# Usability Engineering

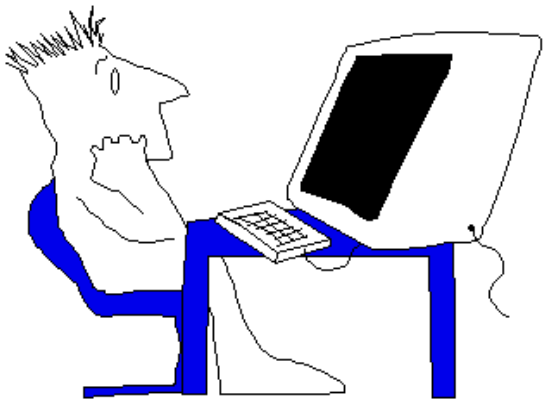
*Der Weg zu bedienungsfreundlicher Software*

Martina Manhartsberger

Langsam aber sicher wird der Term „Usability“, womit „Bedienungsfreundlichkeit“ oder einfach „Benutzbarkeit“ (Benutzerschnittstellen müssen nicht unbedingt freundlich, aber benutzbar sein) gemeint ist, auch hierzulande zu einem Begriff. Zwar ist die Forderung nach benutzbarer Software sogar schon im Arbeitnehmerschutzgesetz verankert, dennoch werden noch sehr wenige Erkenntnisse über Ergonomie in der Praxis auch umgesetzt. Am 7. August 1996 wurde nun auch in der BRD das „Gesetz zur Umsetzung der EU-Rahmenrichtlinie Arbeitsschutz und weiterer Arbeitsschutz-Richtlinien“ vom Bundespräsidenten unterzeichnet. Ein Grund sich zu überlegen, wie Software eigentlich bedienungsfreundlich gemacht wird..

## Motivation

Das enorme Wachstum des Personalcomputer- und Workstationmarktes in den 80iger Jahren hat dazu geführt, daß Bedienungsfreundlichkeit heute mit zu den Qualitätskriterien eines Produktes zählt. Die Erkenntnis, daß der Benutzer im Mittelpunkt einer Mensch-Maschine-Kommunikation stehen muß, setzt sich in der Softwarebranche langsam aber sicher durch. Es ist absehbar, daß Bedienungsfreundlichkeit in Zukunft das wichtigste Kriterium im Softwarewettbewerb sein wird. Schon jetzt verlangen immer mehr künftige Benutzer und Kunden nach graphikorientierten, bedienungsfreundlichen Programmen und über kurz oder lang werden sie bedienungsfreundliche Produkte schwer erlernbaren Produkten vorziehen, auch, wenn diese vielleicht bessere Performance zeigen, billiger sind, oder mehr Funktionalität enthalten. Die Kosten, die in Zeiten steigender Personalkosten durch hohe Einlernzeiten, Unzufriedenheit der Mitarbeiter etc. entstehen, stehen nicht im Vergleich zu den Einsparungen durch Kauf billiger Software oder Hardware.



## Moderne Benutzerschnittstellen

Benutzerschnittstellen durchliefen in den letzten Jahrzehnten eine rasante Entwicklung von der Eingabe über Lochkarten bis zu heute gängigen hochinteraktiven Programmen. Diese, für den Benutzer positive Entwicklung brachte steigende Komplexität im Bereich der Entwicklung von Benutzerschnittstellen.

In modernen Benutzerschnittstellen manipuliert der Benutzer graphische Objekte am Bildschirm mit der Maus. Er öffnet Fenster verschiedener Applikationen eines Fenstersystems, ordnet Objekte, die er für seine Arbeit benötigt nach Belieben am Bildschirm an, oder kann in manchen Anwendungen durch das Übereinanderschieben von Objekten Funktionen auslösen. Dabei werden asynchrone Eingabegeräte verwendet (momentan hauptsächlich Keyboard und Maus, Anwendungen virtueller Realitäten, die momentan noch in Entwicklung sind, sich aber bald für verschiedenste Bereiche durchsetzen werden, erfordern allerdings neue und weit komplexere Eingabegeräte), Echtzeit Feedback, Fenster, kunstvolle und echt wirkende Graphiken.

