

Kosten und Nutzen benutzerfreundlicher Software

Heike Wagner

Studienarbeit

Dezember 2000

Fachbereich Informatik
Universität Hamburg

Betreuer: Prof. Dr. Horst Oberquelle

Inhaltsverzeichnis

1	Einleitung	5
2	Softwarehersteller	9
2.1	Überlegungen vor Projektbeginn	9
2.1.1	Voruntersuchung	11
2.1.2	Projektplanung	13
2.1.3	Betriebswirtschaftliche Überlegungen und Methoden	18
2.1.4	Das Produktivitätsparadoxon	23
2.1.5	Organisationsstruktur und soziale Aspekte	25
2.1.6	Open-Source-Software	26
2.1.7	Werkzeuge zur Rechtfertigung der Kosten	27
2.2	Einmalige Kosten	28
2.2.1	Einrichtung eines Testlabors	30
2.2.2	Arbeitsanalyse des Anwendungsfeldes	30
2.2.3	Erstellung eines Prototypen	33
2.2.4	Einrichtung und Einarbeitung	34
2.2.5	Sonstige Kosten	35
2.3	Laufende Kosten	35
2.3.1	Personal	35
2.3.2	Durchführung	35
2.3.3	Support	37
2.3.4	Sonstige Kosten	38
2.4	Aktuelle Anwendung: Internetangebote	38
3	Softwarenutzer	43
3.1	Einführung	43
3.1.1	Installation und Systemumstellung	44
3.1.2	Organisationsveränderungen	44
3.1.3	Schulung	44
3.1.4	Verkauf	45
3.2	Benutzung	45
3.2.1	Support	46
3.3	Rechtliche Rahmenbedingungen	46
4	Zusammenfassung und Ausblick	49
A	Literaturverzeichnis	51

Kapitel 1

Einleitung

Viele Benutzer von Software ärgern sich über unbenutzbare Programme, die ihnen die Arbeit erschweren, statt sie bei dieser zu unterstützen. Es stellt sich die Frage, wie es kommt, daß viele Benutzer so offensichtliche Probleme mit der Benutzbarkeit der Programme haben. Liegt es daran, daß sich die Entwickler mit diesem Thema einfach nicht befassen oder sind vielleicht die Kosten zu hoch im Vergleich zum erwarteten Gewinn?

Früher waren Programmierer auch gleichzeitig Anwender. In dieser Situation entstanden natürlich kaum Benutzbarkeitsprobleme. Heutzutage werden Programme von Entwicklern geschrieben und von Anwendern benutzt. Dies führt bei den Anwendern zu Problemen mit der Benutzbarkeit. Dem Entwickler ist die Funktionsweise des Programms, z.B. die Menüführung bekannt, so hat er natürlich keine Probleme mit der Benutzung. '[...] why are developers so quick to claim to be user interface experts?' fragen [Bias und Mayhew 1994c]. 'The answer is quite simple: All developers are also users.' (S. 319) Der Anwender allerdings kennt das Programm nicht und ist bei der Anwendung des Programms gezwungen, die Gedanken des Entwicklers nachzuvollziehen. In dem Einsatzfeld der Software sind die Anwender Experten, die Entwickler aber sind es nicht. So wird es einem Entwickler relativ schwer fallen, für ein ihm fremdes Anwendungsgebiet eine gute Software zu entwickeln, ohne daß er sich mit diesem Anwendungsgebiet auseinandergesetzt hat. [Norman 1989] drückt es so aus: 'In their work, designers often become experts with the *device* they are designing. Users are often experts at the *task* they are trying to perform with the device.' (S. 156) Auf der anderen Seite haben die Entwickler ihr eigenes Fachgebiet, das für die Anwender fremd ist: 'The guys who are designing most of these technological products are such techies that they think it's natural for everybody to hold down four buttons and twiddle a knob at the same time.' ([Nussbaum und Neff 1991], S. 38) Ein Beispiel von [Mauro 1994] zeigt, daß Entwickler oft an den Bedürfnissen der Anwender vorbei entwickeln: Ein Nähmaschinenhersteller bot seine neuen Geräte an. Aber viele Kunden benutzten ihre alte Maschine, wenn sie etwas schnell machen wollten. Nach vielen Stunden benutzen sie nur noch ihre alte Maschine. Der Fehler, der bei der Entwicklung entstand, war der, daß häufig benutzte Operationen nicht vereinfacht wurden. Außerdem wurde die Maschine schnell durch falsche Handhabung derart blockiert, daß sie ausfiel. Ein Redesign konnte 90 % der Probleme lösen.

Entwickler haben oft kein Verständnis für die bei Anwendern auftretenden Probleme. Chapanis stellt dazu folgende Einstellung bei Entwicklern fest: 'I can use the system, and if I can use it, it's obviously easy to use' ([Chapanis 1991b], S. 359). Er vermutet zudem, daß Benutzbarkeits-Evaluationen nicht durchgeführt werden, weil Benutzbarkeit schwer zu messen

ist und weil wenige wissen, wie sie dabei vorgehen können.

Seit einigen Jahren machen sich Software-Hersteller mehr und mehr Gedanken um dieses Problem, sei es nun durch die Erkenntnis des Problems an sich oder durch einen Anreiz von außen, z.B. durch das Management, das eine bessere Benutzbarkeit des Produkts als Ziel gesetzt hat. Die Akzeptanz software-ergonomischer Prinzipien ist laut [Ehrlich und Rohn 1994] und [Nielsen 1994a] in Unternehmen sehr verschieden. In vielen Firmen, darunter Sun ([Sun 1999b]), setzt sich die Ansicht, daß Benutzbarkeit unverzichtbar ist, in steigendem Maße durch.

In vielen Software herstellenden Unternehmen herrscht noch immer die Ansicht vor, die Einbindung benutzerfreundlicher Aspekte sei schlichtweg eine Verschwendung von Ressourcen und damit zu teuer. Zudem wird häufig angenommen, der Nutzen lasse sich nicht abschätzen. Hier werden von vielen Firmen noch Einsparungsmöglichkeiten gesehen. Die Ausgangssituation ist in Unternehmen oft nicht besonders günstig für gute Software-Ergonomie. Wünschenswert ist eine Unternehmenskultur, in der ein Austausch über das Thema Software-Ergonomie ausdrücklich gewünscht wird.

[Ehrlich und Rohn 1994] benutzen den Begriff *user centered design*¹. Dies soll ausdrücken, daß die Entwickler sich stärker mit dem späteren Anwender des Produkts befassen sollen. Der Vorteil des UCD liegt darin, daß schon frühzeitig im Entwicklungsprozeß auf die Bedürfnisse des Anwenders eingegangen werden kann. Je früher Probleme erkannt werden, desto größer ist die Kostenersparnis ([Pressman 1992]). Wird der Benutzer gefragt, was er sich wünscht, kann man dies bei der Entwicklung berücksichtigen, der Benutzer ist zufriedener und der Hersteller muß weniger nachbessern. Außerdem lernt der Entwickler das Anwendungsfeld seiner Software kennen und kann so angepaßte entwickeln, insbesondere kann er Design-Entscheidungen leichter treffen, wenn er weiß, wie die Arbeit der künftigen Anwender von statten geht. Daß die Arbeitsaufgabe in einem engen Zusammenhang mit der Benutzbarkeit steht, sieht man an dem Inhalt der [Bildschirmarbeitsverordnung 1996] und auch an folgenden Definitionen von Benutzbarkeit:

- [Chapanis 1991b], S. 361: 'The usability of a computer is measured by how easily and how effectively the computer can be used by a specific set of users, given particular kinds of support, to carry out a fixed set of tasks, in a defined set of environments.'
- [Corbett, Macleod und Kelly 1993], S. 314: 'Usability is the extend to which a product can be used with efficiency and satisfaction by specific users to achieve specific goals in specific environments.'

Benutzbarkeit besagt also, wie effizient und zufriedenstellend ein Benutzer mit dem Produkt umgehen kann, wenn eine bestimmte Arbeitsaufgabe vorliegt.

Ein recht junger externer Anlaß, sich mit Benutzbarkeit zu beschäftigen, besteht in der Einführung von rechtlichen Grundlagen auf diesem Gebiet, z.B. der Bildschirmarbeitsverordnung. Unternehmen befürchten auch Rechtsstreitereien wegen gesundheitlicher Dauerschäden der Mitarbeiter, z.B. aufgrund von besonderer Beanspruchung am Arbeitsplatz. Dies wird sich in Zukunft nicht nur auf die physikalische Arbeitsumgebung wie Bürostuhl oder Monitor beziehen, sondern auch auf den Einsatz von Software. Diese Art Kostenquelle ist Entwicklern und auch Managern noch recht unbekannt. Dies rührt sicher nicht zuletzt von der Art der Ausbildung her, die Programmierern zwar das technische Know-How zur Erstellung von Programmen vermittelt, nicht aber die zugehörigen Rahmenbedingungen lehrt, z.B. die Analyse des Aufgabenkontextes,

¹benutzerorientierte Systementwicklung(UCD)

in dem eine Software später eingesetzt wird. Aber auch ohne gesetzliche Vorschriften ist das Bewußtsein der Anwender für gute Benutzbarkeit heute stärker ausgeprägt. [Snyder 1991], zitiert nach [Ehrlich und Rohn 1994], fand in einer Befragung von 500 Geschäftsleuten, daß die Benutzbarkeit eines Softwareprodukts als wichtigstes Merkmal für Zufriedenheit angesehen wurde. [Nielsen, Mack, Bergendorff und Grischowsky 1986], zitiert nach [Ehrlich und Rohn 1994], beschreiben sogar einen Fall, in dem Benutzer die Verwendung eines Programms verweigerten, weil dessen Bedienungsanleitung zu dick war.

In dieser Arbeit soll anhand des Software-Entwicklungszyklus² dargestellt werden, wie Kosten und Nutzen von Benutzbarkeit berechnet oder zumindest geschätzt werden können. Hierzu zählen sowohl die Kosten und Gewinne der Hersteller als auch die der Anwender. Die Rollen der Beteiligten – Hersteller, Arbeitgeber (die ihre Mitarbeiter Software benutzen lassen) und Anwender³ – werden betrachtet. Es soll jeweils ergründet werden, welche Einstellung in den einzelnen Personengruppen vorliegen und wie sich diese auf die Herstellung und den Einsatz von Software, und damit auf die Kosten auswirken. Da das Thema Benutzbarkeit heute viel diskutiert wird, sollten sich neuere Untersuchungen finden lassen, die sich mit diesem Thema auseinandersetzen. Diese Arbeit soll einen Überblick über den aktuellen Stand der Forschung geben, einige Beispiele darstellen und gegebenenfalls Defizite aufzeigen.

²nach [Grudin, Ehrlich und Shriner 1987], S. 127: User Requirements \mapsto Feasibility Study \mapsto Requirements Definition \mapsto Preliminary Design \mapsto Coding \mapsto Testing \mapsto Operation and Maintenance (\mapsto Product Retired)

³Dies können Angestellte in einem Unternehmen sein, die betriebs- oder arbeitsspezifische Software benutzen, oder auch Privatpersonen, die in ihrer Freizeit verschiedene Arten von Software benutzen, um eine bestimmte Aufgabe auszuführen, z.B. einen Brief zu schreiben. Nicht betrachtet wird hier Software, die der Unterhaltung dient, obwohl man einige hier dargestellte Punkte auf diese Art Software übertragen kann.

Kapitel 2

Softwarehersteller

2.1 Überlegungen vor Projektbeginn

Die Identifizierung der Kosten ist nach Meinung von Chapanis recht schwierig. ‘Often those costs are buried in generalized personnel or development costs that cannot be allocated to specific projects.’ ([Chapanis 1991a], S. 41). Idealerweise könnte man Kosten und Nutzen feststellen, wenn man zwei ähnliche Systeme betrachtet, eines, das unter Einbeziehung von Softwareergonomie entwickelt wurde und eines, das ohne besondere Berücksichtigung benutzerfreundlicher Aspekte entwickelt wurde. Dies ist jedoch ein eher wissenschaftlicher Ansatz. In der Praxis wird man dies jedoch so nur selten finden. Eine Untersuchung, die genau dies betrachtet, findet sich in [Bailey, Knox und Lynch 1988]. Dort wurden zwei Modelle verschiedener Serien von Meßgeräten untersucht. Es sollte herausgefunden werden, mit welchem Modell die Messungen schneller durchgeführt werden konnten. Als Ergebnis fand sich, daß das Modell, das mehr graphische Komponenten (wie Icons und Pop-Up-Menüs) aufwies, eine um 77% größere Produktivität bei Benutzung durch Experten des Anwendungsfeldes ergab. Leider findet sich in dieser Untersuchung kein Hinweis auf die Entwicklungskosten beider Systeme.

Chapanis weist darauf hin, daß die sogenannten Angaben zu Erfolgsgeschichten einer kritischen Betrachtung bedürfen, weil oft konkrete Angaben zu den Kosten fehlen oder weil die Kosten für die Studien zur Benutzerfreundlichkeit nicht mitgerechnet wurden. Es sei schwer, die Validität solcher Aussagen zu überprüfen, auch, weil es oft zuviele Einflußgrößen gebe und man nicht sagen könne, welche der Faktoren das Ergebnis produziert haben. ‘It’s easy enough to measure [...] cost savings’. What is hard is to prove with certainty what was responsible for those [...] savings. ([Chapanis 1991a], S. 43) Dennoch lohne sich die Überlegung, Benutzerfreundlichkeit in die Entwicklung zu integrieren, denn diese habe ein ‘advertising value’. ([Chapanis 1991a], S. 45) Chapanis fand in vielen Anzeigen Hinweise auf die Benutzerfreundlichkeit des beworbenen Produkts. In den Vereinigten Staaten gebe es sogar schon Urteile gegen Firmen, deren Computer nicht so einfach zu benutzen waren, wie die Hersteller es voraussagten. Das Problem mit experimentellen Studien besteht nach Chapanis darin, daß theoretische Alternativen und keine echten Produkte untersucht werden. Die Aufgaben seien oft nicht realistisch und Unterschiede würden nicht in Geldwert oder Produktivitätsgewinnen ausgedrückt. Zudem seien oft die Probanden nicht die wirklichen Benutzer. Das Nicht-Beachten von Benutzbarkeit kann allerdings Kosten verursachen. Chapanis beschreibt den Fall von IBM, die Tastaturen auf eigene Kosten austauschte, die nicht benutzbar waren.

Das Argument, Kosten nicht beziffern zu können, sollte nach Chapanis nicht dazu führen, diese Kosten zu mißachten. Konkrete Berechnungen eigneten sich für zwei Arten von Kostenersparnis: erstens diejenige, die sich durch reduzierte Trainingskosten ergibt und zweitens diejenige, die sich aus der Reduzierung der Anrufe bei einer Hotline ergibt. Chapanis ist der Meinung, die Beachtung software-ergonomischer Grundsätze lohne sich immer: '[...] a program of human factors improvements would still pay for itself at least ten times over. ([Chapanis 1991a], S. 70)

[Karat 1997] nennt die Vorteile, die sich durch den Einsatz von Software-Ergonomie geben:

- verbesserte Produktdefinition,
- verbessertes Produktdesign,
- erhöhte Produkt-Performanz,
- höhere Zufriedenheit der Benutzer,
- reduzierte Entwicklungszeit,
- reduzierte Entwicklungskosten,
- höhere Verkaufszahlen und Gewinne,
- reduzierte Trainings- und Help Desk-Kosten,
- reduzierte Wartungskosten,
- reduzierte Personalkosten,
- reduzierte Einrichtungskosten,
- erhöhte Benutzerproduktivität.

[Hackos und Redish 1998] geben einige Widerstände gegen Benutzbarkeitsstudien wieder (S.13):

1. Marketing already knows the users.
2. Produkt is new - there aren't any users to observe.
3. Our users are all too different - we can't possibly visit all of them.
4. We don't have enough time in the schedule.
5. We don't have enough money in the budget.
6. One of the members of the development team was a user for 25 years.
7. We really have some great design ideas already - we don't need users to question what we are doing.
8. Users don't know anything about design.
9. Designers should design, not users.
10. We are users ourselves - we know what we need.
11. We're reengineering the process anyway - what can the current user tell us?
12. We've never done user analysis - we don't know how to do it.

13. We've been holding focus groups and the users attend.
14. We did a survey of the users and asked them what they wanted.

Sie geben für die meisten dieser Annahmen Gegenbeispiele (S. 117 ff), z.B. zu Punkt 1): Selbst, wenn die gleiche Benutzergruppe untersucht wurde, bleibt immer noch die Frage, ob auch die gleiche Fragestellung untersucht wurde, eventuell besteht bei den Benutzern der bereits getätigten Studie z.B. ein anderes Vorwissen. Wird doch eine Studie angestrebt, wird vielleicht aus Kostengründen nur der kleine Ausschnitt, der neu gestaltet werden soll, untersucht. Daß das nicht ausreicht, belegen die Autoren mit dem Beispiel einer Datenbank, deren Abfrage neu gestaltet werden soll. Wenn hierbei nicht auch die Eingabe untersucht wird, kann es dazu kommen, daß in die Datenbank keine Eingaben erfolgen, weil die Eingabemaske zu kompliziert ist. Wenn in der Datenbank keine Daten enthalten sind, können allerdings auch keine Abfragen erfolgen. Zu Punkt 2): Angenommen, man versucht, als erster ein Fax-Gerät zu entwickeln, dann bietet es sich an, das Briefeschreiben zu analysieren.

Zu Punkt 5): [Nielsen und Landauer 1993] stellen fest, daß 3–4 Benutzer ausreichen, um die größten Fehler (80%) zu finden. Auch [Rothman 1997] vertritt die Meinung, daß Benutzbarkeitstest auch mit wenig Mitteln durchgeführt werden sollten, weil dadurch immerhin die größten Fehler gefunden werden. Zum Vergleich: Eine repräsentative Marktforschungsstudie benötigt 120–150 Personen.

zu Punkt 11): In einem Beispiel sollte ein papierloses Büro geschaffen werden, aber Berichte wurden noch für andere Zwecke verwendet, jedoch waren sie nun nicht mehr ausdrückbar. Man erkennt, daß die o.g. Widerstände auf Vorurteilen beruhen, die sich sehr leicht entkräften lassen, wenn man sich nur einmal eingehend mit ihnen beschäftigt.

2.1.1 Voruntersuchung

Wie bei jedem Projekt werden natürlich zu Beginn die Projektziele festgelegt. Das oberste Ziel ist im allgemeinen die Erhöhung eines Gewinns. Dieses Ziel kann in Unterziele herunter gebrochen werden, in Ziele wie erhöhte Produktivität der Mitarbeiter bei Inhouse-Entwicklungen oder höheren Absatz bei Produkten, die für andere Unternehmen oder für den Massenmarkt entwickelt werden. Um diese Ziele noch genauer fassen zu können, wird meist eine Voruntersuchung durchgeführt.

Die Voruntersuchung kann mit einer Marktanalyse beginnen. Man untersucht das eigene Produkt und Konkurrenzprodukte, um einen geeigneten Ansatzpunkt, z.B. über die gewünschte Funktionalität, zu erhalten. Dabei sollte die Zielgruppe berücksichtigt werden. Man kann sich Fragestellungen verschiedenster Art überlegen, sich der Verbesserung eines Produktes zu nähern, von einfachen Such- oder Auswahlaufgaben bis hin zu komplexeren Aufgaben wie dem Erstellen eines Berichts. Je nach Art der Aufgabe können so Verbesserungen in Form von Zeiteinsparungen von einigen Sekunden bis zu mehreren Minuten erzielt werden. Diese Zeitersparnis kann man mit Hilfe des Gehalts eines Mitarbeiters in einen Geldwert umrechnen. [Mayhew und Mantei 1994] geben ein Beispiel: Dadurch, daß 250 Benutzer, die an 230 Tagen im Jahr jeweils 60 Masken verarbeiten, einen Stundenlohn von 25 US-Dollar bekommen und eine Sekunde Bearbeitungszeit weniger für eine Maske benötigen, ergibt sich z.B. eine jährliche Ersparnis von 23.958 US-Dollar.

[Mauro 1994] berichtet von einem Fall, bei dem zwei Firmen in Konkurrenz standen. Das Produkt der Firma B besaß eine bessere Benutzungsschnittstelle, so daß diese Firma immer mehr Marktanteile erwarb. Firma A führte eine Benutzbarkeitstudie durch mit dem Ergebnis, daß ihr Produkt trotz größerer Funktionalität für die Anwender einfach nicht benutzbar war. Das Produkt wurde umgestaltet und die Benutzbarkeit wurde als Werbebotschaft eingesetzt. Der Erfolg

Aufgabe	Alternativen	Differenz (in sec.)
Treffen eines Zieles auf dem Bildschirm	Maus vs. Tastatur	1
Finden eines Menüeintrags	Menühierarchie: Tiefe vs. Breite	3
Komplexe Dateimanipulation mit Lese-, Vergleichs- und Editierfunktion	Überlappende vs. geteilte Fenster	150

Tabelle 2.1: Beispiele der Performanz-Unterschiede ausgewählter Design-Alternativen verschiedener Studien. Entnommen aus [Mayhew und Mantei 1994], S. 30 ff.

war der Rückgewinn der Marktanteile nach einem Jahr. [Wixon und Jones 1991] fanden in einer Fallstudie eine 80 %ige Erhöhung der Einnahmen gegenüber der ersten Version eines Produkts, das ohne besondere Beachtung der Benutzbarkeit entwickelt wurde. Damit wurden die Erwartungen um 60 % übertroffen. Die Kunden gaben die bessere Benutzbarkeit als Kaufargument an.

