt (z. B. hör- oder sehgeschädigt)?
- Wissen und Erfahrung: Ist dies das erste Mal, dass Software im Arbeitsablauf eingesetzt werden soll, oder ist schon eine ähnliche Software im Einsatz? Sind die Benutzer erfahrene oder unerfahrene PC-Anwender?
- Arbeitsumgebung: Ist das Arbeitsumfeld extrem laut? Wird am Arbeitsplatz eine Maus verwendet oder nur eine Tastatur? Wird ein Farb- oder evtl. nur ein Monochrom-Display verwendet?
- Anfänger benutzen zu Beginn meist grafische Bedienelemente wie Buttons und lassen sich von Assistenten durch die Arbeitsschritte führen. Jedoch ist zu berücksichtigen, dass sich der Benutzer während der Arbeit mit der Software weiterentwickelt und seine Arbeit nach einiger Zeit schneller durchführen möchte, z. B. mit Hilfe von Shortcuts oder Makros. Das Design sollte so entwickelt werden, dass diese Weiterentwicklung des Benutzers möglich ist und gefördert wird.

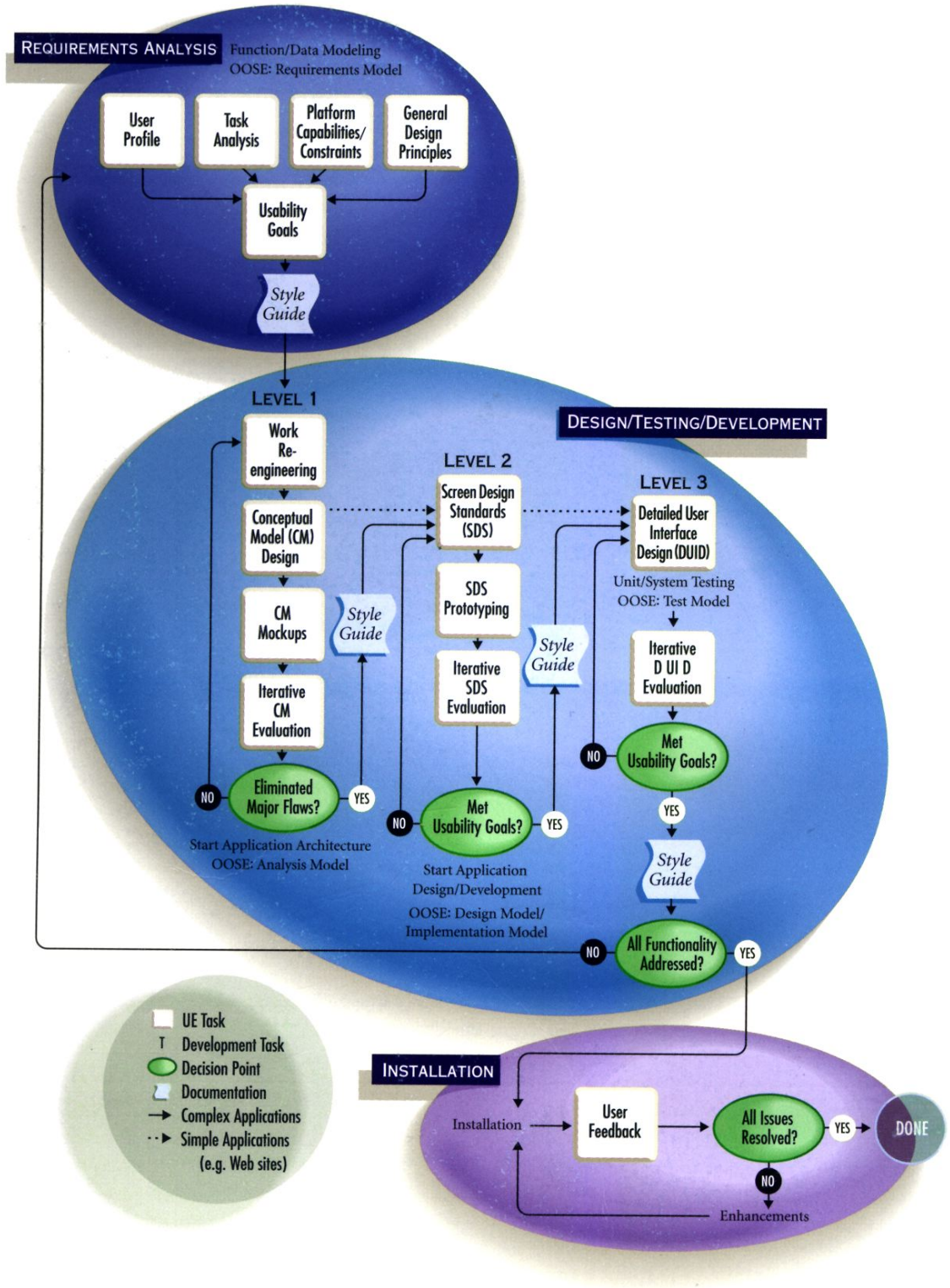


**Abbildung 3:** Der „Usability Engineering Lifecycle“ nach Mayhew [May99, Einband]

- Nielsen sieht an dieser Stelle die *Aufgabenanalyse* vor, die im Modell Mayhews erst zu Beginn der Design/Test/Entwicklung-Phase stattfinden soll. Es wird festgestellt, was die Software leisten soll und ob die Arbeitsabläufe, wie sie bisher bestehen, durch den Einsatz von Software abgeändert und somit verbessert werden sollten. [May99, S. 6], [Nie93, S. 73], [siehe Seminarthema 4: „User Requirements“]

**„Competitive Analysis“** In [Nie93, S. 78] geht Nielsen auf die „Competitive Analysis“ ein. Konkurrenz-Produkte mit ähnlichem Aufgabenfeld sollen angeschaut und als Prototypen für die eigene Software-Entwicklung verwendet werden. Dabei schließt Nielsen auch die Betrachtung von nicht-Computer-Designs mit ein, denn nicht für alle Aufgaben gibt es schon bestehende Software-Lösungen. Als Beispiele sind in etwa Computer-Enzyklopädien zu nennen, denen tatsächliche Lexika als ursprüngliches Vorbild dienten. Ein weiteres, historisches Beispiel ist die Übernahme der Schreibtisch-Metapher für graphische Oberflächen wie sie in den meisten modernen Betriebssystemen zu finden ist.