Während der Benutzer sich in älteren, meist maskenorientierten oder gar kommandoorientierten Softwarepaketen auf einem genau festgelegten Weg durch das Programm bewegt, indem er von einer Maske zur nächsten verzweigt, wird die Arbeitsweise des Benutzers moderner Benutzerschnittstellen weniger eingeschränkt. Der Benutzer bewegt sich nicht mehr auf einem vorgegebenen Weg durch das Programm, sondern er kann einen Weg auswählen, diesen unterbrechen, um einen anderen zu wählen und ihn später wieder fortsetzen. z.B. kann ein Textverarbeitungsprogramm geöffnet werden, über Menü eine Suchfunktion angewählt werden, wodurch eine Dialogbox geöffnet wird, um den Suchbegriff einzugeben. Hier unterbricht der Benutzer die Arbeit mit dem Textverarbeitungssystem und startet ein Spreadsheet. Dort gibt er eine Formel in ein Feld ein. Dann startet er zur Entspannung ein Spiel. Schließlich kehrt er zur Textverarbeitung zurück, um den benötigten Suchbegriff einzugeben. Diese Arbeitsweise ist für den Benutzer verständlich, er agiert ähnlich wie im realen Leben auch auf eine spontane, flexible Art und Weise.

## Entwicklung von Benutzerschnittstellen: es wird immer schwieriger

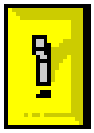
Die Erzeugung qualitativ hochwertiger, bedienungsfreundlicher Software ist für Entwickler eine komplexe, sowie zeitintensive und ressourcenintensive Aufgabe. Um dem Benutzer eine einfache Bedienung zu ermöglichen ist mehr und mehr Aufwand bei der Entwicklung notwendig. Der Anteil des Entwicklungsaufwandes der Benutzerschnittstelle am Entwicklungsaufwand für das Gesamtsystem steigt ständig. Untersuchungen haben gezeigt, daß im Durchschnitt 48% des Codes auf die Benutzerschnittstelle entfallen. Die durchschnittliche Zeitdauer, die für die Benutzerschnittstelle aufgewendet werden muß, beträgt in der Designphase 45%, in der Implementationsphase 50% und in der Wartungsphase 37%.

Während es für die Implementierung bereits verschiedene Ansätze gibt, sie durch spezielle Programmierumgebungen zu erleichtern, gibt es noch kaum Ansätze, den Designvorgang als Subteil der Entwicklung durch strukturierte Methoden zu unterstützen. Auch, welche Kriterien eine gute Benutzerschnittstelle ausmachen, ist noch nicht klar definiert. Existierende Richtlinien geben oft nur grobe Maßstäbe zur Einhaltung an, es gibt keine Richtliniensammlung, die die Erzeugung benutzerfreundlicher Software garantiert. Der Entwurf erfolgt meist nach Intuition, nicht durch Befolgung von Qualitätskriterien.

Ein weiterer Faktor, der die Entwicklung erschwert ist, daß die Details der Benutzerschnittstelle oft erst kurz vor der endgültigen Fertigstellung fixiert

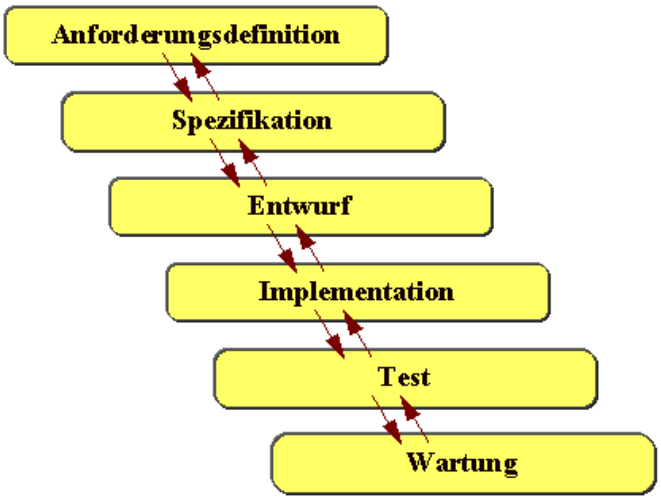
werden. Gründe dafür sind u.a., daß künftige Benutzer ihre Anforderungen meist erst angesichts eines laufenden Systems konkret definieren können.

Qualitativ gute Benutzerschnittstellen können nur dann erzeugt werden, wenn, ein gutes Design vorausgesetzt, es für alle Beteiligten die Möglichkeit gibt, das Design zu verstehen und zu evaluieren. Kommunikation über ein Design und Dokumentation des Designs kann durch Darstellung des Designs mit Designrepräsentationstechniken erfolgen. Die Motivation für geeignete Repräsentationstechniken ist besonders durch neuere Ansätze zum Entwicklungsprozeß gestiegen, die Evaluation und Rapid Prototyping beinhalten. Dadurch sind auch mehr und spezialisiertere Rollen für die Beteiligten am Entwicklungsprozeß entstanden. Solange eine Person für Design und Implementation eines Systems zuständig ist, können keine Kommunikationsprobleme entstehen. Je spezialisierter jedoch Rollen im Entwicklungsprozeß sind, desto größer wird der Bedarf an Kommunikation und geeigneten Kommunikationssprachen. Kommunikationsmängel führen zu Qualitätsmängeln der Software oder verzögern die Entwicklung, weil das Produkt nicht dem Design entspricht und neu überarbeitet werden muß.