Ein zu reichhaltiges Angebot an Funktionalität führt schnell zur Unübersichtlichkeit, wie bei vielen betriebswirtschaftlichen Produkten, z.B. Enterprise Resource Planning (ERP) ([Computer Zeitung 1999]). Im Rahmen einer Voruntersuchung kann man derartige Fehler vermeiden, indem man das Produkt evaluiert. [Chapanis 1991b] unterscheidet zwei Arten der Evaluation: Prototyping und experimentelle Evaluation¹. Prototyping benötige zwar viele Probanden, helfe allerdings bei der Elimination von Problemen und ermögliche das Testen verschiedener Versionen. Experimentelle Evaluation sei der elegantere Weg der Identifizierung von Problemen beim direkten Produktvergleich, allerdings zeige sie keine Alternativen auf. Welche Art der Evaluation man einsetzen sollte, hängt von der Ausgangslage ab, z.B. von der Verfügbarkeit von Probanden. Eine Evaluation durch Experten sei weniger geeignet, weil Experten keine repräsentativen Benutzer seien und sie nicht in einer realen Umgebung getestet werden. Jedoch könne man mit dieser Methode wenigstens die größten Fehler verhindern.

Eine kostengünstige Methode der Voruntersuchung stellen [Cox und Pendley 1994] vor. Sie haben dieses Verfahren bereits 1986/1987 in einem Pilotprojekt eingesetzt. Der Vorteil dieser Methode besteht darin, daß bereits Daten vorhanden sind und nicht erst unter Einsatz von Kosten erhoben werden müssen. Dieses Verfahren ermöglicht so in der Phase der Voruntersuchung kostengünstige und zudem realistische Abschätzung der zu behebenden Benutzerfehler. Dabei betonen die Autoren, daß Software-Ergonomien die Vorteile eines Software-Ergonomie-Projekts in Dollar und Cent angeben sollten, da dies eine 'familiar language in the business community' ist (S. 148). In dem sogenannten UPAR²-Prozeß werden neben der Analyse und Kategorisierung der Benutzerfehler auch gleichartige Produkte am Markt untersucht, um globale Ziele festzulegen. Das Analyse-Team sollte interdisziplinär aus den Gebieten Qualitätssicherung, Entwicklung, Produktplanung, Serviceplanung und Software-Ergonomie zusammengestellt sein, und die Mitarbeiter sollten Kommunikations- und Teamfähigkeiten besitzen. Die Methode zur Analyse der Benutzerfehler besteht aus vier Schritten. Im ersten Schritt werden Rohdaten ermittelt. Durch die Verwendung vorhandener Daten besteht gerade der Vorteil der Kostenreduzierung. Diese Daten brauchen jetzt nur noch codiert zu werden. Im nächsten Schritt werden die Daten

¹[Nielsen und Landauer 1993] betrachten beide Arten in ihrer Untersuchung als gleichwertig

²Usability Problems Analysed and Reported

in einem von einer Statistiksoftware lesbaren Format gespeichert. Im dritten Schritt werden die Daten analysiert. Die Aufgaben, in denen die meisten Fehler entstehen, sind zu identifizieren, um ihnen die höchste Priorität bei der Eliminierung zu geben. Unteraufgaben sollten gruppiert werden, damit nicht gleiche Probleme doppelt gelöst werden. Im letzten Schritt sollen die exakten Gründe für die Probleme erkannt werden, und es sollen Empfehlungen zu ihrer Behebung gegeben werden. Hierbei sollen sich die Teammitglieder von folgenden Fragen leiten lassen:

1. Was verursachte den Fehler?
2. Welche Veränderungen werden den Fehler eliminieren? (Neben dem Redesign sind hier auch andere Empfehlungen wie Automation oder andere Änderungen möglich.)
3. Welche Aktionen sind für die Änderungen notwendig? Es sollen ein spezieller Plan und die benötigte Ressourcen festgelegt werden.

Neben der bereits erwähnten Kostenersparnis besteht ein weiterer Vorteil dieses Verfahrens darin, daß man Daten direkt von den Benutzern seines Produkts erhebt. Die Servicekosten werden dadurch reduziert, daß die Probleme gezielt behoben werden und in einer Dokumentation nur ausgewählte Kapitel überarbeitet werden müssen.

Wichtig in diesem Zusammenhang ist auch, die Arbeitsweise der Benutzer zu analysieren. 'Zwei Drittel der ergonomischen Gesamtqualität beruht auf einem genauen Verständnis der Arbeitsaufgaben des Benutzers.' Diese Qualität beeinflusst die Gebrauchstauglichkeit der Software und sei somit 'eine zu kalkulierende betriebswirtschaftliche Größe'. ([Dzida 1999], S. 24). (vgl. Kapitel über die Arbeitsanalyse des Anwendungsfeldes)

2.1.2 Projektplanung

Bevor man mit der Anforderungsdefinition für ein Softwareprojekt beginnt, wird das gesamte Projekt geplant. Eine gute Projektplanung ist das A und O eines jeden Projekts. Um später gute Benutzbarkeit zu gewährleisten, sollten bereits zu Projektbeginn alle erforderlichen Ressourcen eingeplant werden. Annahmen über die Verkaufszahlen sollten sehr vorsichtig geschätzt werden. Langfristig überzeugt es das Management mehr, wenn sich später herausstellt, daß der Nutzen höher ist als erwartet. Zudem sollte man zur Schätzung der Gesamtkosten auf jeden Fall eine Lebensdauer festlegen.

Softwareherstellung wird meist in Projektarbeit durchgeführt, und daraus resultieren auch einige Probleme. Es beginnt schon mit der Zusammenstellung des Teams. Auch [Pressman 1992] stellt fest, daß Teamarbeit am produktivsten ist. [Ehrlich und Rohn 1994] bemerken, daß durch die wahllose Zusammenstellung der Projektteams in einem Unternehmen die Kommunikation und Kooperation zwischen den Teams häufig vermindert ist. Die Teammitglieder stammen meist aus verschiedenen Abteilungen und werden für jedes Projekt neu zu einer Gruppe formiert. Als Lösung schlagen die Autoren eine zentrale UCD-Abteilung vor, die projektübergreifend arbeitet, um z.B. einheitliche Produkte zu gewährleisten und das Teamwissen zu erhalten. Unternehmen durchlaufen immer kürzere Entwicklungszyklen, um marktfähig zu bleiben, so haben sie immer weniger Zeit, Designfehler zu korrigieren. Deshalb sollten Software-Ergonomen gleich zu Beginn an Projekten beteiligt werden. Erschwerend kommt hinzu, daß Teammitglieder oft innerhalb eines Zyklus wechseln. Dadurch kann Softwareergonomie-Wissen nicht von einem Projekt zum nächsten weitergegeben werden und Wissen geht schlichtweg verloren. Hierbei kann die Existenz eines UCD-Teams hilfreich sein. [Brown 1991] stellte dies ebenfalls fest. Er berichtet von einem Projekt, in dem in Kooperation mit Entwicklern eines Pharmakonzerns ein Prototyp entwickelt

wurde. Es habe sehr lange gedauert, bis ein gemeinsames Verständnis sowie eine gemeinsame Sprache und gemeinsame Ziele bestanden.

Oft besteht das Problem jedoch schon vorher. Benutzbarkeit findet keinen Platz in der Projektplanung, also werden auch keine Experten hierfür in das Team aufgenommen. Lediglich kurz vor Ende des Projektes werden in einigen Fällen Experten gebeten, das Produkt auf diesen Aspekt hin zu untersuchen. Der Berater für Benutzbarkeit [Johnson 2000] zitiert einen Kollegen, der diese Situation vergleicht mit 'putting lipstick on a bulldog' (S. 412). Zum Ende des Entwicklungszyklus können die eigentlichen Probleme nicht mehr behoben werden, nämlich die, die in einer falschen Projektplanung begründet liegen. Johnson hat in seiner Beratertätigkeit festgestellt, daß vielen Projektmanagern die Unterschiede zwischen einem Schnittstellendesigner, einem Grafikdesigner und einem Schnittstellenprogrammierer nicht bewußt sind. Kurz gesagt: ein Grafikdesigner entwirft Grafiken und Icons, die ein Programmierer nach Anweisung des Schnittstellendesigners in eine Oberfläche einbaut. Genauere Aufgaben der einzelnen Arbeitsgebiete finden sich in [Johnson 2000], S. 419. Eine Hauptaufgabe für den Schnittstellendesigner stellt die Arbeitsanalyse dar, mit der er ermitteln kann, wie die Aufgaben der Anwender besser unterstützt werden können³. Das oberste Ziel eines jeden Projektes, egal ob es um eine Entwicklung für den internen Gebrauch oder um den Einsatz eines externen Produkts geht, ist die Erhöhung der Produktivität der Mitarbeiter, z.B. die Verminderung der Kosten für Support, Training, Dokumentation und Service. [Martin und McClure 1983] (zitiert nach [Karat 1994]) stellten fest, daß in einem konkreten Fall 20–30 Milliarden US-Dollar für Wartung ausgegeben wurden. Es war hier also ein großer Handlungsbedarf zu erkennen. Benutzbarkeit findet in der Projektplanung allerdings oft keinen Platz. Das liegt zum einen daran, daß der Bedarf daran nicht erkannt wird, aber auch daran, daß den Verantwortlichen oft die Kriterien für die Bewertung des Kosten-Nutzen-Verhältnisses fehlen. Kosten-Daten sind oft nur schwer zugänglich, weil gerade diese sensiblen Daten von Firmen geheimgehalten werden. In der Literatur findet man häufig nur fiktive Fallbeispiele, die immerhin auf echten Fällen beruhen sollen. Zudem sind die Kosten schwierig zu berechnen, da die spärlich vorhandenen Daten nicht immer aktuell und die Berechnungen oft fehlerhaft sind.

Projektleiter stehen immer vor einem Problem: Wird ein Produkt zu spät ausgeliefert, ist die Konkurrenz möglicherweise im Vorteil und der Marktanteil wird nicht so hoch wie erwartet. Wird das Budget des Projekts überschritten, wird das Produkt möglicherweise zu teuer und es ergibt sich der gleiche Effekt. Die Beachtung software-ergonomischer Prinzipien früh im Entwicklungsprozeß führt jedoch zur Reduzierung der Kosten, weil während der späteren Implementierung weniger Änderungen nötig sind. ([Shneiderman 1998], S. 14)

[Lederer und Prasad 1992] fanden heraus, daß 63 % aller Softwareprojekte sowohl ihr Budget als auch ihren Zeitplan nicht einhalten, und zwar aus den folgenden Gründen:

- ständige Änderungswünsche der Kunden,
- bei der Implementierung übersehene Aufgaben,
- Kunden erkennen ihre eigenen Anforderungen nicht,
- Analyse, Kommunikation und Verständnis sind unzureichend.

[Ehrlich und Rohn 1994] schätzen sogar, daß nahezu 100 % aller Softwareprojekte weder Budget noch Zeitplan einhalten. [Bosert 1991] (zitiert nach [Ehrlich und Rohn 1994]) schätzt, daß

³Zu den häufigsten Aufgaben in seiner Tätigkeit zählt Johnson z. B. die Neuorganisation von Menühierarchien, die Reduzierung der Anzahl der Fenster und die Neuformulierung von Fehlermeldungen

der geplante Entwicklungszyklus um 33–50% durch Einsatz von Techniken zur Entwicklung benutzbarer Software reduziert werden kann.

Für eine erfolgreiche Projektplanung geben [Ehrlich und Rohn 1994] einige Hinweise. Sie haben beobachtet, daß es je nach Unternehmenskultur und Einsicht im Management einige Stunden bis Wochen Überzeugungsarbeit bedeuten kann, Software-Ergonomie in die Projektplanung von Anfang an zu integrieren. Problematisch wird es, wenn versucht wird, zum Ende des Projektes hin, 'noch eben schnell' software-ergonomische Teilprojekte zu integrieren, denn dann stehen weder personelle noch finanzielle Ressourcen zur Verfügung. Nebenbei stellen die Autoren fest, daß es unter den Software-Ergonomen zwei Tendenzen gibt. Die Vertreter der einen Richtung ziehen es vor, in die Breite zu gehen, d.h. möglichst alle Projekte mit wenigstens etwas software-ergonomischer Aufmerksamkeit zu bedenken. Die Vertreter der anderen Seite plädieren dafür, lieber weniger Projekte zu fördern, dafür aber mit mehr Aufwand und Erfolg im einzelnen Projekt, so daß die Ergebnisse dann später auf andere Projekte übertragen werden können.

[Nielsen 1993b] untersuchte 31 Budgets. Er fand heraus, daß im Schnitt 6% eines Budgets und 1.5 Personenjahre für Benutzbarkeit verplant waren, aber 10% und 2.3 Personenjahre letztendlich benötigt wurden. Nach [Martin und McClure 1983] (zitiert nach [Karat 1994]) waren 80% der Kosten nach Auslieferung eines Produkts auf nicht umgesetzte oder übersehene Kundenanforderungen zurückzuführen und nur 20% auf 'echte' Fehler, wie z.B. mangelnde Zuverlässigkeit der Software.

[Ehrlich und Rohn 1994] legen Wert auf Teamarbeit, zum einen unter den Software-Ergonomen selbst, damit sie nach außen hin eine einheitliche Meinung vertreten, zum anderen sollten Software-Ergonomen ein Team mit den Entwicklern bilden, um die Motivation der anderen Teammitglieder zu fördern, die Vorschläge der Software-Ergonomen auch umzusetzen. Hilfreich zur Überzeugung der Entwickler sei auch, ihnen die Möglichkeit zu geben, Tests beobachten zu können, damit sie Probleme der Benutzer selbst sehen, entweder direkt während der Untersuchung oder später auf Video.

Wie in allen Projekten sollten in den Sitzungen des Projektteams auch Vertreter von Marketing und Management anwesend sein. Es sollte eine klare Kommunikation stattfinden, die eine Einigkeit über die gesetzten Ziele erreichen soll. Zudem müssen die Ziele natürlich meßbar sein, und ein Testplan muß als Teilprojekt in das Projekt integriert werden. Um gute Benutzbarkeit in Produkten zu erreichen, müssen Mitarbeiter geschult werden, z.B. könne man sie zu Konferenzen schicken, z.B. zur 'ACM SIGCHI', denn die Methoden dieser jungen Disziplin seien noch in der Entwicklung, und so bleiben die Software-Ergonomen auf dem Laufenden. Auch dies sind Personal- und Sachkosten, die natürlich mit eingeplant werden müssen. ([Ehrlich und Rohn 1994])

[House und Price 1991] berichten von einem Instrument, das bei Hewlett Packard (HP) bereits seit 1987 zur Unterstützung der Projektplanung eingesetzt wird: die 'Return Map' (RM). Die RM bietet die Möglichkeit, Teammitgliedern verschiedener Disziplinen den Einfluß ihrer Entscheidungen auf den Projektfortschritt erkennen zu lassen. Außerdem zeigt sie die Beiträge aller Teammitglieder in den Maßen Zeit und Geld graphisch auf. Sie schließt die kritischen Elemente der Produktentwicklung (Investition und Gewinn) ein und zeigt dem Team, welche Aufgaben anstehen und nicht, wie die Verantwortungen verteilt sind. So helfe sie den Teammitgliedern, Selbstdisziplin zu entwickeln. Es werden vier Maße benutzt:

- Break-Even-Time (BET)

die Zeit bis zu dem Zeitpunkt, zu dem sich die Kosten rentiert haben (also der Zeitpunkt, ab dem jeder Verkauf Gewinn bringt),

- Time-to-Market (TM)
die Zeit bis zur Markteinführung des Produkts; wird besonders von der Entwicklungsabteilung betrachtet,
- Break-Even-After-Release (BEAR)
die Zeit vom Auslieferungstermin bis zum Break-Even-Zeitpunkt; wird besonders von der Marketingabteilung betrachtet,
- Return Factor (RF)
Gewinn/Kosten; zeigt zu einem bestimmten Zeitpunkt, wie rentabel eine Unternehmung ist (z.B. 0.45 nach 1 Jahr, aber 3.1 nach 2 Jahren); manchmal zeigt sich erst nach Jahren, ob ein Produkt erfolgreich war.

Die RM setzt genaue Schätzungen voraus und kann Fragen beantworten, wie 'Was passiert, wenn wir uns um 20 % überschätzt haben?' oder 'Was passiert, wenn wir zu spät ausliefern?'. Sie soll nicht dazu benutzt werden, um Leute zu bestrafen, deren Voraussagen falsch waren. Dies soll vor Einsatz des Instrumentes deutlich gemacht werden. Schätzungen sind Teamaufgabe. Bei jeder neuen Schätzung wird die RM angepaßt. Die Autoren weisen darauf hin, daß es sinnvoll ist, nach der Produkteinführung ein Kernteam für sechs Monate zu behalten: Es ergibt sich daraus der Vorteil, daß das Teamwissen für die nächste Produktgeneration erhalten bleibt. Bei einem wichtigen Markt werden mehrere Generationen des Produkts in den Entwicklungszyklus⁴ aufgenommen. Als Voraussetzung für eine gute RM braucht man gute Daten, um realistische Schätzungen vornehmen zu können. Zudem sollte darauf geachtet werden, daß die zukünftigen Teammitglieder gut zusammenarbeiten können. Außerdem sollten sie Erfahrung mitbringen und ein gutes Urteilsvermögen besitzen. Vorteilhaft ist, wenn Teammitglieder aus verschiedenen Sparten kommen, aber sie sollten wirklich zusammenarbeiten und sich nicht nur Ratschläge geben. Das Problem dabei ist, daß sie sich oft nicht verstehen, gerade weil sie – wie bereits erwähnt – aus verschiedenen Disziplinen kommen.

Software-Berater DeMarco vertritt aufgrund seiner langjährigen Erfahrung eine radikale Meinung über Projektplanung und Projektmanager. Eine bei Managern häufig anzutreffende Meinung über Zeitpläne scheint ihm folgende zu sein: 'Der richtigere Zeitplan ist der, dessen Einhaltung *völlig* unmöglich ist, dem man dies aber nicht *auf den ersten Blick* ansieht.' ([DeMarco 1997], S. 5). Zudem kritisiert er die oft fehlende Nutzenabschätzung. Diese sollte durch den Produktmanager und nicht den Projektmanager durchgeführt werden. Nach Fertigstellung des Produkts sollte die Richtigkeit der Nutzenkalkulation außerdem bewiesen werden. DeMarco ist der Meinung, daß das 'Unvermögen, den Zeitplan einzuhalten, ein Abschätzungs- und kein Produktionsfehler ist.' ([DeMarco 1997], S. 33). In der Projektplanung entstehen Fehler auch deswegen, weil Manager ihre Aufgabe nicht ernst nehmen, sie sich zu wichtig nehmen – oder sie es nicht besser können. Sie sehen zu wenig auf soziologische Aspekte (siehe Kapitel 2.1.5) in ihrem Team. Diese Faktoren haben einen entscheidenden Einfluß auf den Verlauf der Projektes. Um allen Mitarbeitern einen guten Überblick über den Zeitplan des Projekts zu geben und Problemen rechtzeitig begegnen zu können, empfiehlt DeMarco, Projektpläne für verschiedene Verläufe⁵ aufzustellen. Für detaillierte Hinweise zum Software—Projektmanagement siehe auch [Pressman 1992].