**Usability-Ziele festlegen** Man verfolgt mit dem Usability-Engineering bestimmte Ziele, die in diesem Schritt klar formuliert festgehalten werden sollen. Anhand dieser Ziele lassen sich die Ergebnisse des Usability-Engineering überhaupt erst überprüfen. Man unterscheidet qualitative und quantitative Ziele. Qualitative Ziele sind nicht messbare Ziele, die auf den Usability-Anforderungen basieren, wie z. B. dass die Oberfläche leicht und intuitiv erlernt werden kann. Quantitative Ziele sind messbar. Man setzt sich als Ziel, eine bestimmte Prozentzahl an Zufriedenheit bei den Benutzern zu erreichen. Alternativ fordert man, dass ein Benutzer im Schnitt nur eine bestimmte Anzahl an Fehlern pro Stunde mit der Software macht. Besonders deutlich lässt sich dies anhand einer „Usability-Ziel-Linie“ (siehe Abbildung 4) darstellen. [Nie93, S. 81] Der aktuelle Wert (siehe Pfeil) liegt zu Beginn über dem erstrebten Zielwert. Alles oberhalb des aktuellen Wertes ist nicht akzeptabel, da dies eine Verschlechterung der Usability bedeutet. Optimal wäre sicherlich ein Zielwert von 0, d. h. es werden mit dem Programm überhaupt keine Fehler mehr gemacht, aber dies ist natürlich utopisch.

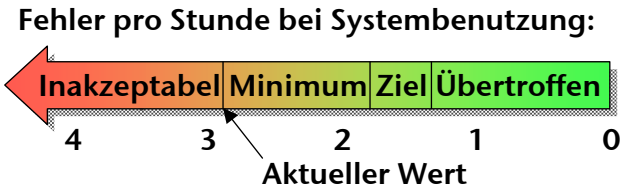


**Abbildung 4:** „Usability-Ziel-Linie“ nach Nielsen

**Gestaltungsrichtlinien formulieren** In Mayhews Modell ist nun vorgesehen, aus den gewonnenen Erkenntnissen Gestaltungsrichtlinien zu formulieren. Diese sind das Ergebnis der Anforderungsanalyse und dienen als Ausgangspunkt für alle weiteren Design-Aufgaben.

## 6.2 Phase 2: Design/Test/Entwicklung

Mayhew unterteilt diese Phase wiederum in drei Stufen.

**Stufe 1: Abstrakte Ideen**
In der ersten Stufe sollen zunächst abstrakte Ideen und Konzepte entwickelt werden. Sie beginnt mit dem Reengineering der Arbeit, wie es Nielsen schon während der Erstellung des Benutzerprofils vorgeschlagen hat. Dabei werden die Organisation und der Ablauf der Benutzeraufgaben konzipiert, ohne allerdings schon detaillierte Oberflächen-Schnittstellen zu gestalten. Es folgen der Entwurf eines konzeptionellen Modells d. h. eines High-Level-Regelwerkes für konsistente Darstellung und Navigation. Skizzen und Prototypen unterschiedlicher Design-Ideen werden erstellt und durch wiederholte Evaluationstechniken iterativ angepasst und verbessert, bis keine wesentlichen Probleme mehr auftreten.

**Stufe 2: Screen-Design unter Berücksichtigung von Gestaltungsrichtlinien**
Unter Berücksichtigung von Oberflächen-Gestaltungsrichtlinien werden erste lauffähige Prototypen erstellt. Dabei werden insbesondere Plattformstandards wie z. B. die *Microsoft Windows User Interface Guidelines* oder die *Apple Human Interface Guidelines* miteinbezogen. Objektive Bewertungstechniken werden angewendet, um das Design iterativ zu verbessern, bis die Usability-Ziele weitgehend erreicht worden sind. Die Gestaltungsrichtlinien sollten um die in dieser Stufe gewonnenen Erkenntnisse ergänzt werden.

**Stufe 3: Detailliertes Design, iterative Verbesserung**
Als letzter Designschritt im Entwicklungszyklus wird ein detailliertes, vollständiges Oberflächendesign auf Grundlage der Gestaltungsrichtlinien erstellt. Wie auch schon am Ende der vorherigen Stufen wird dieses Design iterativ evaluiert. Dafür werden formale Usability-Tests verwendet, bis die gesteckten Usability-Ziele erreicht sind.

Das Erstellen und Anwenden von Richtlinien bildet bei Mayhew einen roten Faden durch den Entwicklungszyklus. Auf das Thema Gestaltungsrichtlinien wird in Abschnitt 6.5 gesondert eingegangen.

Nielsen nimmt keine so hohe Granularisierung vor. Er legt den Fokus stärker auf Design-Techniken:

**Paralleles („diversified“) Design** Verschiedene Designer(-Teams) erarbeiten unabhängig von einander verschiedene Designvorschläge. Dabei konzentrieren sie sich jeweils auf unterschiedliche Aspekte. Ein Team gestaltet z. B. eine Anfängeroberfläche und ein anderes eine Expertenoberfläche. Am Ende werden dann die besten Design-Elemente der unterschiedlichen Entwürfe zusammengeführt. Die Parallele-Design-Phase sollte dabei je nach Komplexität höchstens ein paar Tage dauern. Besonders wichtig ist, dass die Designer nicht miteinander über ihre Design-Entwürfe reden oder sich anderweitig beeinflussen.

**Teilnehmendes („participatory“) Design** Ebenso wie Mayhew ist die frühzeitige Miteinbeziehung von Test-Benutzern für Nielsen unabdingbar. Er warnt jedoch davor, sich ausschließlich auf Befragungen von Test-Benutzern zu beschränken. Aus einer Studie gehe hervor, dass die meisten Benutzer nur anhand der Beschreibung einer Funktion oft den Nutzen nicht richtig beurteilen können.

**Koordiniertes Design der Gesamtoberfläche** Die Konsistenz des Designs sollte durch ein Komitee oder eine einzelne Person sichergestellt werden. Dabei ist es sinnvoll, die zur Verfügung stehenden technischen Möglichkeiten wie z. B. das Verwenden gemeinsamer Bibliotheken und Tools oder die Wiederverwendung bestehenden Codes auszunutzen.

**Prototyping** Mehr dazu im Abschnitt „Prototyping“ (6.4)

**Anwenden von Gestaltungsrichtlinien** Mehr dazu im Abschnitt „Gestaltungsrichtlinien“ (6.5)

**Empirisches Testen** Nielsen weist insbesondere darauf hin, dass die Relevanz von Usability-Problemen wenigstens 2-dimensional ist: zum einen die Schwere eines Problems und zum anderen die Anzahl der von diesem Problem betroffenen Benutzer. [siehe Seminarthema 7: „Usability Labore und Tests“]

**Iteratives Vorgehen** Nielsen hebt als besonders wichtig hervor, dass die Test-Benutzer beim iterativen Testen und Verbessern der Designs nicht selbst am (participatory) Design beteiligt gewesen sein dürfen, da diese das Design kennen und nicht unvoreingenommen agieren können.