⁴bei HP wird der Begriff 'strategic cycle' verwendet

⁵zur Einplanung unvorhergesehener Ereignisse

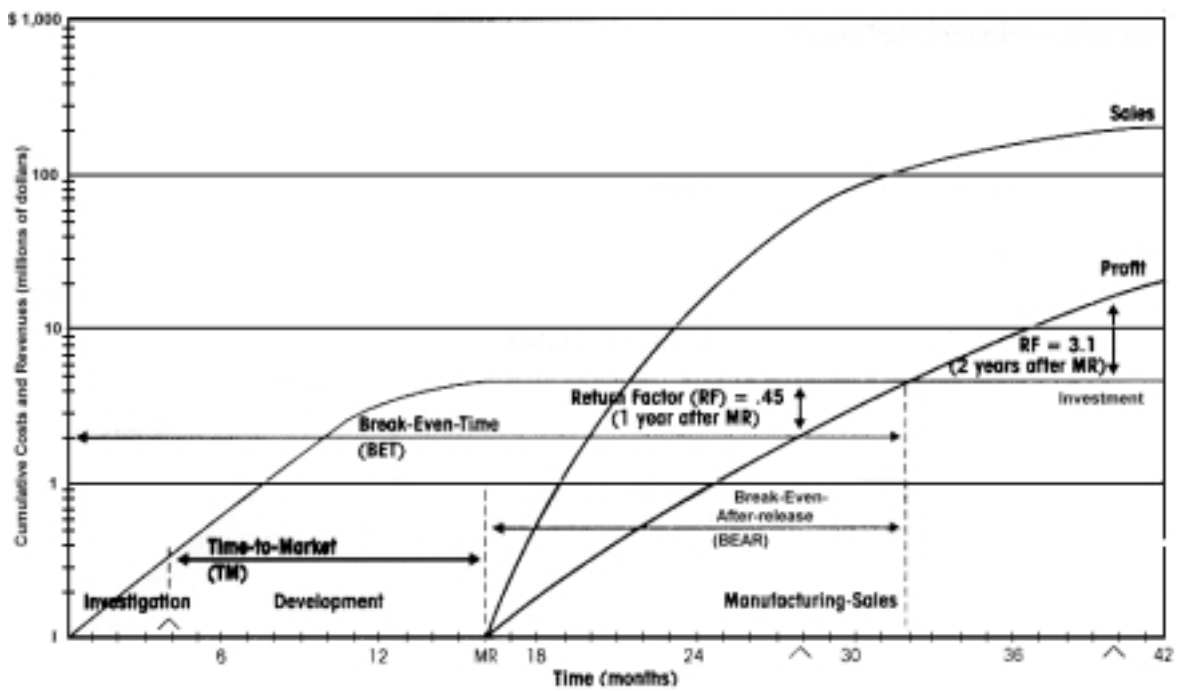


Abbildung 2.1: x-Achse: Zeit in Monaten mit Untersuchung, Entwicklung, Herstellung und Verkauf; y-Achse: Kumulative Kosten und Einnahme. Entnommen aus [House und Price 1991], S. 95.

2.1.3 Betriebswirtschaftliche Überlegungen und Methoden

Viele Autoren stellen fest, daß es sehr hilfreich ist, mit den zuständigen Managern in deren Sprache zu reden, d.h. man sollte den betriebswirtschaftlichen Vorteil betonen, also warum Softwareergonomie in starkem Maße zum Erfolg des Produkts beiträgt. [Brooks 1994] berichtet von seinem Projekt, in dem eine Software zur Früherkennung von Erdbeben für Ingenieure einer Ölplattform geschrieben werden sollte. Wegen der starken Spezialisierung des Anwendungsfeldes und damit der Ingenieure, sind Benutzer-Interviews sehr teuer. Deswegen versucht man, diese möglichst wenig einzusetzen. In einer derartigen Situation bietet es sich an, zu argumentieren, daß man durch den Einsatz software-ergonomischer Methoden Zeit der Benutzer einsparen kann. Da es andererseits nötig ist, als Entwickler das Anwendungsfeld kennenzulernen, behilft man sich damit, mehrere Versionen auszuliefern und an der jeweils aktuellen die Benutzer zu beobachten.

[Lederer und Prasad 1992] fassen in ihrem Artikel gängige Praktiken zu Kostenschätzungen zusammen. Die Daten dazu stammen aus Umfragen unter 115 IT-Managern und IT-Experten, die im Schnitt schon 14 Jahre in ihrem Unternehmen tätig waren. Etwa 60 % aller großen Projekte (derjenigen über 50.000 US-Dollar Projektkosten) überschritten die geschätzten Kosten, während 14 % darunter blieben. Zudem geben Lederer und Prasad einige Richtlinien für bessere Schätzungen an:

- erste Schätzung von Entwicklern durchführen lassen,
- erste Schätzung erst nach gründlicher Studie abschließen,
- mit Änderungswünschen durch Benutzer rechnen und diese kontrollieren,
- Fortschritt des Projektes protokollieren,
- unabhängige Prüfer den Fortschritt beurteilen lassen,
- Schätzung benutzen, um Personal zu beurteilen und zu belohnen,
- IT-Management sollte Kosten-Schätzung überprüfen,
- auf dokumentierte Fakten verlassen und Standards und einfache Formeln verwenden,
- nicht auf die Schätzung unterstützende Software verlassen.

Auch [Mayhew und Mantei 1994] geben Empfehlungen zur realistischen Schätzung der Kosten. Sie raten, Veröffentlichungen über Produktbeurteilungen zu studieren, die sich stark an Benutzbarkeit der Oberfläche und des Handbuchs und an der Sicherheit des Produkts orientieren. Sie geben zusätzlich eine Übersicht über Designalternativen in obiger Tabelle. Diese Designalternativen werden beeinflußt sowohl durch Hardware, z.B. den Einsatz von Maus oder Tastatur, als auch durch Softwareentscheidungen, wie die Anordnung von Menüeinträgen. [Karat 1994] berichtet, daß [Scerbo 1991] und [Bosert 1991] eine Methode zum Strukturieren der Entwicklung unter Beachtung von Nutzeranforderungen entwickelt haben, das quality functional development (QFD). Dies ermögliche eine Ersparnis von einem Drittel bis zur Hälfte der Entwicklungskosten.

[Lund 1997] verfolgt nach eigenen Angaben einen anderen Ansatz als [Bias und Mayhew 1994a]. Das Problem bei dem Ansatz von Bias und Mayhew liegt seiner Meinung nach daran, daß Entscheider die Ergebnisse voriger Projekte nicht als relevant für zukünftige Projekte ansehen. Sie betrachten eher verschiedene zeitgleiche Projekte und fragen sich, in welches Projekt sie die vorhandenen Mittel investieren sollen, um den größten Nutzen zu erreichen. Der alternative

Ansatz von Lund beruht darauf, daß Benutzbarkeits-Tests als Teil der Software-Qualitätssicherung angesehen werden und argumentiert so, daß früh entdeckte Bugs⁶ billiger zu beheben sind. Benutzbarkeitsprobleme werden in diesem Zusammenhang als Bugs aufgefaßt. Lund berichtet von Ameritech⁷, daß alles, was in dem Unternehmen passiert, meßbar sein und in Relation zu den Gewinnen gebracht werden muß. Eine Rechtfertigung der Kosten besteht darin, den Wert der Aktivität und wie er sich im Lauf der Zeit entwickelt, darzustellen. Wenn der Teil des Geschäfts nicht bewertet werden kann, ist er ein Kandidat für die Eliminierung, ebenso, wenn er weniger wert ist als alternative Projekte. Die Meinung, daß man auch mit weniger Probanden auskommt, teilt Lund nicht. Es gehe immer etwas verloren. Er stellt fest, daß Entscheider nicht betrachten, ob sich die Kosten für ein Projekt lohnen, sondern, ob die Investition besser in Projekt A oder B getätigt werden sollte. Vor diesem Hintergrund gibt es zwei Wege: entweder neue Produkte zu entwickeln, bzw. neue Geschäftsideen zu finden, oder zu argumentieren, daß die Arbeit von Softwareergonomen die Gewinne erhöht und die Kosten verringert. Letzteres wird durch das Argument gestützt, daß der Wert des Unternehmens gesteigert wird, weil Systeme verbessert werden und Benutzbarkeit von Produkten Unternehmenskosten reduziere und damit ein Produkt erfolgreicher am Markt sei. Eine Produktivitätssteigerung sollte sich in einer erhöhten Anzahl von Ideen pro Person/Jahr wiederfinden lassen. Bei Ameritech wird eine Art Ideendatenbank geführt; sobald ein Mitarbeiter eine Idee hat, wird sie in eine Datenbank eingegeben. Dann wird sie auf Durchführbarkeit, Gewinnpotential und 'strategischen Nutzen' geprüft.

Gewinne werden mittels NPV⁸ dargestellt. Der Gesamtwert der von einer Abteilung generierten Idee berechnet sich wie folgt:

$$IV = \sum_{k=1}^i (NPV_k) \quad (2.1)$$

mit i = Anzahl der Ideen.

Software-Ergonomen bei Ameritech definieren verbesserte Benutzbarkeit so: 'making a system or a product easy to learn, easy to use, useful, compelling and safe' (S. 53). In interner Entwicklungen werden Kosten gespart durch größere Produktivität der Benutzer, durch geringere Trainingskosten und weniger Kosten aufgrund von Benutzerfehlern. Gewinne von Kommunikations- und Informationsprodukten entstehen durch größere Verwendung und den daraus resultierenden Gewinnen (größerer Markterfolg). Die Erfahrung besagt, daß Kunden, die mit einem Produkt zufrieden sind, eher ein weiteres kaufen. Einige Projekte kann man gut mit der Methode von Bias und Mayhew berechnen, andere eher aufgrund historischer Daten. Ameritech unterhält einen Katalog mit Projektdaten (Vorher-/Nachher-Daten) über Produktivität. Damit kann man Manager nach Lunds Erfahrung überzeugen.

Der Total value of usability berechnet sich wie folgt:

UV = Summe(durch Feldmessungen gemessene und dem Projekt hinzugefügte Werte) + Summe (aufgrund spezieller Verbesserungen der Benutzbarkeit geschätzte Werte) + Summe (10%ige Verbesserung für Projekt)

Die Schätzung von 10% ist sehr konservativ, die Datenbank enthält nur Projekte, deren Verbesserungen mindestens 30% betragen.

⁶Programmierfehler

⁷Dort war Lund zum Zeitpunkt der Veröffentlichung des Artikels beschäftigt.

⁸net present value; in [Bias und Mayhew 1994a] definiert als: Gesamtprojektgewinne + aktuellem Wert der Projektkosten - aktueller Wert des Projektnutzens

Es wird versucht, jeden Software-Ergonomen an wenigen Projekten zu beteiligen, weil die Beteiligung an vielen Projekten dazu führt, daß er wegen seiner geringen Beteiligung am Projekt nicht als vollwertiges Team-Mitglied von den anderen Mitgliedern anerkannt wird. Total Value: Der Ansatz basiert darauf, daß jede Aktivität mit dem Aktienpreis in Verbindung gebracht werden kann. Man berechnet den Total Value aus der Summe von IV und UV. Die jährlichen Gewinne (YE) berechnen sich dann wie folgt:

$$YE = \frac{TV}{n * VP} \quad (2.2)$$

Einen Index in Cents pro Person erhält man durch:

$$I = [(YE/C)/S] * 100 \quad (2.3)$$

mit C = Kapitalisierungsrate, d.h. Dividende + Kursgewinn und S = Anzahl ausgegebener Aktien liefert. Diese Schätzungen werden ständig aktualisiert. Nach Lund ist dies ein effektiver Weg, über ein Programm zur Benutzbarkeitsverbesserung mit dem Management zu sprechen.

Langfristige Vorteile sieht [Karat 1994] in besseren Verkaufszahlen und Einnahmen, sowie in der erhöhten Produktivität der Benutzer und deren Zufriedenheit. Reduziert werden zudem Kosten für Training und Support und Dokumentation. Eine reduzierte Entwicklungszeit kann evtl. Marktvorteile dadurch bringen, daß das eigene Produkt eher am Markt ist als das der Konkurrenz. Dieser Aspekt wird von [Hackos und Redish 1998] unterstrichen. Sie halten die Berechnung des ROI⁹ für notwendig, weil 'The expense today will result in greater savings or greater revenue later' (S. 122). Auch der Verlust von Gewinnen kann als Kostenfaktor gesehen werden. [House und Price 1991] stellen fest, daß Firmen oft 33 % des Nach-Steuer-Gewinns durch eine Verspätung von 6 Monaten in der Auslieferung ihres Produktes verlieren. Durch einer Erweiterung des Entwicklungsbudgets um 50 % wäre in dem Moment nur ein Verlust von 3,5 % entstanden. Auch Conklin hält zu späte Produktauslieferung für eine 'cardinal sin' ([Conklin 1991]). Zitiert nach [Karat 1994]: Eine Verspätung um ein Viertel der geplanten Projektzeit hatte einen 50 %igen Gewinnverlust zur Folge.

[Karat 1994] benennt einige Fallstudien, in denen sehr einfache Berechnungen angestellt wurden:

- [Brown 1991] und [Diverse Autoren 1991]: Die Ausgangssituation bestand in sinkenden Gewinnen und unnötigen Serviceanrufen aufgrund von nicht zuverlässigen Maschinen. Nachdem die redesignten Maschinen den Markt erreichten, steigerten sich sofort die Gewinne. Es entstand eine Gewinnerhöhung um 1 Mrd. \$ über drei Jahre bei geschätzten Kosten von 2 Mio. \$.
- [Karat 1990]: Vor der Auslieferung einer neuen Version eines bestehenden Systems wurden drei Benutzbarkeitstest-Durchgänge vorgenommen. Die Mitarbeiter nutzen das System 12 mal am Tag. Durch Minimierung der Zeit um 4,67 Minuten wurden in den ersten drei Tagen allein schon 41.700 \$ eingespart.
- [Karat 1990]: Durch das Redesign eines Systems wurde eine Reduzierung der Zeit für die Durchführung einer Aufgabe um 9,6 min. vermindert. Hieraus ergab sich eine Ersparnis im ersten Jahr von 6.800.000 \$

⁹Return of Investment

[Karat 1994] erläutert außerdem Methoden der Analyse aus der Betriebswirtschaft, die einem Software-Ergonomen bei der Argumentation dem Management gegenüber helfen sollen. Zunächst werden zwei einfache Analysetechniken vorgestellt.

Die Kosten-Nutzen-Rate berechnet sich ganz einfach. Man teilt den Betrag, der für den Nutzen geschätzt wird, durch den Betrag, der für den Aufwand geschätzt wird. Damit erhält man die Größe des Nutzens pro aufgewendeter GE¹⁰.

Die 'payback period' gibt an, nach welcher Zeitspanne sich ein Aufwand gelohnt hat. Bei einem Aufwand von 50 GE und einem zusätzlichen Ertrag von 10 GE pro Jahr beispielsweise ergibt sich eine 'payback period' von fünf Jahren. Berechnet wird prinzipiell nach der Formel

$$R_1 + R_2 + \dots + R_i \geq C \quad (2.4)$$

mit R = Ertrag - Kosten im Jahr i und C = einmalige Entwicklungskosten. Für das 'payback' hat jede Firma ihre eigenen Grenzen, die, wenn sie nicht eingehalten werden, die Projekt ablehnung zur Folge haben. Prinzipiell gilt: je kürzer das 'payback', desto besser ist die Investition nach der Meinung des Managements. Allerdings ist es kein Maß für Projektprofitabilität.

Zu diesen beiden einfachen Analysetechniken kann man weitere einsetzen. Zur Berechnung der Kosten sollte man auch den Zeitwert des Geldes mitberechnen, d.h. den Ertrag, den das Geld angelegt erbracht hätte. Also, angenommen, man erwartet für ein Projekt einen Ertrag von 50.000 \$ im zweiten Jahr und fragt sich, wieviel Geld man heute anlegen müßte, um in zwei Jahren die genannte Summe zurückzubekommen unter der Annahme einer bestimmten Gewinnrate, z.B. 15%. Diese Gewinnrate wird wie oben die 'payback period' von dem Unternehmen als akzeptierte Minimumrate festgelegt. Man berechnet den heute anzulegenden Wert p wie folgt:

$$p = F_n * PVIF \quad (2.5)$$

mit

$$PVIF = \left(\frac{1}{1+i} \right)^n \quad (2.6)$$

und F_n = zukünftig erwarteter Wert in n Jahren (hier = 2), i = Gewinnrate und PVIF = 'present value interest factor', einem Wert, den man wie gezeigt berechnen oder aus einer Tabelle wie in ([Karat 1994], S. 61) ablesen. In dieser Tabelle kann man ablesen, wieviel Prozent das aktuelle Kapital wert ist bei einer angenommenen Wertsteigerung von x (12-35%). Im genannten Beispiel ergibt sich hier ein Anfangskapital von 37.805 \$ (50.000 \$ * 0,7561), die heute angelegt werden müßten. Man kann ebenso berücksichtigen, daß in den Folgejahren unterschiedliche Erträge erwartet werden, indem man für die einzelnen Jahre jeweils die Einzelsummen berechnet und dann alle aufsummiert:

$$p = \sum_{i=1}^n f_i \frac{1}{1+r} \quad (2.7)$$

mit r = Gewinnrate und n = Anzahl Projektjahre. Beispiel: Einnahmen 50, 100, 200 GE in den ersten drei Jahren, Gewinnrate = 17%. Daraus ergibt sich ein aktueller Wert des Projektnutzens von $50 * 0,85 + 100 * 0,73 + 200 * 0,62 = 240$ GE, d.h. würde man 240 GE einsetzen erhält man nach drei Jahren gerade den Projektgewinn. Erweitert wird diese Berechnung noch durch

¹⁰Geldeinheit(en)

Hinzuziehen der Kosten. Hieraus errechnet sich der aktuelle Nettowert NPV¹¹:

$$NPV = P_{ges} - C \quad (2.8)$$

mit C = aktueller Wert der Projektkosten, also der aktuelle Wert des Nutzens – dem aktuellen Wert der Kosten. Ein Projekt wird nur dann genehmigt, wenn der NPV positiv ist, z.B. bei angenommenen Projektkosten von 200 GE für das obige Beispiel ergibt sich ein Positivbetrag von 40 GE, das Projekt kann also genehmigt werden. Bei konkurrierenden Projekten wird ein Profitabilitätsindex P_i erstellt:

$$P_i = \frac{\text{aktuellerWert(Projekteinnahmen)}}{\text{aktuellerWert(Projektkosten)}} \quad (2.9)$$

Die am meisten verbreitete Methode ist die der Internal Rate of Return (IRR). Unternehmen setzen hierfür ebenfalls eine Minimalrate an, die ein Projekt einhalten muß, um angenommen zu werden. Zur Berechnung bestimmt man dasjenige i , für das NPV = 0 wird.

[Conklin 1991] weist darauf hin, daß Manager die Zeit bis zum Breakeven¹² betrachten sollten. Er splittet die Zeit nach der Markteinführung in zwei Intervalle, das erste endet zu dem Zeitpunkt, an dem signifikant viele Verkäufe stattfinden, das zweite endet bei dem Breakeven-Zeitpunkt.

[Nielsen und Landauer 1993] haben in einer Untersuchung von elf Studien herausgefunden, daß sich das Poisson-Modell zur Planung des Evaluationsumfangs gut geeignet. Betrachtet wurden dabei zwei Arten von Untersuchungen, zum einen Beurteilungen durch Software-Ergonomen¹³ und zum anderen Untersuchungen mit Probanden¹⁴. Die Anzahl voneinander verschiedener Fehler¹⁵ kann wie folgt errechnet werden:

$$FOUND(i) = N(1 - (1 - \lambda)^i) \quad (2.10)$$

Dabei ist i die Anzahl der Evaluatoren bzw. Probanden, N die Anzahl vorhandener Benutzbarkeitsprobleme und λ die Wahrscheinlichkeit, ein durchschnittliches Benutzbarkeitsproblem mit einem einzigen Evaluator bzw. Probanden zu finden¹⁶. N und λ kann man nach mindestens zwei Evaluationen abschätzen. Nach sechs Evaluationen liegt die Standardabweichung sogar bei nur 9%. Eine grobe Schätzung kann oft auch schon vorher aufgrund der Erfahrungen im Unternehmen möglich sein. Für $i = 2$ ergeben sich die Abschätzungen

$$\lambda = 2 - (FOUND(2)/FOUND(1)) \quad (2.11)$$

und

$$N = FOUND(1)/\lambda \quad (2.12)$$

FOUND(i) sei dabei die Anzahl voneinander verschiedener gefundener Fehler nach i Untersuchungen. Eine Kosten-Nutzenanalyse ergab die in Tabelle 2.2 dargestellten Raten.

Auch [Viereck 1995] gibt ein Verfahren zur Nutzenschätzung an:

¹¹net present value

¹²der Zeitpunkt, zu dem sich die Ausgaben rentiert haben; liegt ein ganzes Stück nach der Markteinführung

¹³d.h., Software-Ergonomen untersuchten die Benutzungsschnittstelle unter Einbeziehung ihres Fachwissen

¹⁴d.h., Probanden benutzen die Software und Software-Ergonomen hielten die Probleme fest, auf die die Probanden stießen

¹⁵Anzahl der Fehler, die mindestens einmal gefunden wurden

¹⁶abhängig von Eigenschaften des jeweiligen Systems, Zeitpunkt der Untersuchung innerhalb des Entwicklungszyklus, der verwendeten Methode und anderen Faktoren

Projektgröße	Kosten	Gewinn	Rate
klein	10.000	37.500	3,8
mittel	18.000	613.000	34
groß	46.000	8.200.000	178

Tabelle 2.2: Aufstellung über die Kosten–Nutzen–Raten verschieden großer Projekte bei Untersuchungen mit Probanden, bei heuristischer Evaluation ergeben sich noch bessere Werte; die besten Raten ergeben sich bei 3,2 Probanden oder 4,4 Heuristiken. Entnommen aus [Nielsen und Landauer 1993], S. 211f.

- für das Projekt relevante Faktoren bestimmen
- Faktoren gewichten (entweder mit Verhältnisskala, d.h., die Summe aller Gewichte ist 100, oder Nominalskala, also z.B. Noten)
- Bewertung von Maßnahmen bzgl. Einfluß auf Faktoren
- Gewicht * Bewertung
- Berechnung des Nutzenkoeffizienten $NK = \text{Summe(Produkte)}/\text{Summe(Gewichte)}$;
 $NK > 0 \Rightarrow$ Nutzen gegeben, $NK > 1$ verbesserte Wirtschaftlichkeit des neuen Systems

Das Verfahren hat den Nachteil, daß die Schätzungen subjektiv sind. Dies kann man verhindern, indem man die Berechnungen durch Personen verschiedener Interessengruppen durchführen läßt.

2.1.4 Das Produktivitätsparadoxon

Die Meßbarkeit von Benutzbarkeit spielt im Produktivitätsparadoxon eine entscheidende Rolle. Im vorigen Abschnitt wurde gezeigt, daß es schon eine ganze Reihe von Ansätzen zur Berechnung der Produktivität software–ergonomischer Maßnahmen gibt. Die Existenz des Produktivitätsparadoxons zeigt jedoch, daß diese nicht genug bekannt sind oder einfach nicht genutzt werden. Das Produktivitätsparadoxon besteht nach [Brynjolfsson 1993] darin, daß sich zwar der Einsatz von Computern um mehr als zwei Größenordnungen erhöht hat, aber die Produktivität, speziell die des tertiären Sektors¹⁷, zu stagnieren scheint. Wissenschaftler können den vorhandenen Abfall an Produktivität in den siebziger Jahren im Vergleich zur Nachkriegszeit nicht erklären und kommen so zu der Meinung, IT¹⁸ trage nichts zur Produktivität in diesem Sektor bei.

[Brynjolfsson 1993] betrachtet mehrere Studien zu diesem Thema und stellt fest, daß sämtliche Kernergebnisse besagen, daß der Einsatz von IT im Industriesektor zur Produktivität beiträgt, im Service-Sektor sei er allerdings ineffektiv und teuer. Dies liegt seiner Meinung nach daran, daß die Produktivität eines IT-Mitarbeiters nicht direkt meßbar ist und daß es für die Produktivität zu viele Einflußfaktoren gibt (S. 68).

¹⁷Dienstleistungssektor

¹⁸Informationstechnik

[Brynjolfsson 1993] stellt vier Erklärungen für das Paradoxon vor:

- unzureichende Meßmethoden der Daten:
 - Traditionelle Meßmethoden eignen sich nicht für die neuen Wertschöpfungen, die durch IT-Einsatz geschaffen werden.
 - Es bestehen Schwierigkeiten, die Vorteile in Geldwerten zu messen, z.B. den Vorteil des Kundenservice eines Geldautomaten. Es werden weniger Schecks ausgestellt, damit erniedrigt sich die Produktivität, aber im Gegenzug werden die Vorteile, die daraus entstehen, wie verbesserte Abrechnungszeiten und erhöhter Kundenservice, nicht in der Produktivitätsstatistik aufgeführt.
 - Bei neuen Produkten bestehen keine Vergleichsmöglichkeiten.
 - Gewinne sind versteckt, z.B. werden durch den Einsatz von IT Sekretariatskräfte entlastet. Dies führt aber nicht zu einem direkten Gewinn dadurch, daß diese Kräfte weniger Gehalt bekommen, ihre Arbeitskraft wird frei für neue Aufgaben.
 - Vernachlässigung der Inflationsrate.
 - Kosten und Nutzen werden nicht auf die gleichen Zeiträume berechnet, z.B. werden Trainigskosten, die Vorteile über mehrere Jahre beinhalten, dem Jahr der Anschaffung der neuen IT-Mittel zugerechnet.
- Verzögerung der Vorteile aufgrund von Lern- und Anpassungsprozessen:
 - Man sieht zuerst nur die aktuellen Aufwände, aber nicht die langfristigen Gewinne. Letztere müssen aber miteingerechnet werden, um realistische Aussagen zu treffen.
- Umverteilung und Verschleierung des Gewinns:
 - IT vermag die Produktivität einzelner Firmen zu erhöhen, aber nicht die der gesamten Marktwirtschaft.
 - Der Einsatz von IT kann das Marketing einer Firma unterstützen, aber dadurch wird der Firmengewinn nicht direkt beeinflußt.
 - Firmen mit nicht angepaßten IT-Strategien verlieren Marktanteile an diejenigen Firmen, die ihre IT besser einsetzen, so daß der Ausgleich auf gesamtwirtschaftlicher Ebene stattfindet.
- Fehlmanagement:
 - Veraltete Daten werden zur Entscheidungsfindung im Management herangezogen oder die Firmeninteressen werden gar nicht berücksichtigt.
 - Der Unterschied zum Industriesektor liegt darin, daß dieser stärker dem Konkurrenzdruck auf dem internationalen Markt unterworfen ist und so weniger Nachlässigkeit in den Entscheidungen vertragen kann.
 - Dadurch, daß Schwierigkeiten mit der Meßbarkeit existieren, werden Heuristiken eingesetzt statt genaue Berechnungsmethoden zu wählen.

Nach [Brynjolfsson 1998] bedarf es zusätzlicher Investitionen, neuer Strategien und Arbeitsprozesse, die Organisation muß miteinbezogen und gegebenenfalls umgestaltet werden, um eine Erhöhung der Produktivität zu erreichen. Der alleinige Einsatz von IT an sich reiche nicht aus. 'By definition it doesn't come from working harder.[...] Productivity growth comes from working smarter.' (S. 50). Die Produktivität steigt dann stark an, wenn bahnbrechende Erfindungen wie die Dampfmaschine auf den Markt kommen. Die zusätzlichen Investitionen sind

unabdingbar zur Erreichung einer besseren Produktivität. Firmen schrecken davor zurück, weil sie einen Verlust an Zeit und Kosten und ein zu hohes Risiko befürchten. [Kling 1999] unterstützt die These, daß der reine Umgang mit Technik nicht ausreicht, sondern daß man jene auch effektiv einsetzen muß. Er stellt drei Gründe für das Paradoxon fest:

- Viele Organisationen entwickeln Systeme, die zu einer zu starken Zersplitterung der Organisation führen.
- Wenige Organisationen entwickeln Systeme, die den Mitarbeitern die Arbeit wirklich erleichtern.
- Es wird unterschätzt, wie groß der Einfluß qualifizierter Arbeit auf den effektiven Einsatz von IT ist.

2.1.5 Organisationsstruktur und soziale Aspekte

[Bias und Mayhew 1994b] stellen fest, daß die erfolgreiche Einführung softwareergonomischer Techniken Veränderungen in der Organisation des Unternehmens erfordert. Es ist erforderlich, die Gründe für Widerstände zu verstehen, um sie zu vermeiden. Im folgenden werden bei Managern und Entwicklern häufig anzutreffende Annahmen aufgelistet (S. 289):

- Qualität der Benutzungsschnittstelle ist nicht wichtig.
- Mit Richtlinien für Schnittstellen vertraute Designer garantieren eine gute Benutzerschnittstelle. (Das allein reicht nicht, man benötigt auch effektive Management-Strategien.)
- Design-Aufgaben für Benutzerschnittstellen werden erst spät im Projekt integriert, sie sollten jedoch schon früh eingebunden werden.
- Benutzbarkeit ist subjektiv und kann nicht gemessen werden.
- Das Design einer Benutzerschnittstelle kann im ersten Anlauf gut werden. (vgl. [Norman 1989])
- Das Design von Benutzerschnittstellen ist impliziter Teil des Software-Designs und muß nicht extra geplant und budgetiert werden.

In vielen Firmen befinden sich Entwicklung und Marketing in verschiedenen Abteilungen, so daß eine Kosten-Nutzenanalyse keine Aufmerksamkeit gewinnt. Software-Ergonomen befinden sich meist nicht in der Position, angemessene Veränderungen in der Organisation durchzuführen, also müssen sie jemanden finden, der so hoch in der Unternehmenshierarchie angesiedelt ist, daß er derartige Veränderungen anordnen kann. Ein anderes Problem besteht in einer sozialen Komponente. Software-Ergonomen sind oft die 'only person on the project team who generates zero lines of code' ([Bias und Mayhew 1994b], S. 301). Zudem sind sie oft 'einsame Arbeiter', weil sie keine Kollegen in dem Unternehmen haben. So werden sie oft von anderen Projektmitgliedern nicht akzeptiert. Der Lösungsansatz für dieses Problem besteht darin, Software-Ergonomen als Vollzeit-Projektmitglieder einzusetzen und zusätzlich eine zentrale Abteilung einzurichten, die die Aufgabe hat, Informationen zu sammeln und Evaluationen durchzuführen und die einzelnen Software-Ergonomen in ihrer Projektarbeit zu unterstützen. [Brown 1991] berichtet, daß die Forschungsabteilung von Xerox auch Untersuchungen von Organisationsstrukturen und ihre Beeinflussung durch den Einsatz neuer Techniken durchführt. Er befürwortet, die späteren Anwender direkt an ihrem Arbeitsplatz zu beobachten, um ihre Arbeit zu analysieren, denn Leute arbeiten oft anders, als das Organisationshandbuch es vorschreibt, weil sie z.B. auf unvorhergesehene Probleme reagieren. Diese Meinung vertreten auch Beyer und Holzblatt, die ihre

Methode ‘Contextual Design’ propagieren ([Beyer und Holtzblatt 1998]). (vgl. Kapitel über die Arbeitsanalyse des Anwendungsfeldes)

[DeMarco 1997] stellt fest, daß Manager soziale Aspekte oft unbeachtet lassen. Seiner Meinung nach sind für Manager Kenntnisse in Soziologie wichtiger als Kenntnisse über Technologien. Im Rahmen seiner Tätigkeit als Software-Berater hat er oft beobachtet, daß Manager oft die Vorstellung haben, daß ein enger Terminplan ein guter Motivationsanreiz sei. Aber nach seiner Erfahrung führt zuviel Druck auf die Mitarbeiter dazu, daß diese das Unternehmen oder das Projekt verlassen. Vielmehr sollten Manager ihre Software-Entwickler von Routinearbeiten¹⁹ befreien. Die Ressource Softwareentwickler wäre so besser genutzt, einmal, weil Routineaufgaben dann nicht mehr von teuren Arbeitskräften durchgeführt werden müssen, und zum anderen, weil die Motivation der Entwickler steigt, da sie sich interessanteren Aufgaben widmen können. Außerdem sollte man die sogenannten Denkkosten berücksichtigen und nicht nur auf die LOC’s²⁰ schauen. Zeilenanzahlen seien ein ungeeignetes Maß, um Softwarequalität zu messen. Als Beispiel führt er die Software ‘Quicken’ an. Dies sei ein sehr kleines, aber weit verbreitetes und gut benutzbares Produkt. Zudem herrsche das allgemeine Bild vor, daß die Mitarbeiter, die sehr viele Überstunden machen, Helden seien, jedoch der Mitarbeiter, der die ihm übertragene Aufgabe in der vorgegebenen Zeit erledige, nur durchschnittlich sei. Überstunden brächten aber nur kurzfristigen Erfolg, langfristig seien Auswirkungen wie ein ‘burnout’ und ein Abwandern der Mitarbeiter die Folge. DeMarco propagiert auch die ‘Einheit des Ortes’: ‘Der Alltag eines Software-Spezialisten besteht darin, mit anderen Leuten zu reden, [...] über eine Entfernung hinweg zu kommunizieren verdoppelt [...] die Kosten [...]’. ([DeMarco 1997], S. 52) Die Kommunikationskultur im Projekt spielt auch eine große Rolle im frühzeitigen Erkennen von Problemen. Ein ‘Das-schaffen-wir-Management’ verhindert, daß schlechte Nachrichten nach oben durchdringen, dies sei eine echte Katastrophe.

2.1.6 Open-Source-Software

Open-Source-Software (OSS) ist Software, die ohne Lizenzgebühren und mit Offenlegung des Quellcodes geliefert wird. In der [Computerwoche 2000] läßt sich nachlesen, daß der Einsatz von OSS eine Reihe von Vorteilen bietet. Zum einen bekommt der Anwender leichten Zugang zum Programmierer, denn in der Regel steht seine Email-Adresse im Programm selbst. Zum anderen ist der Service besser, wird Eric Raymond zitiert, weil der Verdienst bei Anbietern von OSS in dem Angebot von Service besteht und nicht im Verkauf der Software, wie bei herkömmlichen Anbietern von Software.

Jedoch wird hier ausdrücklich darauf hingewiesen, daß gerade diese Software einer gewissen Benutzerfreundlichkeit entbehre. ‘Open-Source-Entwickler haben [...] häufig eine stark technische Ausrichtung und denken beim Zielpublikum oft an ihresgleichen.’ ([Computerwoche 2000], S. 22) Dies führt zu wenig benutzerfreundlichen Produkten und damit wiederum zu erhöhten Kosten bei der Benutzung (siehe z.B. [Brodbeck 1991] auf Seite 45). Auch [Kuniavsky 2000] stellt fest, daß OSS keine benutzerfreundlichen Schnittstellen aufweist. Dies liegt seiner Meinung nach daran, daß OSS zumeist für Anwender geschrieben wird, die Programmiererfahrung haben. Es wird bei der Herstellung Wert auf Flexibilität gelegt, nicht auf Benutzerfreundlichkeit. Außerdem gebe es dort keine etablierten Methoden, Feedback von den Benutzern zu bekommen.

¹⁹z.B. Kommentieren von Code

²⁰Anzahl der produzierten Code-Zeilen

2.1.7 Werkzeuge zur Rechtfertigung der Kosten

[Harrison, Hennemann und Blatt 1994] sahen die dringende Notwendigkeit, ein Werkzeug für die Aufgabe der Kostenrechtfertigung bereitzustellen. In einer Untersuchung wollten sie herausfinden, welche Funktionalitäten ein solches Werkzeug bieten muß, um akzeptiert zu werden. Zu diesem Zweck wurden zwei Wege beschritten: Zum einen wurde eine Literaturübersicht aus verschiedenen Disziplinen (Marketing, Finanzwesen, Softwareergonomie, Werbung, Betriebswirtschaft, Management, Ingenieurswesen, Qualitätssicherung, Kundenservice und Informatik) erstellt, zum anderen wurden Interviews mit potentiellen Anwendern²¹ geführt. Ein Problem besteht darin – wie auch im Bereich Marketing –, den Effekt der einzelnen Methoden zu messen, da zu viele Einflußfaktoren vorhanden sind, die Seiteneffekte auslösen können. Eine mögliche Lösung bestehe darin, die Wahrscheinlichkeiten zu berechnen und den Nutzen abzuschätzen. Alternativ könne man eine Sensitivitätsanalyse durchführen, d.h. den besten, den schlechtesten und den wahrscheinlichsten Fall zu berechnen, wie es auch bei Finanzanalysen gemacht wird, bei denen keine Annahmen über die Variablen oder deren Beziehung zueinander gemacht werden können. Eine Übersicht über die untersuchten Studien ist in [Harrison, Hennemann und Blatt 1994], S. 210 ff. zu finden. Hier zwei Beispiele:

1. Nutzen für den Endanwender:
Benutzer geben gute Benutzbarkeit als zweitwichtigsten Faktor an, mit 6.8 von 10 Punkten.
2. Nutzen für die softwareherstellenden Firmen:
Wenn durch Qualitätssicherungsmaßnahmen Fehler in der Anfangsphase eines Projekts gesenkt werden können, kann sich daraus eine Ersparnis von 50 % der Kosten für Fehlerbeseitigung ergeben.

Befragt nach Kriterien, die von dem Werkzeug überzeugen, wurden als wichtigste Referenzen und die Präsentation der Ergebnisse genannt. Es wurde gefordert, daß sich dieses Werkzeug als Überzeugungshilfe für das Projekt eigne, um die Integration von Benutzbarkeitsanalysen in ein Projekt zu integrieren. Außerdem soll es Kostenschätzungen ermöglichen. Aus technischer Sicht wurde gefordert, daß das Werkzeug die Kriterien Validität und Korrektheit erfüllt sowie flexibel und einfach zu handhaben ist. Die zu berücksichtigenden Kosten betrafen die meisten Benutzbarkeitsanalyse-Methoden wie Rapid Prototyping, Szenarien, Trainingskosten oder Produktvergleiche. Als Ausgabe wird eine einseitige Präsentation mit den Hauptergebnissen der Analyse gewünscht.

Die Autoren stellen fest, daß zwei Aufgaben anstehen: Zum einen müssen weitere Informationen über den konkreten Zusammenhang zwischen dem Einsatz von Benutzbarkeitsanalysen und den Vorteilen für die Benutzer der zu erstellenden Software gesammelt werden, zum anderen müssen die Anforderungen der Anwender an das Werkzeug weiter untersucht werden. Eine baldige Umsetzung der Ergebnisse in ein marktfähiges Produkt ist sicher wünschenswert.

Eine alternative Metrik für die Schätzung des Nutzens haben [Person et al. (keine Angabe)] bei ihrem 'CERTAIN tool'²² benutzt. Es dient der Abschätzung der Kosten für die Entwicklung

²¹Diese sind 'cognitive engineering professionals', also Personen, die Erfahrung in der Entwicklung und Benutzung von Werkzeugen und Methoden haben, die zum Zweck von Benutzbarkeitstests eingesetzt werden

²²Dieses Werkzeug umfaßt hauptsächlich Richtlinien und Checklisten für die Identifizierung und Quantifizierung benötigter Ressourcen. Erstellt wurden diese Listen nach einer umfangreichen Untersuchung, die folgende vier Punkte einschloß: Identifizierung kritischer Faktoren, Einschätzung sozio-ökonomischer Faktoren, Fallstudien und Studien zur Anwendbarkeit.

unterstützender Technik für Behinderte. Der Nutzen wurde hierbei nicht in Geldwert gemessen sondern in den Möglichkeiten, das tägliche Leben dieser Benutzergruppe zu erleichtern.

2.2 Einmalige Kosten

Im folgenden sollen die einmaligen Kosten betrachtet werden. Dies sind Kosten, die einmalig zu Beginn eines Projektes anfallen, wie z.B. Kosten für Schulung von Mitarbeitern oder die Anschaffung von Geräten oder Software.

Bereits 1988 gaben [Mantei und Teorey 1988] eine Übersicht über die Kosten, die dadurch entstehen, daß software-ergonomische Methoden in den Entwicklungsprozeß integriert werden. Sie schätzten dafür einen Gesamtaufwand von 128.330 US-Dollar (S. 431), eine Studie mit fünf Benutzern kostete 7.320 US-Dollar. Daß man Benutzbarkeitsstudien auch viel günstiger erfolgreich durchführen kann, zeigt [Nielsen 1994a]. Er stellt das sogenannten 'discount usability engineering' vor. Dies besteht aus dem Einsatz von drei Methoden:

1. Szenarien und Paper-mock-ups²³: Horizontale Prototypen reduzieren den Grad der Funktionalität und vertikale Prototypen bieten weniger Funktionalität, aber diese dann komplett. Szenarien verbinden diese zwei Ansätze. Papier-Prototypen sind günstig und können schnell geändert werden; als Nebeneffekt lernen die Entwickler das Anwendungsfeld besser kennen.
2. Lautes Denken - Diese Untersuchung findet nicht im Labor statt, sondern es werden drei bis fünf Anwender mit der Durchführung typischer Aufgaben direkt an ihrem Arbeitsplatz betraut. Dabei wird die Methode des lauten Denkens angewendet, d.h., die Benutzer äußern durchgehend ihre Gedanken, während sie arbeiten.
3. Heuristische Evaluierung - Überprüfung der Gültigkeit folgender Kriterien durch Experten:
 - Einfachheit und Natürlichkeit der Dialoge,
 - Verwendung der Sprache des Anwenders,
 - Minimierung der Anforderungen an das Gedächtnis des Anwenders,
 - Konsistenz,
 - Feedback,
 - Klare Markierung der Ausstiegsmöglichkeiten,
 - Existenz von Shortcuts (für Experten),
 - Qualität der Fehlermeldungen,
 - Vorbeugung von Fehlern,
 - Existenz einer qualifizierten Hilfe und Dokumentation.