## 6.3 Phase 3: Installation

Nach Abschluss der Designphase und der Implementierung der Software wird diese (beim Auftraggeber) installiert und zur Benutzung freigegeben. Als Entwickler ist

man natürlich daran interessiert, wie gut das Produkt vom Benutzer angenommen und benutzt werden kann. Es sollte also nach der Installation Benutzer-Feedback gesammelt werden. Im Idealfall sind die Kundenzufriedenheit und die wirtschaftlichen Auswirkungen des Usability-Engineering direkt messbar und man kann diese analysieren. Bspw. lässt sich dies bei Web-basierten Projekten dadurch erreichen, dass man Seitenabrufe oder die Benutzung bestimmter Funktionen in Log-Files protokolliert. Anhand derer lässt sich dann das Benutzerverhalten genau auswerten und das daraus gewonnene Wissen kann in zukünftige Entwicklungen einfließen. Für nächste Produktversionen oder oft auch neue Entwicklungen lässt sich das aktuelle Release hervorragend als Prototyp verwenden, ganz nach dem Spruch in [HeV03, S. 72]:

> Nach dem Release ist vor dem Release.

## 6.4 Prototyping

Während des Entwicklungszyklus wird der jeweilige Stand der Designentwicklung mit Hilfe von Prototypen evaluiert. Die Verwendung von Prototypen ist in klassischen Industriezweigen Gang und Gäbe. Anhand von mehr oder weniger funktionstüchtigen Versuchsmodellen lassen sich schon früh Planungsfehler und Verbesserungsanregungen finden. In der Automobilindustrie wird ein erstes Karosserie-Modell zunächst aus Ton modelliert, um einen Eindruck des Erscheinungsbildes zu gewinnen. Da man nicht für jeden neuen Entwicklungsstand die volle Funktionalität benötigt, um einzelne Änderungen und Fortschritte zu testen, sind Prototypen ein probates und kostengünstiges Mittel, um schnell ein sinnvolles Versuchsmodell für Testzwecke zu erhalten.

Welche Möglichkeiten gibt es nun, um auch im Usability-Engineering Prototypen einzusetzen?

**Skizzen (engl. paper mock-ups)** Verschiedene Oberflächen werden auf Papier aufgezeichnet. Dies ist besonders schnell durchführbar, so dass sogar während des Testens Änderungen am Design vorgenommen werden können. Dem Test-Benutzer wird beim Test das auf seine jeweilige Aktionen folgende Papier vorgelegt. Dadurch wird eine Interaktivität simuliert, die es bei einem statischen Prototyp natürlich nicht geben kann. [siehe Seminarthema 5: „Paper Prototyping“]

**Rapid Prototyping** Durch graphische Entwurfsumgebungen lassen sich einfach und schnell Oberflächen mit beschränkter Funktionalität erstellen, die dem endgültigen Produkt schon sehr nahe kommen. Somit kann dem Benutzer schon früh „greifbar“ demonstriert werden, wie das fertige Produkt in etwa aussehen soll.

**Interactive Prototyping** Eine flexible Oberflächenstruktur und fähige Programmierer ermöglichen es, interaktiv (d. h. während der Tests der Oberfläche) Änderungen auf Vorschlag der Test-Benutzer oder der Designer zu verwirklichen und die Oberfläche zu modifizieren. [Nie93, S. 93], [Jan01, S. 78]

Prototypen lassen sich in zwei Richtungen gestalten: horizontal (beschränkt in Funktionalität) und vertikal (beschränkt in Anzahl der Funktionen).

*Horizontale Prototypen* gehen in die Breite und bieten alle Funktionen, die das endgültige Produkt haben soll. Jedoch ist die Funktionalität noch beschränkt bzw. sogar noch gar nicht vorhanden. Anhand von horizontalen Prototypen lässt sich schnell ein Überblick über den Funktionsumfang gewinnen.

*Vertikale Prototypen* hingegen bieten für wenige Funktionen die vollständige Funktionalität. Diese Prototypen beleuchten spezielle Aspekte des Systems in aller Tiefe, so dass Test-Benutzer einzelne Aufgaben sehr detailliert testen können. Abbildung 5 (angelehnt an [Nie93, S. 94]) verdeutlicht die Beziehung der beiden Typen.

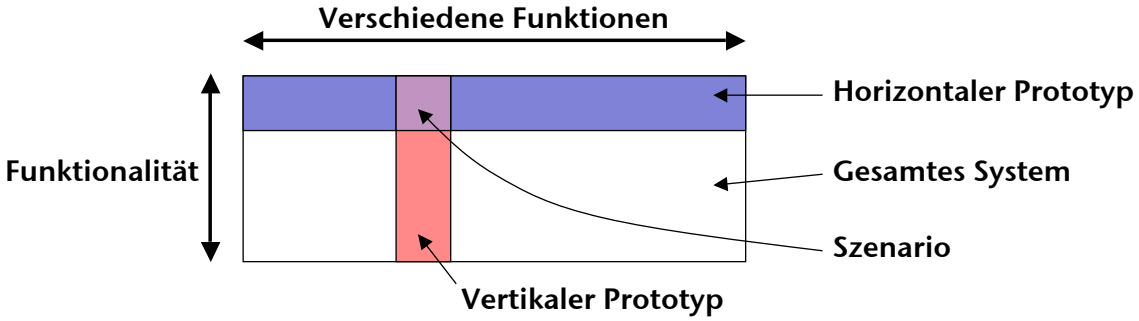


**Abbildung 5:** Schematische Darstellung von horizontalen und vertikalen Prototypen

## 6.5 Gestaltungsrichtlinien

Wie schon in Abschnitt 6.2 bemerkt, zieht sich das Erstellen, Erweitern und Befolgen von Gestaltungsrichtlinien im Prozessmodell Mayhews als roter Faden durch den Entwicklungszyklus. Auch Nielsen beschreibt die Notwendigkeit der Miteinbeziehung von Gestaltungsrichtlinien.

Man kann drei wesentliche Ausprägungen unterscheiden:

**generelle Richtlinien** Richtlinien für alle Oberflächen, wie sie bspw. durch Normen (siehe Abschnitt 5.1) beschrieben werden

**Kategorie-spezifische Richtlinien** Richtlinien für speziellere Anwendungsgebiete, wie z. B. Gestaltungsrichtlinien für Betriebssysteme

**Produkt-spezifische Richtlinien** Richtlinien für ein bestimmtes Software-Produkt, wie sie z. B. vom Software-Unternehmen selbst erstellt werden [Nie93, S. 91]

Sie enthalten Informationen über die Verwendung von Gestaltungs- und Bedienelementen, Strategien zu ihrem Einsatz und der Positionierung und weitere Konsistenz-Kriterien. Sie werden während des Entwicklungszyklus vom Entwicklungs-Team erstellt und dienen als Schnittstelle zwischen Aufgaben- und Benutzeranalyse, Entwicklung und Evaluation von Benutzungsoberflächen. [Jan01, S. 79f.], [May99, S. 7]

Produkt-spezifische Gestaltungsrichtlinien gewährleisten einheitliche Oberflächen bei großen Entwickler-Teams. Zudem ist bei Inkonsistenzen eine konkrete Angabe der missachteten Richtlinie möglich, wodurch die Fehlerbehebung einfacher und zumeist eindeutig ist.

## 6.6 Usability-Engineering und Object Oriented Software Engineering

Das zentrale Konzept des iterativen Vorgehens beim Object Oriented Software Engineering (OOSE) stimmt mit der iterativen Entwicklung von Benutzeroberflächen im Usability-Engineering überein. OOSE, erstmalig von Ivor Jacobson et. al. 1992 beschrieben, ist ebenso wie Mayhews Prozessmodell in drei Phasen unterteilt. OOSE ist – wie zuvor erwähnt – überaus kompatibel zum Usability-Engineering, so dass beide Verfahren parallel ausführbar sind. Dieses Vorgehen ist besonders zu empfehlen – Deborah Mayhew schreibt in [May99, S. 30]:

> Usability Engineering and OOSE are not competing approaches, but rather are potentially complementary approaches to software development.

Des Weiteren führt sie in [May99, S. 27f.] aus, welche Schritte des Usability-Engineering welchen Schritten der OOSE zugeordnet werden können. Dies ist in Tabelle 1 gegenüber gestellt:

Allerdings werden im OOSE viele grundlegende Ziele des Usability-Engineering nicht behandelt. Beispielsweise wird im OOSE auf die Analyse der Arbeitsumgebung nicht ausdrücklich eingegangen. Durch OOSE ohne die Methoden des Usability-Engineering wird daher nicht unbedingt gebrauchstaugliche Software entstehen. Mayhew betont in [May99, S. 30] deshalb die Notwendigkeit der Usability-Engineering-Maßnahmen:

| OOSE | | Usability Engineering Lifecycle | |
|---|---|---|---|
| 1.1) | Anforderungsmodell | 1) | Benutzerprofil erstellen |
| 1.2) | Analysemodell | 2.1) | abstrakte Ideen, Reengineering der Arbeit |
| 2.1) | Designmodell | 2.2) | Screen-Design unter Berücksichtigung von Gestaltungsrichtlinien |
| 2.2) | Implementierungsmodell | 2.3) | detailliertes Design, iterative Verbesserung |
| 3) | Testmodell | 2.3/3) | iterative Evaluation; Feedback |

**Tabelle 1:** Gegenüberstellung OOSE und „Usability Engineering Lifecycle“ von Mayhew

> The Usability Engineering techniques [...] can be thought of as extensions to OOSE that will greatly enhance the ultimate usability of object-oriented software systems.