Zur Schätzung des Nutzens stellt Nielsen fest, daß der einzige Weg, exakte Daten über den Nutzen der heuristischen Evaluation zu bekommen, der ist, zwei Versionen zu entwickeln und zu testen (vgl. Chapanis im Kapitel Vorüberlegungen). Da Berechnungen zwar möglich, aber zu zeit- und kostenintensiv sind, wird geschätzt. Die Evaluatoren betrachten zwei Parameter: die Reduzierung der Lernzeit und die Erhöhung der Performanz bei Experten. Beide Werte

²³Papierprototypen

Table 2.1 Planned usability engineering program and estimated costs

Life Cycle Stage	UI TASK/Technique	Cost/Technique	Number/Technique	Total Cost (\$)
—	HF lab setup	20,000	1	20,000
Scoping	USER PROFILE User interviews	2,425	2	4,850
Functional specification	TASK ANALYSIS User interviews User questionnaire Usage study	2,425 6,000 6,220	4 1 1	9,700 6,000 6,220
Design	Style guide Simulation test Purchase of UIMS Prototype construction PROTOTYPE TESTING Prototype test Prototype change	16,800 6,220 15,600 5,600 6,220 280	1 3 1 1 3 20	16,800 18,660 15,600 5,600 18,660 5,600
Testing/ implementation	SYSTEM UI TESTING Prototype test Prototype change UI EVALUATION User survey User interview Usage study	6,220 280 6,000 2,425 6,220	3 20 1 3 1	18,660 5,600 6,000 7,275 6,220
Total cost				\$132,185

Abbildung 2.2: Aufstellung von Kosten für den Einsatz verschiedener Analysetechniken. Entnommen aus [Mayhew und Mantei 1994], S. 17 f.

werden aufgrund von Analysen des konkreten Arbeitsablaufs geschätzt. Betrachten könne man weiterhin die Frequenz von Benutzerfehlern und die subjektive Zufriedenheit der Benutzer. In seinen Untersuchungen verzichtete Nielsen jedoch darauf. Nielsen schätzt die Ersparnisse in der Lernzeit von einzelnen Personen und rechnet dann mit der Anzahl der Personen und den einzelnen gesparten Zeiten hoch ([Nielsen 1994a], S. 262). [Tognazzini 1992] bestätigt den Ansatz ‘discount usability’ von Nielsen und gibt ebenfalls einige Richtlinien für die Benutzbarkeitsuntersuchung:

- Verwendung horizontaler Prototypen, um das allgemeine Design zu testen,
- Verwendung vertikaler Prototypen, um das Verhalten bestimmter Programmteile zu testen,
- Prototypen müssen innerhalb von Tagen gebaut werden (nicht innerhalb von Wochen oder Monaten; es sollte dazu ein Werkzeug verwendet werden),
- Methode ‘Lautes Denken’ verwenden,
- Tognazzini’s Paradoxon: ‘Areas of an application where problems are expected have none, while areas known to be perfect are fatally flawed. ([Tognazzini 1992], S. 86).

2.2.1 Einrichtung eines Testlabors

Zu den einmaligen Kosten gehören die Kosten zur Einrichtung eines Testlabors. Ein Testlabor sollte nach [Ehrlich und Rohn 1994] mindestens aus zwei Räumen mit Einwegspiegel bestehen. Entwickler können so zusehen und Probanden fühlen sich nicht beobachtet. Zusätzlich entstehen Kosten für Analysetools, z.B. zum Protokollieren der Tastatureingaben, für Video- und Audio-Equipment und Möbel. Hier sind Kosten von einigen 100 US-Dollar bis zu einigen 100.000 US-Dollar möglich, je nach Budget. [Mantei und Teorey 1988] schätzen die Kosten der Einrichtung eines Labors auf 17.600 US-Dollar.

Aufgrund ihrer Erfahrung stellen [Mayhew und Mantei 1994] ein Beispiel einer Übersicht über die Kosten verschiedener Analysetechniken vor (siehe Abbildung auf Seite 29). Sie stellen z.B. fest, daß die Einrichtung eines Labors etwa 20.000 US-Dollar kostet. Die Erstellung eines Style Guides erfordert 16.800 US-Dollar, das Erstellen eines Prototypen 5.600 US-Dollar. In ihrem Beispiel kommen die Autoren auf eine Summe von 132.185 US-Dollar für ein gutes Software-Ergonomie-Programm. Damit liegt ihre Schätzung in derselben Größenordnung wie die von [Mantei und Teorey 1988] einige Jahre zuvor getätigte Schätzung. Man kann aber auch gute Ergebnisse mit kleineren Budgets erreichen (vgl. Discount Usability Engineering von Nielsen, S. 28). Da die Kosten all dieser einmaligen Investitionen teilweise recht hoch sind, empfiehlt es sich, sie über mehrere Projekte zu verteilen. Ein Labor etwa wird zwar einmal eingerichtet, die Unterhaltungskosten sind jedoch niedrig und das Labor kann von mehreren Projekten genutzt werden. Denkbar ist auch, die Untersuchungen auszulagern und von Fremdfirmen durchführen zu lassen.

2.2.2 Arbeitsanalyse des Anwendungsfeldes

Die Arbeitsanalyse des Anwendungsfeldes eines Produkts ist die wichtigste Aufgabe, die vor der Entwicklung eines Produkts durchzuführen ist. Es reicht nicht, Annahmen über das Produkt und seine Benutzbarkeit zu machen. [Wixon und Wilson 1997] nennen einige Annahmen, die sie häufig bei Entwicklern festgestellt haben (S. 655):

- ‘The product will be usable because it contains online help and pull-down menus.’
- ‘A direct manipulation user interface will make this product user-friendly.’
- ‘The use of a toolbar and icons will ensure that this product is intuitive for new users.’

Eine Arbeitsanalyse hilft, derartigen Annahmen vorzubeugen. Die günstigste Art, eine Arbeitsanalyse durchzuführen, sind Fragebögen über die gewünschten Eigenschaften des zu erstellen oder zu verbessernden Produkts. Auch [Mayhew und Mantei 1994] nennen den Einsatz (strukturierter) Fragebögen und (strukturierter oder freier) Interviews als günstige Methode neben der direkten Beobachtung der Arbeit zur Entwicklung von Benutzerprofilen. Man sollte mit den Anwendern gemeinsam eine Aufgabenanalyse durchführen, um deren Arbeitsabläufe kennenzulernen. Hierzu verwendet man neben Benutzungsstudien²⁴ und Walk-throughs²⁵ auch Feldstudien²⁶ und Benutzbarkeitstests²⁷. Die Kosten jeder dieser Technik kann einzeln geschätzt werden. Hinzu kommen Kosten, die in der Entwicklung begründet liegen, wie die Erstellung eines Style Guides²⁸ und die Prototyperstellung oder -änderung. Zu Beginn entstehen einmalige Kosten, z.B. für die Errichtung eines Labors oder die Anschaffung eines Werkzeugs zur Erstellung eines Prototypen. Hierbei ist allerdings zu beachten, daß ein Werkzeug allein nicht dafür sorgt, daß eine gute Oberfläche entsteht. Wenn es nicht von einem Experten für Schnittstellen verwendet wird, besteht die Gefahr, daß sich ein Programmierer zu sehr von den Möglichkeiten des Werkzeugs bei dem Design der Oberfläche leiten läßt als von den Anforderungen der Benutzer. Die Kosten werden geschätzt, indem man zuerst einen Gesamtplan aufstellt und dann die einzelnen Aufgaben derart detailliert identifiziert (z.B. Materialanschaffungskosten oder Personstunden), daß man deren Kosten schätzen kann.

Ein sehr beeindruckendes Beispiel, in dem offensichtlich keine Untersuchung des Arbeitsablaufes stattgefunden hat, zeigen [Ansorge und Haupt 1997]. Sie untersuchten ein elektronisches Formular zur Erfassung eines Papierformulars. Sie haben die Reihenfolge, in der die Felder des Papierformulars ausgefüllt werden, festgehalten und auf das elektronische Formular übertragen. Das Ergebnis ist in der Abbildung auf Seite 32 zu sehen. Durch eine Untersuchung vor Erstellung des elektronischen Formulars, hätten die aufgetretenen Fehler leicht verhindert werden können. Hier entstehen überflüssige Personalkosten durch die längere Verarbeitungszeit seitens der Mitarbeiter, die die Erfassung der Felder in anderer Reihenfolge gewohnt sind. Auch [Hackos und Redish 1998] liefern mehrere Beispiele, in denen das Arbeitsumfeld der Benutzer vor der Herstellung nicht analysiert wurde, z.B. wurde ein Gerät für Rechtshänder optimiert, das aber aufgrund der Arbeitsumstände nur mit der linken Hand verwendet werden konnte.

[Norman 1989] hat einige Prinzipien aufgestellt, die ein gutes Design sichern sollen (S. 52 f.):

- visibility: Die Möglichkeiten, die ein Anwender hat, sind für ihn auch sichtbar.
- good conceptual model: Die Funktionen sind der Arbeit angepaßt.
- good mappings: Die Zuordnungen von Symbolen zu Funktionen sind leicht erkennbar.

²⁴ Benutzungsstatistiken durch automatisierte Systeme oder Beobachtungen

²⁵ Simulation einer Aufgabenlösung auf einem manuellen oder automatisierten System, um eine Beschreibung der Aufgabe zu erhalten oder die Effizienz der Aufgabe mit dem aktuellen oder dem geplanten System zu ergründen

²⁶ Beobachtung der Benutzer

²⁷ verschiedene Labortechniken, um Performanzdaten zu erhalten, z.B. Fehler und Zeiten

²⁸ Anleitung, die vorschreibt, wie ein Design zu gestalten ist

Abbildung 2.3: Beispiel eines elektronischen Formulars, bei dessen Entwicklung der Arbeitsprozeß des Ausfüllens nicht berücksichtigt wurde. Die Pfeile zeigen die Reihenfolge, in der die Felder auf dem Papierformular ausgefüllt wurden. Entnommen aus [Ansorge und Haupt 1997], S. 6.

- feedback: Das Programm liefert dem Anwender Rückmeldungen über Erfolg oder Mißerfolg von Operationen und über die Dauer von Wartezeiten.

Norman geht außerdem darauf ein, wie wichtig ein gutes Modell für den Erfolg des Designs ist (S. 52 f.). Dies kann nur nach einer Arbeitsanalyse erstellt werden und selbst dann braucht man mehrere Versuche, um ein gutes Design zu entwickeln. Diese Vorgehensweise kann man 1:1 auf die Entwicklung von Software übertragen.

Nach [Mantei und Teorey 1988] sind drei Arten von Benutzer-Studien nötig: die Aufgabenanalyse, die Prototypen-Studie und die Studie mit dem implementierten System. Eine Methode zur Aufgabenanalyse bietet das Contextual Design nach [Beyer und Holtzblatt 1998]. [Wixon und Jones 1991] setzten u. a. die Methode des Contextual Designs ein und erzielten eine enorme Verbesserung des Produkts in seiner zweiten Version. Der Vorteil dieser Methode besteht darin, daß es früh im Entwicklungsprozeß einsetzbar ist, die Probleme, die Anwender bei der Nutzung des Produktes haben, offenlegt und daß es eine gemeinsame Sichtweise der am Entwicklungsprozeß Beteiligten ermöglicht. Ein Beispiel, in dem derartige Studien anscheinend nicht durchgeführt wurden, liefert [Brown 1991]: Anfang der achtziger Jahre gab es immer mehr Beschwerden über das neue Kopierer-Modell von Xerox, weil die Leute mit dem Gerät ihre Arbeit nicht erledigen konnten. Das Problem lag im Design. Der Kopierer hatte derart viele Funktionen, daß ein Anfänger nicht ohne weiteres mit dem Gerät umgehen konnte, z.B. existierten Fehlermeldungen in Form von Nummern. Obwohl bei dem Design Benutzbarkeit an sich beachtet wurde, wurden nicht die spezifischen Arbeitsaufgaben der zukünftigen Benutzer betrachtet. Der Einsatz von Videoaufzeichnungen schaffte hier Abhilfe und zeigte den Entwicklern auf, wo die Probleme lagen. Ein neues Design wurde erarbeitet. Vorher lag die durchschnittliche Behebungsdauer von Papierstaus bei 28 Minuten, mit dem neuen Design bei 20 Sekunden. 'And because such breakdowns are easier to fix, customers are more tolerant of them when they occur.' (S. 107), d.h. die Benutzer waren so zufriedener.

2.2.3 Erstellung eines Prototypen

Viele Manager sind der Meinung, ein Prototyp sei eine fertige Version eines Produkts. Jedoch dient der Prototyp lediglich der Ermittlung der Wünsche der Benutzer und des Grades der Übereinstimmung der Annahmen von Benutzer und Entwickler. Hier läßt sich erkennen, ob die Arbeitsanalyse des Anwendungsfeldes erfolgreich war. [Mantei und Teorey 1988] schätzen die Kosten für die Entwicklung eines Prototypen auf 6.400 US-Dollar. Als Anhaltspunkt geben sie an, man solle einen Prototypen dann einsetzen, wenn die Kosten zur Erstellung höchstens ein Viertel der Projektkosten betragen. Auch für die eigentliche Entwicklung dient der Prototyp als Vorlage. [Cooper 1999] gibt dazu folgende Analogie: 'If the 998th brick deviates by a quarter of an inch, the tower can still probably achieve 1000 bricks, but if the deviation is in the fifth brick, the tower will never get above a couple of dozen' (S. 55). Er fügt hinzu, daß sich mit Papierprototypen sogar bessere Ergebnisse erzielen lassen. 'Software, that actually works – regardless of how badly – exerts a powerful pull on those who must pay for its development' (S. 56). Zudem äußern Anwender ihre Änderungswünsche freier, wenn sie ein Mock-Up vor sich haben statt eines elektronischen Prototypen, sie vermuten, daß ihre Änderungswünsche sonst zu teuer werden und äußern sie deswegen erst gar nicht. Mit einem Beispiel aus seiner Praxis zeigt Cooper, daß die Entwicklung ohne Verwendung des Prototypen in der Programmierung keineswegs zu Verlusten führen muß. 1988 hat er einen Prototypen seiner Software Ruby erstellt, die von Bill Gates gekauft wurde, um sie für Visual Basic zu verwenden. Cooper legte den Prototypen beiseite und begann die Programmierung von Anfang an neu. Trotz der heftigen Einwände des Projektmanagers, der darüber sehr erbost war, wurde das Projekt pünktlich

abgeschlossen.

[Karat 1990] beschreibt ein Projekt, in dem verschiedene Prototypen im Rahmen einer Benutzbarkeits-Untersuchung eingesetzt wurden. Untersucht wurde die Anmeldeprozedur an einem System. Das Ziel des Projekts bestand darin, daß sich 95 % der Benutzer nach dem dritten Versuch fehlerfrei anmelden können sollten. Es wurden drei Tests durchgeführt, die Probanden hatten alle im Schnitt zwei Jahre Erfahrung in ihrer aktuellen Position und waren seit etwa fünf Jahren in dem Unternehmen beschäftigt. Alle drei Tests maßen die Dauer der Ausführung.

- Test 1: Benutzung eines Prototypen, fünf Probanden, Test in Arbeitsumgebung
Ergebnis: Zeit nach 1. Versuch: 195 sec., nach 2. Versuch: 76 sec. und nach 3. Versuch: 48 sec.
- Test 2: Benutzung eines Prototypen, zehn Probanden, Test im Labor.
- Test 3: Benutzung des Realsystems, zwölf Probanden, Test im Labor
Ergebnis: Zeit nach 1. Versuch: 24 sec., nach 2. Versuch: 8 sec. und nach 3. Versuch: 7 sec.

Die Kosten-Nutzenrechnung wurde konservativ durchgeführt, d.h. es wurden folgende Aspekte nicht berücksichtigt: bis zu 12 Anmeldungen wurden pro Tag durchgeführt, es wurden weniger Fehler gemacht, die Helpline wurde weniger genutzt und die Administration des System nahm weniger Zeit in Anspruch. Trotzdem wurde ein Return-Faktor von 2:1 errechnet²⁹. Berechnet wurden die Einsparungen aus Zeitersparnis pro Mitarbeiter, Anzahl der Mitarbeitern und deren Gehalt. Zudem wurde miteinbezogen, daß das Ziel darin bestand, höchstens drei Versuche zu benötigen. Karat weist darauf hin, daß bei größeren Projekten größere Ersparnisse möglich sind. Auch die Returnquote sei bei kleineren Projekten geringer.

2.2.4 Einrichtung und Einarbeitung

Die Arbeitsplätze der Entwickler müssen unter Umständen neu eingerichtet oder mit neuer Software ausgestattet werden, das kann auch die Anschaffung neuer Hardware bedeuten. [Neuhaus und Rock 1999] weisen darauf hin, daß gegebenenfalls auch Kosten durch neu anzuschaffende Hardware anfällt (z.B. 17-Zoll-Monitore). Die Entwickler müssen sich z. B. gegebenenfalls die passenden Software-Bibliotheken anschaffen und müssen eventuell geschult werden. Bevor also eine Zeile Programmcode geschrieben wurde, ist schon ein Teil des Personalbudgets verbraucht worden. Dies ist allerdings notwendig, um spätere Änderungen zu vermeiden, die nach [Pressman 1992] um ein mehrfaches teurer sind. Hinzu kommen, wie [Ansorge und Haupt 1997] feststellen, Kosten für die Schulung der Entwickler in Software-Ergonomie. Außerdem brauchen die Entwickler etwas Zeit, um Erfahrung in der Umsetzung zu bekommen. Natürlich können sich die Entwickler auch einiges an Wissen anlesen, dies sind häufig Kosten, die sich in den Personalkosten verstecken und nicht als Schulungskosten erkannt werden. 'Software-Ergonomie erfordert grundsätzlich andere Denk- und Herangehensweisen, die sich von den gewohnten, technisch dominierten Denkweisen der Software-Entwickler deutlich unterscheiden.' ([Ansorge und Haupt 1997], S. 2) So kann es je nach Vorbildung der Entwickler und nach Umfeld und Projektgröße auch angebracht sein, einen externen Berater zu engagieren. Da zur Zeit auf dem Arbeitsmarkt wenig Software-Ergonomen zur Verfügung stehen, bleibt den Softwareherstellern nur die Möglichkeit, ihre Mitarbeiter auf diesem Gebiet zu schulen (Haupt in einem Interview in [Benning 1999], S. 79).

²⁹Kosten für die Untersuchungen: 20.700 US-Dollar, Einsparungen 41.700 US-Dollar

2.2.5 Sonstige Kosten

Hierunter fallen Kosten wie Raum- und Umbaukosten und Materialverbrauchskosten z.B. für Datenträger ([Viereck 1995]). Außerdem gehören auch Anschaffungskosten für neue Hardware, z.B. Videogeräte, dazu. Diese Kosten können eventuell auf mehrere Projekte verteilt werden, so daß sich die Kosten pro Projekt reduzieren.

2.3 Laufende Kosten

2.3.1 Personal

Zu den laufenden Kosten zählen die Kosten für Probanden. Für allgemeine oder Standard-Software lädt man optimalerweise externe Probanden ein, interne Personen sind durch ihr Wissen zu sehr vorbelastet, als daß ein objektives Ergebnis erzielt werden kann. Zur Auswahl der Probanden ist die Hinzunahme der Abteilungen Marketing und Entwicklung hilfreich. [Ehrlich und Rohn 1994] schätzen nach ihrer Erfahrung einen Zeitaufwand von bis zu sechs Stunden bei einem Durchschnitt von 3,4 Stunden, um einen Probanden zu finden. Die Kosten für die Probanden können entweder in Form eines Stundenlohns oder auch in Form von Geschenken wie T-Shirts und ähnlichem anfallen. Auch eine Agentur kann mit der Beschaffung der Probanden beauftragt werden. [Nielsen 1994a] stellt fest, daß mit 4–5 Probanden 80 %, auf jeden Fall also die wichtigsten, der Probleme entdeckt werden. Dies ist schon mit geringem finanziellen Aufwand möglich. Man sollte nach Möglichkeit zumindest einen Softwareergonom im Team haben, eventuell beschäftigt man einen externen Berater oder man stellt einen neuen Mitarbeiter ein. Wenn die Benutzergruppe sehr inhomogen mit verschiedenen Hintergründen/Wissensständen ist, sind mehr Probanden nötig.

Der Hauptposten der laufenden Kosten sind die Gehälter für Mitarbeiter. Hierzu gehören auch Kosten für die Datenpflege und –erfassung. Dies führt häufig dazu, daß Benutzbarkeit nicht berücksichtigt wird, denn in Projekten, die schon hinter ihrem Zeitplan sind, werden die Projektleiter die Berücksichtigung von Benutzbarkeit in späteren Phasen kaum noch in Betracht ziehen können.

2.3.2 Durchführung

Eine nicht unerhebliche Kostenersparnis entsteht durch frühzeitiges Erkennen und Beheben von Benutzbarkeitsproblemen im Entwicklungsprozeß. Die Ersparnis ergibt sich bei Entwicklung durch das Neuschreiben von Code und dadurch, daß Handbücher nicht mehrfach gedruckt werden müssen, weil nochmals Änderungen darin vorzunehmen sind. Hinzu kommen Kosten durch Schulungen, die schon im Entwicklungsprozeß vorgenommen werden, um Zeit zu sparen. Diese müssen gegebenenfalls wiederholt werden, wenn sich das Produkt zum Ende hin in seiner Benutzung ändert. Nach [Mantei und Teorey 1988] kosten frühe Änderungen ein Viertel soviel wie späte Änderungen und auch nach [Pressman 1992]³⁰ werden Änderungen teurer, je mehr Entscheidungen gefallen sind. Leider halten Manager die Kosten für den iterativen Prozeß³¹ für billiger als die Kosten für ein gutes Design im Vorwege ([Cooper 1999]). Wenn man diese Phase jedoch erst nach Auslieferung des Produktes startet, verliert man so schon von vorneherein viele Benutzer, so daß man nicht das gewünschte Feedback bekommt. Man bekommt auf diese Weise Feedback von den Benutzern, die an dem Produkt und dessen Weiterentwicklung

³⁰Kosten von Änderungen: 1 Einheit in der Planung, 1,5–6 Einheiten während der Entwicklungsphase und 60–100 nach Release

³¹Erstellen, Testen, Fehler melden, Korrigieren, Testen usw.

interessiert sind, jedoch nicht von den potentiellen Nutzern, die ihre Arbeit mit diesem Produkt vereinfachen sollen. Diese wechseln dann eher zu Konkurrenzprodukten.

[McIntyre, Estep und Sieburth 1990] (zitiert nach [Ehrlich und Rohn 1994]) fanden, daß 47-60% des Codes für die Benutzungsoberfläche verwendet wird. Das bedeutet, daß je früher Daten aus Benutzbarkeitsstudien in den Entwicklungsprozeß hineinfließen, desto geringer die Kosten eines Redesigns sind.

[Cooper 1999] kritisiert die allgemeine Haltung der Softwarehersteller, daß ein Produkt lieber unfertig, aber pünktlich ausgeliefert werden sollte. 'Shipping a product that angers and frustrates users in three months is *not* better than shipping a product that pleases the user in six months, as any business person knows full well' (S. 41). Das Hauptproblem liegt seiner Meinung nach darin, daß vor Beginn der Erstellung die Anforderungen nicht ausreichend festgelegt werden. Er werden zwar die Funktionalitäten festgelegt, aber nicht ihre Beziehungen zueinander. 'A shopping bag filled with flour, sugar, milk and eggs is not the same thing as a cake' (S.42). Das Problem liege im Management und in der Projektplanung, die keine realistischen Planungen zuließen. Als Beispiele führt er an:

- Der PalmPilot kam 1990 auf den Markt, sechs Jahre später als der erste Handheld 'Penpoint', war aber der erfolgreichste, weil er genau im richtigen Moment den Markt getroffen habe. Benutzer sind nicht an besonderen Funktionen interessiert, sie wollen lediglich ihre Arbeit durchführen. Die Entwickler von PalmPilot hatten dies erkannt.
- Die Textverarbeitung WriteNow hatte 1987 mehr Marktanteile als MS Word, konnte diese aber nicht halten, weil - aus Zeitgründen, um die Auslieferungstermine einzuhalten - nie die Funktion 'Fußnote' implementiert wurde, diese aber wurde von den Benutzern gewünscht.

[Corbett, Macleod und Kelly 1993] haben in ihrem MUSiC³²-Projekt in Zusammenarbeit mit der Industrie einige Instrumente entworfen, die Benutzbarkeit quantifizierbar machen. Sie untersuchen den Kontext, in dem die Benutzung der Software stattfindet, sowie die Aufgaben, die organisatorischen Gegebenheiten, die technischen Gegebenheiten und die physikalischer Umgebung. Folgende Maße wurden identifiziert:

- analytische Maße
Sie basierend auf Modellen von Benutzerschnittstellen und Aufgaben und sollen früh im Entwicklungsprozeß angewendet werden. Es wurden Editoren und Werkzeuge für die Simulationen, die Analysen und die Modellierungen entwickelt. 25 Maße beschreiben verschiedene Aspekte von Benutzbarkeit (wie Effizienz, Lernanforderungen, kognitive Arbeitslast, Aufgabenangemessenheit und Robustheit).
- Performanz-Maße
Sie messen Effizienz und Aufgabenangemessenheit sowie Lernzeiten. Es wird das Instrument DRUM³³ zur Analyse von Videoaufzeichnungen benutzt.
- Maße für kognitive Arbeitsbelastung
Sie unterteilen sich in objektive und subjektive Maße. Die objektiven messen z.B. Herzfrequenz und Atmung, die subjektiven sind durch Fragebögen zu ermitteln, und fragen z.B. nach der psychischen Belastung.

³²Metrics for Usability Standards in Computing

³³Diagnostic Recorder for Usability Measurement: bietet insbesondere die Möglichkeit, bestimmte Aufzeichnungen schnell wiederzufinden

- Maße für die innere Einstellung der Anwender
Sie vergeben Punkte für allgemeine Benutzbarkeit. Außerdem erstellen sie ein Benutzbarkeitsprofil für verschiedene voneinander unabhängige Aspekte der Benutzbarkeit wie Erlernbarkeit und Effizienz.

So können schon während der Entwicklung die Einsparungen geschätzt werden, die später bei der Nutzung entstehen.

2.3.3 Support

[Cooper 1999] berichtet, daß Microsoft 800 Mio. US-Dollar jährlich für technischen Support ausgibt. Hier besteht also ein großes Potential zum Sparen.[Ehrlich und Rohn 1994] (S.95) schätzen, daß eine Supportanfrage je nach Firma 12 – 250 US-Dollar kostet. Sie fanden verschiedene Beispiele, in denen die Supportkosten durch gute Benutzbarkeit der Produkte gesenkt werden konnten:

- Ein Fall bei Microsoft (in [Reed 1992]): Dort wurden im Schnitt 45 Minuten für die Behebung eines Problems mit dem Drucker benötigt. Die Behebung des Problems erbrachte eine enorme Ersparnis in den Supportkosten.
- Ford hatte ein neues Abrechnungsprogramm für seine Verkäufer eingesetzt (in [Kitsuse 1991]): Vorher benötigten die Verkäufer drei Anrufe beim Support, um anzufangen, damit zu arbeiten. Nachher waren es fast Null. Das ergab eine Ersparnis von 100.000 US-Dollar bei einem Aufwand von 70.000 US-Dollar für ein Labor.
- [Nakamura 1990]: Eine kontextsensitive Hilfe reduzierte die Zeit der Hilfesuche um 40 %.
- [Nussbaum und Neff 1991]: 95 % der Befragten antworteten bei einer Umfrage, sie hätten drei der wichtigsten Funktionalitäten eines Produkts nie benutzt, entweder, weil sie sie nicht kannten oder nicht benutzen konnten.

[Dworschak 2000] berichtet, daß viele Firmen – darunter z.B. Compaq – aus Kostengründen ihre Hotline-Zentrale nach Irland verlegt haben. Ein Beratungsgespräch dauert bis zu zweieinhalb Stunden. Leider werden die Kosten hierfür nicht beziffert. Immerhin scheint sich ein derartiger Aufwand zu lohnen, denn 'Dell weiß aus Umfragen, daß Kunden, die schon mal von einem Techniker gerettet worden sind, eher geneigt sind, den nächsten Computer wieder bei Dell zu kaufen, als diejenigen, die überhaupt nie ein Problem hatten.' (S. 198).

Ein weiteres interessantes Beispiel beschreibt [Mauro 1994]. Eine (nicht genannte) Firma hatte Kosten von über 900.000 US-Dollar dadurch, daß sie einen fehlerhaften Druckertreiber auslieferte. Um das Problem zu beheben, riefen die Kunden die Hotline an, es entstanden enorme Supportkosten. Zusätzlich entstanden Kosten durch das Versenden von Kundenanschriften mit den Updates. Nur wenig Testen hätte diese Kosten gespart. Die Entwickler hatten die Software zwar getestet, hatten aber keine Probleme damit. Hier sieht man wieder das eingangs erwähnte Problem: Entwickler halten sich für Anwender, können aber die Probleme der Endanwender gar nicht erkennen. Mauro schätzt die Service- und Unterhaltungskosten pro Support-Call auf 4–10 US-Dollar. Darin enthalten sind Kosten für Gehalt, Gerätschaften, Versicherung, Computer, Training, Updates, Unterhaltung einer Kundendatenbank, User-Registrierung, Korrespondenz, Telefon usw. Diese Kosten können nicht nur den Gewinn verringern, sondern sogar finanzielle Verluste durch das betreffende Produkt hervorrufen. Gutes Design kann hier die Notwendigkeit von Supportnutzung dramatisch vermindern. Zudem schätzt Mauro die Kosten für die Problembehebung, wenn ein Produkt einmal in Kundenhand ist, höher als die Kosten für den Einsatz

von Software-Ergonomie im Entwicklungszyklus.

[Dray und Karat 1994] stellen den Fall der Neugestaltung eines Help Desk³⁴ vor. Das Hauptproblem bestand in der extrem mangelnden Benutzbarkeit des vorhandenen Systems, hauptsächlich durch Inkonsistenzen in Namensgebung und Navigation, irreführende Fehlermeldungen und mangelnde Anpassungsfähigkeit des Systems an die Aufgaben der Benutzer. Dies bedeutete eine Ausgangssituation, in der neue Benutzer sechs Monate zur Einarbeitung benötigten. Danach arbeiteten die Mitarbeiter weitere 12 Monate unter Anleitung mit dem System. Erst danach konnten sie selbständig damit umgehen. Eine Dokumentation in Papierform wurde teilweise täglich aktualisiert. Eine Systemanalyse ergab die Notwendigkeit, zwei Module zu überarbeiten: das Hilfesystem und die Benutzungsoberfläche. Prototypen wurden im Labor getestet. Das Ergebnis war sehenswert. Mit der Annahme, daß im folgenden Jahr 20 neue Mitarbeiter eingestellt werden sollten, ergaben sich verschiedene Möglichkeiten für Kostenersparnisse. Durch Reduzierung der Trainingszeiten um 35 % konnten jährlich 32.900 US-Dollar gespart werden – ausgehend von derzeitigen Trainingskosten von 94.000 US-Dollar pro Mitarbeiter. Dies wurde dadurch erreicht, daß

- die Zeit zum Lernen der Navigation der 250 Screens, deren Codes und deren Unterschiede untereinander vermindert wurde,
- die Coachingzeit verringert war und
- die Zeit zum Aktualisieren des Handbuchs verringert wurde – 6 US-Dollar x 50 Wochen x 67 Angestellte = 20.100 US-Dollar gespart, da jetzt online aktualisiert wird.

Zudem wurden Kostenersparnisse von 76.400 US-Dollar pro Jahr erwartet durch geringeren Personalwechsel und weniger Nachfragen der Mitarbeiter nach besserem Schulungsmaterial. Weitere Ersparnisse wurden dadurch erwartet, daß die Mitarbeiter weniger Zeit benötigten, um Hilfe im online-Manual zu finden (148.250 US-Dollar), und dadurch, daß die Hilfeanrufe kürzer werden, und es reduzierten sich die Folgekosten der Frustrationen bei den Mitarbeitern (vgl. Edelman in [MORI 1999]). Insgesamt wurden 1.355.293 US-Dollar vor Steuern jährlich gespart – bei einer angesetzten Projektdauer von acht Jahren.

2.3.4 Sonstige Kosten

Hierunter fallen Mieten für Räume und Materialkosten, z.B. für Büromaterial. Außerdem gehören Unterhaltungskosten für Hardware dazu.

2.4 Aktuelle Anwendung: Internetangebote

Seit einigen Jahren verbreitet sich das WWW³⁵ zusehends. Das besondere Problem bei Anwendungen für das Internet besteht darin, daß man die Zielgruppe nicht genau fassen kann, so ist eine besonders gute Vorbereitung nötig. Der für Benutzbarkeit besonders interessante Bereich ist hier der, mit dem man Geld verdienen kann, der E-Commerce. Es existieren auch schon die ersten Untersuchungen über diese ganz neue Art von Software. [Cooper 1999] ist der Meinung, daß viele Websites zu schnell und ohne Planung erstellt werden.

[Nielsen 1998a] schließt sich diesem Urteil an und stellt fest, daß es 39 Stunden dauert, wenn

³⁴Hotline für Mitarbeiter eines Unternehmens

³⁵World Wide Web; Teil des Internets

man zum ersten mal eine Website testet. Hat man bereits Erfahrungen gesammelt, kommt man mit zwei Werktagen aus. Er ließ einige Studenten neun große dänische Sites³⁶ testen:

- 11 Benutzbarkeitskatastrophen,
- 20 ernsthafte Benutzbarkeitsprobleme,
- 29 Schönheitsfehler.

Obwohl jeweils nur ein kleiner Teil jeder Website untersucht wurde, war das Ergebnis beeindruckend. Nach Niensens Erfahrung genügt bereits ein Test mit fünf Probanden, um 80 % der größten Fehler auf der Site und etwa 50 % der Fehler innerhalb jeder besuchten Seite zu finden. Man sollte zumindest die wichtigsten Teile einer Site testen, etwa die, in denen etwas verkauft werden soll. Es stellt sich die Frage, warum diese Seiten trotz solch schlechter Benutzbarkeit besucht werden. [Nielsen 1998b] schätzt, daß 90 % der kommerziellen Seiten schlecht benutzbar sind aufgrund folgender Merkmale:

- aufgeblähte Seiten, die eine lange Ladezeit benötigen;
- Werbung für Produkte, ohne wirklich Informationen über sie zu geben;
- unverständliche Struktur der Site, die vielleicht nach dem Organigramm des Unternehmens gefertigt wurde;
- keine Unterstützung der Navigation;
- ausschweifender Erzählstil, der für Printmedien optimiert wurde (Benutzer lesen nicht, sie scannen);

Hinzu kommt, daß der Service nicht am Ende einer Website zu Ende sein sollte. 'Amazon.com has trained users to expect a confirmation email within a few minutes and the product within a few days, unless the website has warned about shipping delays.' ([Nielsen 1998a], S. 1) Auf der Site von Dell machte Nielsen die Erfahrung, daß die Bestätigungsemail nach drei Tagen kam und der von ihm bestellte Computer nach sechs Wochen kommen sollte. Dadurch verlor Dell einen Umsatz von 3.035 US-Dollar durch einen einzigen Kunden. Es ist schwer abzuschätzen, wie vielen Benutzern das ähnlich ergangen ist. Benutzer verbringen nicht sehr viel Zeit auf schlechten Seiten. Nach Nielsen ist es eigentlich ganz einfach, gute Webseiten zu bauen, da die Konkurrenz sich nicht viel Gedanken darum macht. [Nielsen 1998c] schätzt, daß die Unternehmen mit schlechten Seiten im Internet 50 % der potentiellen Umsätze und 40 % der Folgebesuche verlieren. [Nielsen 1999a] stellt die Ergebnisse seiner Untersuchung über die Gründe von Käufen im Internet bezüglich dänischer Websites dar.

- 83 % der Kunden kaufen im Internet, weil sie die Annehmlichkeit schätzen. Allerdings kaufen Kunden nur in 5 % der Fälle, in denen sie eine E-Commerce-Seite besuchen.
- 63 % schätzen die große Auswahl und die günstigen Preise.
- 52 % freuen sich über schnellen Service und schnelle Lieferung.
- 40 % suchen detaillierte Produktinformationen.
- 39 % schätzen die Atmosphäre, in der die Kunden nicht unter Kaufdruck stehen und
- 36 % bevorzugen die einfachen Zahlungsmethoden.

³⁶Zusammenfassung aller Seiten eines Internetangebots eines Anbieters

Auf die Frage, wie oft die Kunden kaufen, nachdem sie sich informiert haben, antworteten

- 2 % mit ‘fast immer’,
- 14 % mit ‘3/4’,
- 30 % mit ‘etwa jedes zweite Mal’,
- 45 % mit ‘jedes vierte Mal’,
- 9 % mit ‘fast nie’.

Auf die Frage, warum die Kunden die Website besuchten, antworteten nur 5 % damit, daß sie etwas kaufen wollten.

Etwas später stellt [Nielsen 1999b] fest, daß sich noch nicht viel geändert hat. Hier die schlimmsten Webdesign-Fehler:

- Die Funktion des ‘Back-Buttons’, die nach dem Verfolgen eines Links häufigst genutzte Option, wird außer Kraft gesetzt durch das Öffnen eines neuen Browser-Fensters oder die Verwendung des `redirect`³⁷.
- Durch das Öffnen eines neuen Browser-Fensters wird die Konsistenz, die so wichtig für das Gefühl der Sicherheit des Benutzers ist, gestört.
- Der Gebrauch von Radio-Buttons als Aktions-Button stört ebenfalls das Konsistenzgefühl von Benutzern, denn Radio-Buttons ermöglichen eigentlich eine Auswahl aus mehreren Möglichkeiten.
- Das Versetzen von Seiten führt zu ‘broken links’³⁸.
- Langsame Antwortzeiten von Servern führen bei Anwendern zu Ungeduld und Frust.

Zudem stellt [Nielsen 1999c] fest, daß sich Websites oft eher an den Bedürfnissen des Unternehmens orientieren als an den Bedürfnissen der Kunden, z.B., wenn Kunden erst ein Registrierungsformular mit Fragen zur Person ausfüllen müssen, um auf eine Site zu gelangen. Dies sei so, als würde man am Eingang eines Kaufhauses einen bewaffneten Schutzmann postieren, der nur Kunden hereinläßt, die Fragen zu ihrem Stammbaum beantworten.

Eine Studie von Zona Research Inc. in [Seminerio 1998] mit 239 erfahrenen Internetbenutzern besagt, daß 28 % Schwierigkeiten hatten, bestimmte Produkte im Internet zu finden, 62 % gaben innerhalb der letzten zwei Monate auf, danach zu suchen. Es wird darauf hingewiesen, daß der Benutzer im Internet genauso das Gefühl von Erfahrung braucht, wie in der physischen Welt.

Auch [Haupt, Ansorge und Frick 1999] stellen fest, daß Anbietern von E-Commerce-Angeboten ein nicht unerheblicher Teil des Umsatzes verloren geht. In Laboruntersuchungen, in denen Probanden die Aufgabe hatten, bestimmte Artikel zu bestellen, fanden sie, daß 43 % eines möglichen Bestellwertes nicht bestellt wurden aufgrund von Benutzbarkeitsproblemen. Die wichtigsten Mängel:

³⁷automatische Weiterleitung auf eine andere Seite

³⁸Links, hinter denen keine Seite steht, und die eine Fehlermeldung des Browsers auslösen

- unbenutzbare Seiten
 - lange Ladezeiten wegen großer Grafiken
 - inkonsistente Bezeichnungen
 - verwirrende Animationen und zu starker Einsatz von Farben
 - unübersichtliche Navigation und Struktur der Seite
- mangelnde Handlungsorientierung
 - Benutzern ist oft nicht klar, welche Handlungsmöglichkeiten sie haben.
 - Benutzer wissen oft nicht, ob sie bestellt haben.
 - Wahl der falschen Metaphern. Z.B. sollte statt Warenkorb³⁹ besser die aus dem Versandhandel bereits bekannte Metapher Bestellpostkarte verwendet werden.

[Hackos und Redish 1998] kalkulieren die Test⁴⁰-Kosten für eine größere internationale Site auf 59.780 US-Dollar zuzüglich Reisekosten, Kosten für Werbegeschenke und gegebenenfalls Dolmetscherkosten (S. 129). [Klein 2000] stellte im August 2000 immer noch fest, daß es an der Benutzbarkeit von Web-Angeboten hapert, weil z.B. oft eine Bestellmöglichkeit fehlt. Die Internet-Auftritte seien oft geprägt von Informationen, die den Benutzer gar nicht interessieren.

Für die Hersteller von Software gibt es also eine ganze Reihe von Möglichkeiten, Kosten durch benutzergerechte Software zu sparen. Im folgenden soll nun betrachtet werden, wie die Situation auf der Seite der Benutzer aussieht.

³⁹dieser Begriff wird vom statistischen Bundesamt zur Feststellung von Preis-Indizes verwendet

⁴⁰zwei Wochen Dauer, mehrere Benutzer

Kapitel 3

Softwarenutzer

Computerzeitschriften testen Softwareprodukte und geben Kaufempfehlungen. Viele Kunden, gerade auch unerfahrene, lassen sich in ihren Kaufentscheidungen von solchen Veröffentlichungen leiten. Bei konkurrierenden Produkten hat die Benutzbarkeit einen immer stärkeren Einfluß auf die Kaufentscheidung, ein Produkt wird nicht nur nach seiner Funktionalität beurteilt. Das Bewußtsein für gute Benutzbarkeit ist heute im Zuge der immer weiteren Verbreitung von Heimcomputern ein wichtiges Thema. Potentielle Kunden haben die Möglichkeit, sich vor der Kaufentscheidung umfassend zu informieren. Zudem legen sie heute mehr Wert auf Benutzbarkeit. Das Argument 'besser geht's nicht' wird nicht mehr akzeptiert.

Nach [Ehrlich und Rohn 1994] glauben Verkaufs- und Marketingexperten, daß sich Benutzer innerhalb einer Stunde entscheiden, ob ein Produkt benutzbar ist. In einer eigenen Studie haben sie ermittelt, daß ein überzeugter Kunde mindestens vier weitere Kunden beeinflusst, teilweise sogar bis zu zehn. Dies gilt sowohl für Kunden, die eine Software für ihren privaten Bedarf auswählen als auch für diejenigen, die Software für den Einsatz in ihrem Unternehmen anschaffen.