# 7 Resümee

## 7.1 „Todsünden“ und übliche Fehler beim Usability-Engineering

Die Ein- und Durchführung von Usability-Engineering läuft nicht immer problemlos ab. Selbst wenn man einem benutzerorientierten Usability-Prozessmodell wie bspw. dem von Mayhew oder Nielsen zu folgen versucht, bedeutet das nicht automatisch, dass das Ergebnis wirklich ein gebrauchstaugliches Produkt ist. Es gibt verschiedene „Stolpersteine“, die Kalbach in [HeV03, S. 75f.] beschreibt und die das praktische Durchführen erschweren:

### Fehlende Unterstützung im Management

Es reicht nicht aus, wenn eine einzelne Person sich für Usability-Maßnahmen verantwortlich fühlt. Das gesamte Team muss „eine benutzerfreundliche Sicht verinnerlichen“. Insbesondere ist natürlich essentiell, dass die Entscheidungsträger von der Wichtigkeit der Usability-basierten Entwicklung überzeugt sind und die nötigen Maßnahmen fördern. Also muss Benutzerfreundlichkeit „ein zentrales Ziel von Management und Team sein, um zum Erfolg zu führen.“

### Mangelnde Überzeugungskraft von Usability-Experten

Es ist am Usability-Experten, den Gedanken der benutzerfreundlichen Entwicklung im Entwicklungsteam zu verbreiten. Deshalb ist reines Fachwissen im Bereich Usability-Engineering nicht ausreichend. Usability-Spezialisten „müssen Nutzer-orientiertes Design vertreten und evangelisieren“. Der Erfolg des Usability-Engineerings hängt also zu einem großen Teil davon ab, wie gut der Usability-Experte die Augen der Team-Mitglieder für eine benutzerorientierte Vorgehensweise öffnen und ob er „eine Benutzer-orientierte Denkweise mit Überzeugung und Zuversicht ins Team tragen kann“.

### Kosten-Nutzen-Rechnung oft nicht gemacht

Usability-Experten kommen meist nicht aus dem Bereich Betriebswirtschaft. So fällt es ihnen manchmal schwer, das Management von den Vorteilen und der Wichtigkeit von Usability zu überzeugen. Dabei spielt eine wichtige Rolle klar aufzuzeigen, dass der „Return on investment“ von Usability-basierter Software höher ist als bei Software ohne Usability-Engineering. „Für jedes Projekt empfiehlt sich eine Usability-Benefit-Analyse“, so dass den entscheidenden Managern die wirtschaftliche Bedeutung von Usability verdeutlicht und die Benutzerfreundlichkeit „in die Unternehmens-Ziele integriert“ wird.

### Durchführung zu spät

Oft werden Usability-Maßnahmen erst gegen Ende der Entwicklung durchgeführt. Zu diesem Zeitpunkt können sie allerdings nicht mehr als eine Standortbestimmung sein. Um grundlegende Designschwächen, die bei diesen späten Tests gefunden werden, noch ausbessern zu können, ist dann meist keine Zeit mehr. Damit Usability-Maßnahmen ihre „maximale Wirkung entfalten können“, müssen sie „in der Konzeptionsphase oder sogar schon früher eingesetzt werden“. Anstatt also bis zum Abschluss der Entwicklung zu warten, um dann noch schnell Usability in das Produkt „hineinzutesten“, ist nach der Regel zu verfahren: „Usability: So früh wie möglich beginnen“.

### Unklare Entscheidungsstruktur für Usablity-Fragen und bei Design-Entscheidungen

Gibt es in einem Unternehmen keine klare Rollenverteilung bzgl. Entscheidungsfällung, kann es vorkommen, dass verschiedene Designer mit unterschiedlichen Ansich-

ten einen Streit beginnen, welche der Designideen die bessere sei. Dabei werden dann häufig egoistische Gründe verfolgt. „Der Verlierer ist dann oft der Nutzer“, der dabei vollkommen aus der Sicht verloren wird. Es ist also unbedingt notwendig, dass eine Entscheidungsstruktur existiert und vor allem klar ist, „wie Design-Entscheidungen getroffen werden“.

### Keine Spezifizierung der Usability-Ziele

Da es oft nicht einfach ist, greif- und überprüfbare Ziele festzulegen, wird dies oft komplett ausgelassen. Allerdings ist es nur bei klarer Spezifizierung der Usability-Ziele möglich, eine objektive Bewertung während der Evaluations-Schritte des Entwicklungszyklus zu gewährleisten, anhand derer man über die Notwendigkeit eines weiteren Verbesserungsschrittes entscheiden kann. „Die wichtigsten Metriken sollten aufgezeigt und in messbarer Form formuliert“ und in die Produkt-Spezifikation mit aufgenommen werden.

### Unklare Definition der Nutzer

Wenn den Entwicklern und Designern nicht genau klar ist, für welche Zielgruppe von Benutzern sie das Produkt entwickeln, so ist es ihnen so gut wie unmöglich, dieses für die Zielgruppe gebrauchstauglich zu gestalten. Es ist notwendig, den typischen Benutzer der Software zu beschreiben, „am besten durch die Erstellung von User-Profilen“. Nur so können „Usability-Probleme und -Fehler“ von Designern und Entwicklern „richtig interpretiert und bewertet werden“.

### Technik/Feature-itis dominieren

Stehen die Technologie oder bestimmte, meist in Brainstorming-Sessions erdachte, Features im Mittelpunkt der Entwicklung, werden oft die tatsächlichen Bedürfnisse des Benutzers außer Acht gelassen. Grade die oft phantasievollen Features, an denen festgehalten wird, obwohl ihre tatsächliche Nützlichkeit minimal ist, führen oft dazu, dass „die Benutzerfreundlichkeit in den Hintergrund“ tritt. In solch einem Fall ist es Aufgabe des Usability-Experten, dem Team seine „Scheuklappen“ abzunehmen und „immer wieder die Nutzungsperspektive als wichtiges Kriterium in die Waagschale“ zu werfen. Er muss während des gesamten Entwicklungszeitraums darauf achten, dass die Entwicklung von allen Perspektiven aus betrachtet wird – „von der Technik über das Design bis hin zur Usability“.

## 7.2 Abschließende Bemerkungen

Bis zur breiten Anerkennung der Usability war es ein weiter Weg. Erstaunlich, wenn man bedenkt, dass die grundlegenden Usability-Ideen schon frühzeitig entstanden sind. Gould und Lewis beschrieben schon 1985 die benutzerorientierte und iterative Vorgehensweise. In den Xerox PARCs und bei Apple wurden Anfang der 80er GUIs in Usability-Laboren entwickelt. Trotzdem dauerte es von da ab noch viele Jahre, bis das Thema „Usability“ von der Mehrheit der Software-Entwickler und vor allem der Geldgeber als wichtig akzeptiert wurde. Insbesondere das Aufkommen immer weiter verbreiteterer Gadgets wie Handy, PDA und mp3-Player mit ihrem immensen Absatz sorgte dafür, dass Usability sehr wohl als Massenmarkt anerkannt wurde.

Nach wie vor ist Usability eine neue Disziplin, die inzwischen durchaus ernsthafte und professionelle Bemühungen hervorbringt. Seit 2002 existiert bspw. ein Berufsverband für Usability-Professionals (German Chapter der Usability Professionals' Association, GC-UPA). Das GC-UPA „unterstützt Usability Profis in Fragen der praktischen Umsetzung von Methoden und Verfahren im Arbeitsalltag. Es knüpft ein Netzwerk zum Erfahrungsaustausch und fördert die öffentliche Meinungsbildung zum Thema 'Usability'.“ [GCU04], [HeV03, S. 2]

Des weiteren findet jährlich die Konferenzreihe „Mensch & Computer“ statt (dieses Jahr in Paderborn!). Man sieht sich selbst als „eine Anstrengung, um verschiedene Fachgebiete und Praxisfelder in einen fruchtbaren Diskurs zu bringen und um voneinander zum Nutzen möglichst vieler Menschen zu lernen.“ [MuC04]

Vogt und Heinsen schreiben in [HeV03, S. 6]:

> Benutzerfreundlichkeit ist kein leichtes Ziel. Aber ein erreichbares. Mit der richtigen Ausrüstung und der nötigen Vorsicht und dem festen Willen kann auch der 8000er-Berg Usability erklommen werden. Wir glauben, dass aber noch viel zu tun ist. [...] Packen wir es an!

# Abbildungsverzeichnis

1 Kosten/Nutzen von Usability-Maßnahmen nach Mantei und Teorey . 13
2 Prozessmodell nach ISO 13407 . . . . . 17
3 Der „Usability Engineering Lifecycle“ nach Mayhew [May99, Einband] 20
4 „Usability-Ziel-Linie“ nach Nielsen . . . . . 21
5 Schematische Darstellung von horizontalen und vertikalen Prototypen 25

# Tabellenverzeichnis

1 Gegenüberstellung OOSE und „Usability Engineering Lifecycle“ von Mayhew . . . . . 27

# Literatur

Die Angabe verwendeter Literatur erfolgt in nach den vergebenen Kürzeln sortierter Reihenfolge. Hinter jedem Eintrag ist eine Liste der Seiten angegeben, von denen ein Verweis erfolgte.