Aus Kostengründen mag es reizvoll sein, Open-Source-Software¹ einzusetzen. Der wesentliche Unterschied zu sonst üblicher Software besteht darin, daß die Einnahmen der Hersteller nicht in den Lizenzgebühren liegen, sondern in den Einnahmen aus Supportdiensten. Jedoch ist bei solchen Produkten aus software-ergonomischer Sicht Vorsicht geboten. Die Benutzbarkeit ist hier im Entwicklungsprozeß selten berücksichtigt. (vgl. Kapitel über OSS im zweiten Teil)

3.1 Einführung

Bei Entwicklungen für den Massenmarkt findet die Einführung der Software nach Projektabschluß statt. So entstehen nach der Anschaffung der Software Kosten für die Einführung, bzw. die Schulung der Mitarbeiter. Die Kosten für die Einführung sind abhängig von mehreren Faktoren. Bei Eigenentwicklungen fallen hier die Kosten für die Schulung der Mitarbeiter an, bei Fremdentwicklungen wird vielleicht eine Hotline eingerichtet. In einer Studie von Diagnostic Research von 1990 (zitiert nach [Ehrlich und Rohn 1994], S. 98) wurde herausgefunden, daß für einen Neuling an einem PC 21 Trainingsstunden benötigt werden: Durch bessere Benutzbarkeit konnte man diesen Aufwand auf elf Stunden reduzieren.

¹Software, die frei verfügbar ist und bei deren Auslieferung der Quellcode mitgeliefert wird

3.1.1 Installation und Systemumstellung

Für die Installation und die Umstellung, bzw. Anpassung des vorhandenen Systems werden Kosten anfallen. Diese sind natürlich abhängig vom vorhandenen System und der neu einzuführenden Software. Gegebenenfalls muß als Voraussetzung für den erfolgreichen Einsatz des neuen Produkts zusätzlich Software oder auch Hardware angeschafft werden. Möglicherweise entstehen auch Kosten durch den Einsatz externer Berater. [Brynjolfsson 1998] stellt fest, daß für jeden Dollar, der für SAP R/3 ausgegeben wird, drei bis vier US-Dollar für einen Unternehmensberater nötig sind. Hiervon lebt eine ganze Branche – die Unternehmensberatungen –, die Kosten tragen die Anwender. Gute Benutzbarkeit könnte hier Abhilfe schaffen, wenn die Software so gut benutzbar wäre, daß ein fachlich qualifizierter Mitarbeiter diese Software bedienen kann, ohne größeres Computerwissen zu besitzen. Man sollte meinen, daß dies den Widerstand der genannten Branche hervorruft, deren Unternehmen teilweise ausschließlich derartige Dienste anbieten. Andererseits, so lange gute Benutzbarkeit noch nicht wirklich gängige Praxis ist, muß diese Branche sich keine Sorgen machen.

3.1.2 Organisationsveränderungen

Unabdingbar für die erfolgreiche Einführung neuer Software ist eine gute Unternehmenskommunikation. Wenn man neue Software in Unternehmen einführt, muß man sich darüber im klaren sein, daß sich daraus Veränderungen in der Organisation ergeben. Dies manifestiert sich aktuell ganz stark in der Diskussion um die Einführung von Wissensmanagement-Systemen in Unternehmen. Weit verbreitet ist die Ansicht, daß es genügt, ein technisches Instrument zur Unterstützung anzuschaffen und den Mitarbeitern zur Verfügung zu stellen. Man kann jedoch Akzeptanz für das neue Produkt nur erreichen, wenn die Voraussetzungen in der entsprechenden Organisation geschaffen wurden, z.B. eben eine offene Kommunikation. Es nützt in diesem Falle nichts, ein besonders benutzerfreundliches Produkt einzuführen. Man muß auch dafür sorgen, daß es benutzt wird, d.h. auch, daß die Mitarbeiter bereit sein müssen, Informationen in dem unterstützenden Instrument abzulegen. Ist die Akzeptanz bei den Mitarbeitern nicht vorhanden, entstehen mindestens die Kosten für die angeschaffte Software, die nun keinen Nutzen bringt. Zusätzlich können Kosten dadurch entstehen, daß die Software doch eingesetzt wird, z.B., weil die Mitarbeiter dazu verpflichtet sind, sie das Produkt aber nicht korrekt einzusetzen und dadurch ihre Aufgaben mit mehr Aufwand erledigen².

3.1.3 Schulung

Im Bereich Schulung lassen sich durch gut benutzbare Softwareprodukte enorme Einsparungen erzielen. Deren Höhe hängt sehr vom konkreten Einzelfall ab. Prinzipiell überlege man sich hierzu, wieviel Zeit gespart wird und wieviele Personen die Benutzung des Produkts pro Jahr erlernen. Daraus läßt sich eine Schätzung der gesparten Kosten ableiten ([Mantei und Teorey 1988]). Schulungskosten lassen sich dadurch reduzieren, daß standardisierte Software verwendet wird. Es lohnt sich, einmal die Kosten einer etwas größeren Schulung aufzuwenden. Beim Anwenden einer Software, die den gleichen Standard benutzt, ist der Schulungsaufwand im folgenden gering, gegebenenfalls gar nicht nötig. (vgl. [Norman 1989], S. 200 ff.)

²In einer Firma wurde ein neues Buchhaltungsprogramm angeschafft, ohne daß die Mitarbeiter daran geschult wurden. Da die Mitarbeiter mit dem neuen Produkt nicht umgehen konnten, haben sie zur Sicherheit, damit keine Buchungen verloren gehen, die Buchungen sowohl im alten als auch im neuen System erfaßt

3.1.4 Verkauf

Kunden erwarten mehr und mehr gute Benutzbarkeit, vielleicht geben sie die Produkte in Zukunft deswegen zurück. Ein Unternehmen sollte Benutzerbeschwerden zum Anlaß nehmen, Probleme umgehend zu beheben. So kann man sinkenden Verkaufszahlen vorbeugen.

3.2 Benutzung

Viele Softwareprodukte werden gar nicht in ihrem vollen Umfang genutzt (konkrete Zahlen liefert [Viereck 1995], S. 185), weil die Anwender nicht mit den Produkten zurechtkommen. Die Lösung sehen die Hersteller in der Minimierung der Benutzungsschnittstelle bei gleichbleibendem Umfang der Software. Dies bedeutet, daß die Funktionalität der Produkte (und damit z.B. der Umfang des benötigten Speicherplatzes) erhalten bleibt. Lediglich die Benutzerschnittstelle wird vereinfacht, so daß nicht mehr auf alle Funktionen zugegriffen werden kann. Dem Benutzer soll so eine einfache Handhabbarkeit suggeriert werden. In der Folge aber wird in Unternehmen eine überproportional wachsende Administration beobachtet, es wird über Personalmangel geklagt, über steigende Instabilität der Anwendungen und sich hinziehende Projekte. ([Computer Zeitung 1999], S. 1) Dies alles verursacht überflüssige Kosten, denen man mit angepaßter und gut benutzbarer Software entgegenwirken kann. Außerdem können Kosten dadurch entstehen, daß Benutzer ein Produkt sabotieren, weil es unbenutzbar ist, so daß sie weiter arbeiten wie bisher, ohne die angebotene Unterstützung, die ja für sie ohnehin keine ist, zu nutzen.

In [Compaq 1999] stellen Mitarbeiter in ihrer Befragung von Computeranwendern fest, daß $\frac{4}{5}$ der über 1.250 Befragten ihre Kollegen frustriert am Computer sehen, über die Hälfte gestand dies für sich selbst ein. Der Arbeitspsychologe Robert J. Edelman erkennt ein neues Krankheitsbild, die technikbezogene Aggression (computer rage). Diese Erkenntnis entstammt einer Studie des Marktforschungsinstituts Market and Opinion Research International, die unter anderem zum Ziel hatte, herauszufinden, ob Technologie als Erleichterung oder als Last empfunden wird. In [MORI 1999] kann man dazu genauere Zahlen nachlesen, z.B. gaben 23 % der Befragten an, daß ihre Arbeit täglich von Computerfehlern verschiedener Art gestört wird, $\frac{2}{5}$ von ihnen überschreiten dadurch Abgabetermine. Ein Drittel von ihnen gab an, daß die Behebung des Problems im Schnitt mindestens eine Stunde dauert. Dies verursache auf die Dauer enorme Kosten. Je höher der Stundenlohn ist, desto höher sind die anfallenden Kosten.

Angestellte in einem Unternehmen benutzen schlechte Software, weil sie diese benutzen müssen, meint [Cooper 1999]. Die Kosten, die hier entstehen, kann man sehr schlecht messen, sie entstehen indirekt durch geringere Produktivität, Fehler in der Arbeit und auch dadurch, daß Mitarbeiter das Unternehmen verlassen. Das Einstellen und Einarbeiten eines Mitarbeiters kostet. Es müssen Anzeigen geschaltet und Bewerbungsgespräche geführt werden. Hier hat jedes Unternehmen seine eigenen Erfahrungen, welches Budget dazu veranschlagt werden muß. Gerade in der Softwarebranche besteht zur Zeit ein hoher Bedarf an Arbeitskräften, so daß es hier gewiß zu einem hohen Kostenfaktor kommt.

[Brodbeck 1991] untersuchte die Fehlerbewältigungsdauer bei Büroarbeiten am Computer. Er fand heraus, daß im Durchschnitt von 85 Minuten Arbeit am Computer die durchschnittliche Fehlerbewältigungsdauer 8,5 Minuten betrug, also etwa 10 %. Hochgerechnet auf einen Arbeitstag ergeben sich daraus 50 Minuten, und zwar trotz der Tatsache, daß die Probanden erfahrene³ Computerbenutzer waren. Dies kann man hochrechnen auf das gesamte Unternehmen. Bei an-

³ein bis zwei Jahre Erfahrung

genommenen 300 Mitarbeitern, einen Personalkostenaufwand von 200,- DM pro Tag und 200 Arbeitstagen pro Jahr ergeben sich so überflüssige Kosten von 12 Mio. DM allein für die Behebung der Fehler. Nicht mitgerechnet sind hier die Kosten, die durch den Streß der Mitarbeiter entstehen, dadurch, daß ihre Konzentration leidet und sie weitere Fehler machen. Bei 70 % der Fehler, die mehr als zehn Minuten Berichtigungszeit brauchten, hatten die Probanden sich mit 'mittelstarken, starken oder sehr starken negativen emotionalen Reaktionen' zu arrangieren (S. 82). Eine Befragung der Probanden ergab, daß in 36 % aller Fälle ein Kollege zu dem Problem befragt wurde, in 14,6 % der Fälle wurde erst eine andere Hilfe verwendet und dann ein Kollege gefragt. Hier entstehen zusätzlich Kosten dadurch, daß der helfende Kollege nicht weiter seiner Arbeit nachgehen kann. 'Den Kollegen zeichnet aus, daß er die Arbeitsaufgabe am besten kennt und unmittelbar erreichbar ist' (S. 88). Zudem fand Brodbeck heraus, daß kommerzielle Handbücher kaum zu Hilfe genommen wurden, weil sie entweder nicht erreichbar waren oder die Informationen nicht verständlich darstellen und die Mitarbeiter lieber ihre eigens angefertigten Unterlagen verwendeten. [Mantei und Teorey 1988] stellen fest, daß sich durch die Vermeidung von Benutzerfallen⁴ Kosten einsparen lassen. [Bulkeley 1992] stellt zudem fest, daß versteckte Supportkosten von zwischen 6.000 und 15.000 US-Dollar pro PC existieren, davon entfallen etwa 2000 – 6500 US-Dollar Kosten auf die 'Tips unter Kollegen'.

[Dzida 1999] empfiehlt neben dem Abschluß eines Wartungsvertrages, auch einen Pflegevertrag abzuschließen, der die Beseitigung ergonomischer Mängel regelt. Außerdem könne man vom Hersteller eine Erklärung nach DIN EN 45014 verlangen. Hierin erklärt der Hersteller, daß er in seiner Software die ergonomischen Normen nach ISO 9241, Teile 10 bis 17, eingehalten hat. Diese Erklärung soll den Arbeitgebern als Absicherung dienen für die rechtlichen Folgen (siehe Kapitel 3.3).

3.2.1 Support

Auch bei den Softwarenutzern entstehen Supportkosten. In der [Computer Zeitung 2000] ist zu lesen, daß die R+V-Versicherung 40 Mitarbeiter in der Support-Abteilung beschäftigt. Eine Mitarbeiterin der Advance Bank berichtet, daß ein Support-Mitarbeiter dort mit einem Jahreseinkommen von 80.000 DM rechnen kann.

3.3 Rechtliche Rahmenbedingungen

[Mauro 1994] stellt fest, daß schon in den 60er Jahren Klagen anerkannt wurden, in denen gesundheitliche Schäden bei den Benutzern durch schlechtes Design ausgelöst wurden. Einige Firmen wurden so zum ersten Mal mit der (Un-)Benutzbarkeit ihrer Produkte konfrontiert. Eine Klage kostete im Minimum 1 Mio. US-Dollar (an Gebühren für Experten, Anwälte, Dokumentationen u.ä.) Diese Situation erzwang die Einsicht in die Notwendigkeit, für Benutzbarkeit ein Budget zur Verfügung zu stellen. Leider sind die Kosten nicht immer komplett dokumentiert, teilweise mit der Absicht, Vorstände und Aktionäre nicht zu verunsichern.

Die rechtliche Seite benutzbarer Software gewinnt immer stärker an Bedeutung. Seit dem 1. Januar 2000 ist die [Bildschirmarbeitsverordnung 1996] vollständig in Kraft getreten. Sie ist die Umsetzung einer Richtlinie des Europäischen Rates [EG-Richtlinie 1990] zur Festlegung von Mindestvorschriften bezüglich der Sicherheit und des Gesundheitsschutzes bei der Arbeit an

⁴Fehler, die dadurch entstehen, das Benutzer eine bestimmte Reaktion eines Systems gewohnt sind, das neue System aber anders reagiert. So machen Benutzer immer wieder dieselben 'Fehler'.

Bildschirmgeräten; sie gilt für sämtliche europäische Firmen, die Software einsetzen. Der Arbeitgeber trägt die rechtliche Verantwortung für die Einhaltung der Richtlinie für die Software, die er seinen Mitarbeitern zur Verfügung stellt. Arbeitgeber werden in Zukunft möglicherweise versuchen, diese Verantwortung auf die Softwarehersteller abzuwälzen. Hier besteht ein großer Bedarf an einer Art Zertifizierung, z.B. durch den TÜV⁵, so daß sich ein Arbeitgeber sicher sein kann, daß das von ihm eingesetzte Produkt den Richtlinien entspricht. Hier werden Grundsätze für die Gestaltung von Mensch-Maschine-Schnittstellen festgelegt. Diese Schnittstelle muß benutzerfreundlich sein, dies heißt im besonderen, daß z.B. die Software der auszuführenden Tätigkeit angepaßt sein muß und daß die Systeme dem Benutzer die Beseitigung eines Fehlers mit begrenztem Arbeitsaufwand gestatten müssen. Diese Anforderungen werden nach Meinung von Haupt in einem Interview in [Benning 1999] von den meisten Produkten nicht erfüllt. Eine von der Zeitschrift "Computerbild" in Auftrag gegebene Studie des TÜViT⁶ hat nach Angaben von [Kabel 1.de 1999a] festgestellt, daß z.B. die Software Microsoft Office 2000 die Bildschirmrichtlinie nicht erfüllt. Dies gelte für 80 % der auf dem Markt erhältlichen Produkte [Kabel 1.de 1999b]. Für einen Arbeitgeber könne dies bis zu 50.000 DM Strafe bedeuten.

⁵Technischer Überwachungsverein

⁶TÜV-Informationstechnik

Kapitel 4

Zusammenfassung und Ausblick

In dieser Arbeit wurde gezeigt, daß ein Bewußtsein für die Vorteile, Benutzbarkeit in den Entwicklungsprozeß miteinzubeziehen auch im Sinne einer Kosten–Nutzen–Abschätzung durchaus schon vorhanden ist. Auch gibt es schon viele konkrete Berechnungsmethoden, die recht genaue Aussagen über die Erfolge der zur Verbesserung der Benutzbarkeit eingesetzten Entwicklungsmethoden erlauben. Das Argument, man könne den Nutzen nicht voraussagen, ist also widerlegt. In Zukunft sollte untersucht werden, woran es wirklich liegt, daß die vorhandenen Methoden wenig eingesetzt werden.

Unterstützt wird der Einsatz effektiver Maßnahmen durch die Existenz von Leitfäden, die ein gewisses Maß an Benutzbarkeit sicherstellen und die Arbeit in Projekten erleichtern, an denen kein Software–Ergonom beteiligt werden kann. Ein Beispiel für einen derartigen Leitfaden ist die Bildschirmrichtlinie, die viele der von Software–Ergonomen geforderten Anforderungen enthält. Zudem gibt es viele Fallbeispiele, die als Entscheidungshilfe herangezogen werden können. In einigen Artikeln fanden sich Hinweise auf die Existenz von Werkzeugen zur Abschätzung der Erfolge von Investitionen in Benutzbarkeit. Diese Werkzeuge sollten einmal zusammengestellt und auf ihre Anwendbarkeit hin untersucht werden.

Der Einsatz software–ergonomischer Maßnahmen ergibt bei den Herstellern von Software direkt einen Kostenvorteil durch verminderte Support–Kosten und höhere Verkaufszahlen. Bei jeder konkreten Anwendung muß man jedoch individuell Betrachtungen anstellen, um den Nutzen zu schätzen. Der mangelnden Sensibilisierung für Benutzbarkeit im Herstellungsprozeß liegt häufig das Unverständnis der Programmierer und Projektleiter für die Bedürfnisse der Benutzer zugrunde. Obwohl dieses Hindernis relativ leicht durch eine Arbeitsanalyse beseitigt werden kann, wird diese selten durchgeführt, einfach, weil die Notwendigkeit nicht gesehen wird. Häufig trifft man noch auf die Einstellung, Programmierer seien Benutzer. Vielen Programmierern ist scheinbar immer noch nicht bewußt, daß sie keine typischen Benutzer sind und daß es Benutzer gibt, die wenig oder gar keine Erfahrung im Umgang mit Computern haben.

Ein Ansatz zur Verbesserung dieser Situation liegt in der Ausbildung der Personen, die am Herstellungsprozeß von Software beteiligt sind, d. h., sowohl Programmierer als auch Manager müssen auf diesem Gebiet geschult werden. In Zukunft sollte man schon in der Lehre das Bewußtsein für Software–Ergonomie der Studenten in der Ausbildung in Softwaretechnik und Betriebswirtschaft wecken.

Das Entstehen des Internet ist eine willkommene Möglichkeit, zu überprüfen, ob Benutzbarkeit

ihren Stellenwert erhalten hat. Leider ist oft festzustellen, daß selbst die Internetauftritte, die darauf ausgerichtet sind, etwas zu verkaufen, oft eher durch "poppige" Erscheinung als durch einfache Benutzbarkeit auffallen. Obwohl Untersuchungen gezeigt haben, daß sich durch gute Benutzbarkeit gute Gewinne erzielen lassen, kann man an neueren Anwendungen im Bereich des Internet und des E-Commerce nicht erkennen, daß sich die Ergebnisse dieser Untersuchungen in der Praxis durchgesetzt haben. Immer noch gibt es schicke und bunte aber unbenutzbare Seiten im Internet, die einigen Firmen einiges an Verlusten durch nicht getätigte Käufe einbringen. Man sollte in Zukunft das Bewußtsein der zuständigen Personen in den entsprechenden Unternehmen für diese Vorgänge schärfen. In diesem Bereich können Unternehmen sehr leicht und schnell feststellen, ob die Kunden mit der angebotenen Software zufrieden sind. Gerade hier ist es besonders einfach, bei Unzufriedenheit den Anbieter zu wechseln. Unternehmen, die schon lange am Markt sind, können ihre Umsätze im Internet mit denen aus ihrer Erfahrung vergleichen. Hier kann jedes Unternehmen recht einfach – zumindest im nachhinein – den Erfolg der Maßnahmen messen. Es gibt auch schon erste Versuche dazu: Kürzlich war in der Presse zu lesen, daß ein großes Kaufhaus einen breit angelegten Benutzertest mit seinem Internetangebot starten will.

Wie sieht es nun auf der Seite der Benutzer aus? Sie werden selbstbewußter und achten darauf, benutzbare Produkte zu erwerben - sowohl für den privaten als auch für den gewerblichen Gebrauch. Durch die neu geschaffenen rechtlichen Grundlagen tragen nun Arbeitgeber die Verantwortung für gute Software am Arbeitsplatz ihrer Mitarbeiter. Diese Verantwortung werden sie an die Hersteller abgeben wollen. Es gibt bereits die Möglichkeit, Software vom TÜV prüfen zu lassen. Vielleicht wird es in Zukunft Zertifikate geben, die den Käufern die Entscheidung für oder gegen ein bestimmtes Produkt erleichtern.

Literaturverzeichnis

- [Ansorge und Haupt 1997] Ansorge, Peter und Uwe Haupt (1997). *Ergonomie-Reviews und Usability-Testing als Beratungs- und Qualifizierungsinstrumente*. <http://www.tzi.de/tzi/berichte/TZi002.ps.zip> (Übersichtsseite: <http://www.tzi.de/tzi/berichte/kurzfassungb002.html>) Letzter Aufruf: 12.10.1999
- [Aykin 1994] Aykin, Nuray (1994). *Software Reuse: A Case Study on Cost-Benefit of Adopting a Common Software Development Tool* In: Bias, R. G. und D. J. Mayhew (Hrsg.) (1994). *Cost-Justifying Usability*. Academic Press, Boston, S. 177–202
- [Bailey, Knox und Lynch 1988] Bailey, W. A., Knox, S. T. und E. F. Lynch (1988). *Effects of interface design upon user productivity*. In: Proc. ACM CHI '88 Conf. (Washington, DC, 15-19 May), S. 207–212.
- [Baumann 2000] Baumann, R. (2000). *Schuld sind die Entwickler*. In: Netinvestor, 7/00, S. 85
- [Bennet 1991] Bennet, J. L. (1991). *Accepting the Challenge*. In: Human Computer Interface Design: Success Cases, Emerging Methods, and Real-World Context. Boulder, Co., July 24-26, S. 375–385
- [Benning 1999] Benning, M. (1999). *Form follows me*. In: c't, Heft 18, S. 78–791
- [Beyer und Holtzblatt 1998] Beyer, H. und K. Holtzblatt (1998). *Contextual Design*. Morgan Kaufmann Publishers, San Francisco
- [Bias und Mayhew 1994a] Bias, R. G. und D. J. Mayhew (Hrsg.) (1994). *Cost-Justifying Usability*. Academic Press, Boston
- [Bias und Mayhew 1994b] Bias, R. G. und D. J. Mayhew (1994). *Organizational Inhibitors and Facilitators*. In: Bias, R. G. und D. J. Mayhew (Hrsg.) (1994). *Cost-Justifying Usability*. Academic Press, Boston, S. 287–318
- [Bias und Mayhew 1994c] Bias, R. G. und D. J. Mayhew (1994). *A Place at the Table* In: Bias, R. G. und D. J. Mayhew (Hrsg.) (1994). *Cost-Justifying Usability*. Academic Press, Boston, S. 319–326
- [Bias 1994] Bias, R. G. (1994) *Wherefore Cost Justification of Usability: Pay Me Now or Pay Me Later - But How Much?* In: Bias, R. G. und D. J. Mayhew (Hrsg.) (1994). *Cost-Justifying Usability*. Academic Press, Boston, S. 3–8
- [Bildschirmarbeitsverordnung 1996] *Verordnung über Sicherheit und Gesundheitsschutz bei der Arbeit an Bildschirmgeräten*. vom 20.12.1996 <http://www.sozialnetz-hessen.de/ergo-online/recht/Bilscharbv/bilscharbv.htm> Letzter Aufruf: 09.01.2000

- [Bosert 1991] Bosert, J. L. (1991). *Quality Functional Deployment: A Practitioner's Approach*. ASQC Quality Press, New York
- [Brodbeck 1991] Brodbeck, F. (1991). *Fehlerbewältigungsdauer und die Nutzung von Unterstützungsmöglichkeiten*. In: Frese, Michael und Dieter Zapf (Hrsg.) (1991). Fehler bei der Arbeit mit dem Computer: Ergebnisse von Beobachtungen und Befragungen im Bürobereich. Huber, Bern (1991), S. 80–94
- [Brooks 1994] Brooks, R. (1994) *Justifying Prepaid Human Factors for User Interfaces*. In: Bias, R. G. und D. J. Mayhew (Hrsg.) (1994). Cost–Justifying Usability. Academic Press, Boston, S. 273–285
- [Brown 1991] Brown, J. S. (1991). *Research that reinvents the corporation*. In: Harvard Business Review 69 (1), S. 102–111
- [Brynjolfsson 1993] Brynjolfsson, E. (1993). *The productivity paradox of information technology*. In: Commun. ACM 36, 12 (1993), S. 66–77
- [Brynjolfsson 1998] Brynjolfsson, E. und L. M. Hitt (1998). *Beyond the productivity paradox*. In: Commun. ACM 41, 8 (1998), S. 49–55
- [Bulkley 1992] Bulkley, W. M. (1992). *Study finds hidden costs of computing*. In: The Wall Street Journal, 2 November, B4
- [Chapanis 1991a] Chapanis, A. (1991). *The business case for human factors in informatics*. In: Shakel, B. und S. Richardson (Hrsg.), Human Factors for Informatics Usability. Camebridge University Press, Camebridge, U.K., S. 21–37
- [Chapanis 1991b] Chapanis, A. (1991). *Evaluating Usability*. In: Shakel, B. und S. Richardson (Hrsg.), Human Factors for Informatics Usability. Camebridge University Press, Camebridge, U.K., S. 359–395
- [Compaq 1999] Compaq Press Release (1999). *Employees Get 'It' Out Of Their Systems*. <http://compaq.co.uk/news/temp/2e12.htm> Letzter Aufruf: 29.06.1999
- [Computer Zeitung 1999] Computer Zeitung Nr. 37 (1999). *Standardpakete erschlagen den Anwender*.
- [Computer Zeitung 2000] Computer Zeitung Nr. 3 (2000). *Helpdesk-Mitarbeiter brauchen starke Nerven und ein offenes Ohr*.
- [Computerwoche 2000] Computerwoche Nr. 8 (2000). *Freie Software verlangt aufgeklärte Anwender.*, S. 21–22
- [Conklin 1991] Conklin, P. F. (1991). *Bringing Usability Effectively into Product Development* In: Human Computer Interface Design: Success Cases, Emerging Methods, and Real–World Context. Boulder, Co., July 24–26, S. 367–385
- [Cooper 1999] Cooper, A. (1999). *The Inmates are Running the Asylum*. Sams, Indianapolis, S. 42–58
- [Corbett, Macleod und Kelly 1993] Corbett, M., M. Macleod und M. Kelly (1993). *Quantitative Usability evaluation – The ESPRIT MUSiC Project*. In: Proc. of HCI International Conference, Florida, USA, S. 313–318

- [Cox und Pendley 1994] Cox, M. E. und W. L. Pendley (1994). *UPAR Analysis: Dollar Measurement of a Usability Indicator for Software Products*. In Bias, R.G. und D. J. Mayhew (Hrsg.) (1994). *Cost-Justifying Usability*. Academic Press, Boston, S. 145–158
- [DeMarco 1997] DeMarco, T. (1997). *Warum ist Software so teuer? ...und andere Rätsel des Informationszeitalters*. Carl Hanser Verlag, München Wien
- [Diverse Autoren 1991] Zuhoff, S., W. R. Sutherland, J. S. Brown, I. Wladawsky-Berger, P. M. Senge, C. Argyris, D. A. Schon, C. U. Ciborra, T. Winograd, B. R. Guile, R. E. Kraut, K. Yoshida, P. A. Fisher, M. Nakanishi und A. S. Wassermann (1991). *Debatte: Can Research reinvent the corporation?* In: Harvard Business Review 69 (2), S. 164–175
- [Dray und Karat 1994] Dray, S. M. und C.-M. Karat (1994). *Human Factors Cost Justification for an Internal Development Project*. In: Bias, R.G. und D. J. Mayhew (Hrsg.) (1994). *Cost-Justifying Usability*. Academic Press, Boston, S. 111–142
- [Dworschak 2000] Dworschak, M. (2000). *Die Lust am Störfall*. In: Spiegel, Nr. 8, 2000, S. 196–198
- [Duden Informatik 1993] *Duden Informatik*, Dudenverlag, 1993
- [Dzida 1999] Dzida, W. (1999). *Software-Ergonomie: Softwarenutzung in der betrieblichen Praxis verbessern*. In: Information Management & Consulting Nr. 14, 3, S. 23–27
- [EG-Richtlinie 1990] *Richtlinie des Rates über die Mindestvorschriften bezüglich der Sicherheit und des Gesundheitsschutzes bei der Arbeit an Bildschirmgeräten (90/270/EWG)* vom 29. Mai 1990
- [Ehrlich und Rohn 1994] Ehrlich, K. und J. A. Rohn (1994). *Cost-Justification of Usability Engineering: A Vendor's Perspective*. In: Bias, R.G. und D. J. Mayhew (Hrsg.) (1994). *Cost-Justifying Usability*. Academic Press, Boston, S. 73–110
- [Grudin, Ehrlich und Shriner 1987] Grudin, J., S.F. Ehrlich und R. Shriner (1987). *Positioning human factors in the user interface development chain*. In: Proc. ACM CHI+GI'87 Conf. (Toronto, Canada, 5–9 April), S. 125–131.
- [Hackos und Redish 1998] Hackos, J. T. und J. C. Redish (1998) *User and task analysis for interface design*. Wiley, New York, S. 11–14 und S. 111–126
- [Harrison, Hennemann und Blatt 1994] Harrison, M., R. L. Henneman und L. A. Blatt (1994). *Design of a Human Factors Cost-Justification Tool*. In: Bias, R.G. und D. J. Mayhew (Hrsg.) (1994). *Cost-Justifying Usability*. Academic Press, Boston, S. 203–241
- [Haupt, Schulze, Ansorge und Frick 1998] Haupt, U., M.-O. Schulze, P. Ansorge und G. Frick (1998). *Benutzbarkeit von E-Commerce-Angeboten. Internetshopping Report 98/99. Die große Nutzerumfrage. Käufer, Produkte, Zukunftsaussichten*. Symposium Publishing, Düsseldorf, S. 170–176 PDF Benutzbarkeit von E-Commerce-Angeboten 129 KB http://selab24.informatik.uni-bremen.de/docs/Haupt_Schulze_Ansorge_Frick_1998_Benutzbarkeit_von%20E-Commerce-Angeboten_Fehlklicks_kosten_%20Geld.PDF Letzter Aufruf: 25.10.1999
- [Haupt, Ansorge und Frick 1999] Haupt, U., P. Ansorge und G. Frick (1999). *Fehlklicks kosten Geld*. http://selab24.informatik.uni-bremen.de/docs/Haupt_Ansorge_Frick_1999_Fehlklicks_kosten_Geld.pdf Letzter Aufruf: 12.10.1999

- [House und Price 1991] House, C. H. und R. L. Price (1991). *The return map: Tracking product teams*. In: Harvard Business Review 69(1), S. 92–100
- [Johnson 2000] Johnson, Jeff. (2000). *GUI Bloopers*. Morgan Kaufmann Publishers, London
- [Kabel 1.de 1999a] Kabel 1 (1999). *Schwarzes Jahr für Microsoft?*
<http://www.kabel1.de/computer/1999/1208/microsoft/> Letzter Aufruf: 22.08.2000
- [Kabel 1.de 1999b] Kabel 1 (1999). *Das Problem der Arbeitgeber ...*
<http://www.kabel1.de/computer/1999/1208/microsoft2/> Letzter Aufruf: 22.08.2000
- [Karat 1990] Karat, C.–M. (1990). *Cost benefit analysis of iterative usability testing*. In: Diaper, D. (Hrsg.). Human–Computer Interaction – Proc. Interact '90, Elsevier, Amsterdam, S. 351–356
- [Karat 1994] Karat, C.–M. (1994). *A Business Approach to Usability Cost Justification*. In: Bias, R.G. und D. J. Mayhew (Hrsg.) (1994). *Cost–Justifying Usability*. Academic Press, Boston, S. 45–70.
- [Karat 1997] Karat, C.–M. (1997). *Cost–Justifying Usability Engineering in the Software Lifecycle*. In: Helander, M. T., T. K. Landauer und P. Prasad V. Prabhu (Hrsg.) (1997). *Handbook of Human Computer Interaction*, Elsevier 1997 (2nd Edition), S. 767–778
- [Klein 2000] Klein, Pit. (2000). *Leere Körbe*. In: internet world, August, S. 38–39
- [Kling 1999] Kling, Rob (1999). *What is Social Informatics and Why Does it Matter?* In: D–Lib Magazine January 1999 Vol.5 No.1:
<http://www.dlib.org:80/dlib/january99/kling/01kling.html> Letzter Aufruf: 10.10.1999
- [Kitsuse 1991] Kitsuse, A. (1991). *Why aren't computers...* Across the Board (October) 28, S. 44–48
- [Kuniavsky 2000] Kuniavsky, M. (2000). *It's the User, Stupid*
<http://sendmail.net/?feed=interviewkuniavsky> Letzter Aufruf: 18.04.2000
- [Lederer und Prasad 1992] Lederer, A. L. und J. Prasad (1992). *Nine management guidelines for better cost estimating*. In: Communications of the ACM 35, 2 (February), S. 51–59.
- [Lund 1997] Lund, A. M. (1997). *Another Approach to Justifying the Cost of Usability*. In: Interactions, May + June, S. 48–56.
- [Mantei und Teorey 1988] Mantei, M. M. und Teorey, T. J. (1988). *Cost–benefit analysis for incorporating human factors in the software lifecycle*. In: Communications of the ACM 31, 4. April, S. 428–439.
- [Martin und McClure 1983] Martin, J. und C. McClure (1983). *Software Maintenance: The Problem and its Solution*. Prentice Hall, New Jersey
- [Mauro 1994] Mauro, C. L. (1994). *Cost–Justifying Usability in a Contractor Company* In: Bias, R.G. und D. J. Mayhew (Hrsg.) (1994). *Cost–Justifying Usability*. Academic Press, Boston, S. 123–141
- [Mayhew und Mantei 1994] Mayhew, D. J. und M. Mantei (1994). *A Basic Framework for Cost–Justifying Usability Engineering*. In: Bias, R.G. und D. J. Mayhew (Hrsg.) (1994). *Cost–Justifying Usability*. Academic Press, Boston, S. 9–43

- [Mayhew 1994] Mayhew, D. J. (1994). *Cost-Benefit Analysis of Upgrading Computer Hardware*. In: Bias, R.G. und D. J. Mayhew (Hrsg.) (1994). *Cost-Justifying Usability*. Academic Press, Boston, S. 159-175
- [Mayhew 1999] Mayhew, D.J. (1999). *The Usability Engineering Lifecycle*. Morgan Kaufman Publishers, Inc., San Fransisco, S. 449-481
- [McIntyre, Estep und Sieburth 1990] McIntyre, F., K. W. Estep und J. M. Sieburth (1990). *Cost of user-friendly programming*. In: *Journal of Fourth Application and Research* (2), S. 103-115
- [MORI 1999] MORI Market and Opinion Research International (1999). *Rage Against the Machine. The Impact an Implications of Computer Rage on UK Business*. Unabhängige von Compaq in Auftrag gegebene Studie
- [Nakamura 1990] Nakamura, R. (1990). *The X Factor*. In: *Infoworld* (19.November), S. 51-55
- [Neuhaus und Rock 1999] Neuhaus, K. und B. Rock (1999). *EnjoySAP - Einsatz ohne Reue?* In: *Information Management & Consulting* Nr. 14, 3, S. 7-10
- [Nielsen, Mack, Bergendorff und Grischowsky 1986] Nielsen, J., R. L. Mack, K. H. Bergendorff und N. L. Grischowsky (1986). *Integrated software in the professional work environment: Evidence from questionnaires and interviews*. In: *Proceedings of the ACM CHI '86 Conference*, Boston, Massachusetts, 13-17 April, S. 162-167
- [Nielsen und Landauer 1993] Nielsen, J. und T.K Landauer (1993). *A Mathematical Model of the finding of usability problems*. In: *Proc. INTERCHI '93*, S. 206-213
- [Nielsen 1993b] Nielsen, J. (1993) *Usability Engineering* Academic Press, Boston
- [Nielsen 1994a] Nielsen, J. (1994). *Guerrilla HCI: Using Discount to Penetrate the Intimidation Barrier* In: Bias, R.G. und D. J. Mayhew (Hrsg.) (1994). *Cost-Justifying Usability*. Academic Press, Boston, S. 245-272; auch zu finden unter <http://www.useit.com/papers/guerrilla.hci.html>
- [Nielsen 1998a] Nielsen, J. (1998). *Cost of User testing A Website*. <http://www.useit.com/alertbox/980503.html> Letzter Aufruf: 21.10.1999
- [Nielsen 1998b] Nielsen, J. (1998). *The Web Usage Paradox: Why Do People Use Something This Bad?* <http://www.useit.com/alertbox/980809.html> Letzter Aufruf: 21.10.1999
- [Nielsen 1998c] Nielsen, J. (1998). *Failure of Corporate Websites*. <http://www.useit.com/alertbox/981018.html> Letzter Aufruf: 21.10.1999
- [Nielsen 1999a] Nielsen, J. (1999). *Why People Shop on the Web*. <http://www.useit.com/alertbox/990207.html> Letzter Aufruf: 21.10.1999
- [Nielsen 1999b] Nielsen, J. (1999). *The Top Ten New Mistakes of Web Design*. <http://www.useit.com/alertbox/990530.html> Letzter Aufruf: 21.10.1999
- [Nielsen 1999c] Nielsen, J. (1999). *Web Search: Believe the Data* <http://www.useit.com/alertbox/990711.html> Letzter Aufruf: 21.10.1999
- [Norman 1989] Norman, D. A. (1989). *The Design of Everyday Things*. Currency Doubleday, New York (Reprint)

- [Nussbaum und Neff 1991] Nussbaum, B. und R. Neff (1991). *I can't work this thing*. In: Business Week (29 April), S. 58-66
- [Person et al. (keine Angabe)] Person, J., H. Brodin, O. Lorentsen, K.-G. Hem, R. Andrich, M. Ferrario, T. van Beekum und W. Oortwijn. *Cost-Utility Analysis of Assistive Technology – Report on the CERTAIN project*. <http://www.stakes.fi/tidecong/824perss.html> Letzter Aufruf: 25.11.1999
- [Pressman 1992] Pressman, R. S. (1992). *Software Engineering: A Practitioner's Approach*. McGraw Hill, New York, S. 98–135 und S. 440–442
- [Reed 1992] Reed, S. (1992). *Who defines Usability? You do!* In: PC Computing (Dec), S. 220–232
- [Rothman 1997] Rothman, Johanna. (1997). *Is Your Investment in Quality and Process Improvement Paying Off?* <http://www.jrothman.com/Papers/QW97.html> Letzter Aufruf: 11.09.2000
- [Scerbo 1991] Scerbo, M. W. (1991). *Usability engineering approach to software quality*. In: Annual Quality Congress Transactions 45, S. 726–733
- [Seminerio 1998] Seminerio, M. (1998). *Study: One In Three Experiences Surfers Find Online Shopping Difficult*. <http://www.zdnet.com/intweek/quickpoll/981007/981007b.html> Letzter Aufruf: 01.10.1999
- [Shneiderman 1998] Shneiderman, B. (1998). *Designing the User Interface*. Addison Wesley Longman Inc., Reading
- [Snyder 1991] Snyder, K. M. (1991). *Reported characteristics of quality computer systems and usable computer systems*. In: Unpublished IBM Report. Reported in Snyder, K. M. A Guide to Software Usability. IBM, White Plains, New York.
- [Sun 1999a] Sun (1999) *What is Usability Engineering* <http://www.sun.com/usability/about.html> Letzter Aufruf: 25.11.1999
- [Sun 1999b] Sun (1999) *Benefits of Usability Engineering* <http://www.sun.com/usability/benefits.html> Letzter Aufruf: 25.11.1999
- [Sun 1999c] Sun (1999) *Cost Savings of Usability Engineering* <http://www.sun.com/usability/savings.html> Letzter Aufruf: 25.11.1999
- [Tognazzini 1992] Tognazzini, B. (1992). *Tog on Interface*. Addison-Wesley, Reading, S. 79–89
- [Viereck 1995] Viereck, A. (1995). *Kosten und Nutzen ergonomischer Anwendungssysteme. Ein Plädoyer für das Zusammenwirken von Betriebswirtschaft und Software-Ergonomie bei Software-Entwicklungsprojekten*. In: Daldrup, U. (Hrsg.). *Menschengerechte Softwaregestaltung. Konzepte und Werkzeuge auf dem Weg in die Praxis*. (Peter Gorny zum 60. Geburtstag). Berichte des German Chapter of the ACM Nr. 46, Teubner, Stuttgart, S. 181–215
- [Wixon und Jones 1991] Wixon, D. und S. Jones (1991). *Usability for fun and profit: A Case Study of the Design of DEC RALLY*. In: Rudisill, M., C. Lewis, P. G. Polson und T. D. McKay (Hrsg.) (1991). *Human-Computer Interface Design*. Morgan Kaufmann Publishers, San Francisco, S. 3–35

- [Wixon und Wilson 1997] Wixon, D. und C. Wilson (1997). *The Usability Engineering Framework for Product Design and Evaluation*. In: Helander, M. T., T. K. Landauer und P. Prasad V. Prabhu (Hrsg.)(1997). *Handbook of Human Computer Interaction*. Elsevier 1997 (2. Aufl.), S. 767–778