[Bat99] Battey, Jim. *IBM's redesign results in a kinder, simpler Web site.* InfoWord, 19. April 1999. (Online verfügbar unter http://interface.free.fr/Archives/IBM_redesign_results.pdf, Aufruf: 18.06.2004) 9

[BCW97] Bloomer, Sarah und Croft, Rachel und Wright, Lloyd. *Collaborative design workshops: a case study.* ACM Press, Interactions, Volume 4, Issue 1, S. 31–39, Jan./Feb. 1997. http://doi.acm.org/10.1145/242388.242401. 10

[GCU04] German Chapter der UPA e. V. *gc-UPA.* 2004. http://www.gc-upa.de/, Aufruf: 18.06.2004 30

[GoL85] Gould, John D. und Lewis, Clayton. *Designing for usability: key principles and what designers think.* ACM Press, Commun. ACM, Volume 28, Number 3, S. 300–311, 1985. http://doi.acm.org/10.1145/3166.3170. 5

[HaK94] Haunold, Peter und Kuhn, Werner. *A keystroke level analysis of a graphics application: manual map digitizing.* ACM Press, Proceedings of the SIGCHI conference on Human factors in computing systems, S. 337–343, 1994. http://doi.acm.org/10.1145/191666.191779. 12

[HeV03] Heinsen, Sven und Vogt, Petra. *Usability praktisch umsetzen.* Hanser Verlag, München, 2003. 6, 7, 10, 11, 14, 17, 24, 27, 30

[Jan01] Janson, André. *Usability-Engineering als Instrument des Managements informationstechnologischer Veränderungsprozesse in Unternehmen* Inaugural-Dissertation zur Erlangung des Grades eines doctor rerum politicarum (Dr. rer. pol.) der Fakultaet Sozial- und Wirtschaftswissenschaften der Otto-Friedrich-Universitaet Bamberg, 2001. http://elib.uni-bamberg.de/volltexte/2001/5.html. 18, 25, 26

[JIMK03] Jokela, Timo und Iivari, Netta und Matero, Juha und Karukka, Minna. *The standard of user-centered design and the standard definition of usability: analyzing ISO 13407 against ISO 9241-11.* ACM Press, Proceedings of the Latin American conference on Human-computer interaction, S. 53–60, 2003. 16, 17

[Kal99] Kalin, Sari. *Mazed and confused.* CIO Web Business Magazine, 11. April 1999. (Online verfügbar unter http://www.cio.com/archive/webbusiness/040199_use.html, Aufruf: 18.06.2004) 12

[MaB94] Mayhew, Deborah J. und Bias, Randolph G. *Cost-Justifying Usability.* Morgan Kaufmann, San Francisco, 1994. 11

[MaT88] Mantei, Marilyn M. und Teorey, Toby J. *Cost/benefit analysis for incorporating human factors in the software lifecycle.* ACM Press, Commun. ACM, Volume 31, Number 4, S. 428–439, 1988. http://doi.acm.org/10.1145/42404.42408. 12

[May99] Mayhew, Deborah J. *The Usability Engineering Lifecycle.* Morgan Kaufmann, San Francisco, 1999. 5, 6, 11, 18, 20, 21, 26, 31

[MuC04] Konferenzreihe Mensch & Computer. *Mensch & Computer.* 2004. http://www.mensch-und-computer.de/, Aufruf: 18.06.2004 30

[MyR92] Myers, Brad A. und Rosson, Mary Beth. *Survey on user interface programming.* ACM Press, Proceedings of the SIGCHI conference on Human factors in computing systems, S. 195–202, 1992. http://doi.acm.org/10.1145/142750.142789. 11

[Nie93] Nielsen, Jacob. *Usability Engineering.* Morgan Kaufmann, San Francisco, 1993. 4, 12, 14, 18, 21, 25, 26

[Nie97] Nielsen, Jacob. *Alertbox: Jakob Nielsen's Column on Web Usability.* 1997. http://www.useit.com/alertbox/, Aufruf: 18.06.2004.

[Nie97-1] Nielsen, Jacob. *Loyalty on the Web.* 1997. http://www.useit.com/alertbox/9708a.html, Aufruf: 18.06.2004. 12

[Nor02] Norman, Donald. *Emotion and Design: Attractive Things Work Better.* 2002. http://www.jnd.org/dn.mss/Emotion-and-design.html, Aufruf: 18.06.2004. 8

[Usa04] UsabilityNet (project funded by the European Union). 2003. http://www.usabilitynet.org/, Aufruf: 18.06.2004.

[Usa04-1] UsabilityNet. *Business Case.* 2003. http://www.usabilitynet.org/management/c_cost.htm, Aufruf: 18.06.2004. 9