Ideen zu einer Benutzerschnittstelle sind schwer zu erklären, solange sie nicht am Bildschirm sichtbar sind („Ein Bild sagt mehr als tausend Worte“). Der Entwurf einer Benutzerschnittstelle muß aber an die künftigen Benutzer, Manager und Entwickler weitervermittelt werden. Dies sollte so früh wie möglich geschehen, um spätere Änderungen zu vermeiden.

## Softwarelebenszyklus: Neue Modelle ersetzen den alten Wasserfall

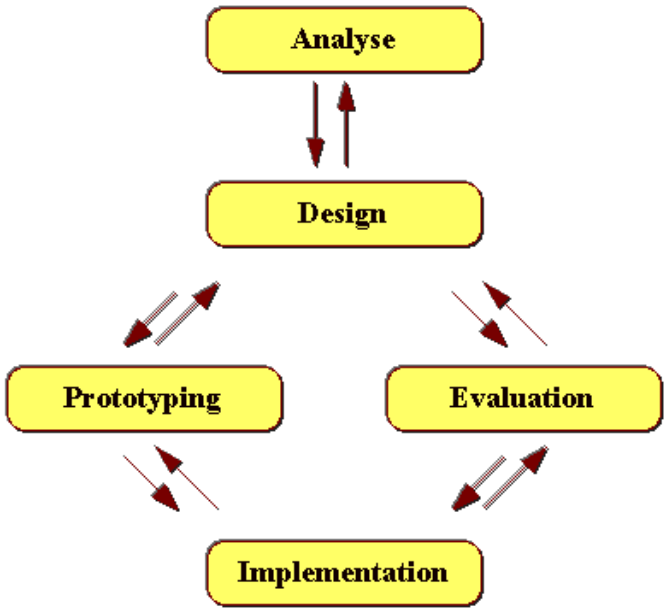



Der traditionelle Softwarelebenszyklus (das sogenannte „Wasserfallmodell“), der in Softwareentwicklungstheorie lange Zeit seinen angestammten Platz hatte, hält den Anforderungen, die eine benutzerzentrierte Entwicklung mit sich bringt, nicht mehr Stand. Bis auf die Tatsache, daß von den Phasen Anforderungsdefinition, Spezifikation, Entwurf, Implementation, Test und Wartung jede einen Weg zurück in die vorherige Phase hat, ist das Wasserfallmodell sequentiell. Bei der Anwendung dieser Methodologie für die Entwicklung von Benutzerschnittstellen hat sich bald gezeigt, daß diese streng sequentielle Vorgangsweise dafür nicht geeignet ist. Obwohl besonders für die Phase der Spezifikation formale Methoden verwendet werden, ist es schwierig in dieser Phase bereits alle Details eines Systems zu überblicken und zu berücksichtigen. Dies gilt im besonderen für den Benutzerschnittstellenbereich.

Aber noch weitere Gründe sprechen für die Notwendigkeit einer alternativen Entwicklungsmethodologie. Zum einen ist es im Benutzerschnittstellenbereich besonders schwierig, am Beginn der Entwicklung schon alle Details und Anforderungen zu erfassen. Weiters wurde beobachtet, daß Designer es bevorzugen, zunächst konkrete Beispiele durchzuspielen, also z.B. Bildschirmskizzen zu zeichnen, bevor sie den Grobentwurf des Systems, eventuell mit formalen Spezifikationsmethoden, angehen, d.h. sie arbeiten teils top-down, teils bottom-up.

Der wichtigste Faktor aber, der für alternative Entwicklungsmethoden spricht, ist der menschliche. Das Verhalten des Benutzers kann nicht vorhergesehen werden. Es ist daher kaum möglich, im ersten Durchgang bereits ein ideales Design zu finden. Oft haben auch Designer nicht genügend Einblick in den Arbeitsbereich des künftigen Benutzers.

Für die Entwicklung von Benutzerschnittstellen wurde daher ein Prozeß entwickelt, der den Benutzer und Benutzertests als wesentliches Element sieht. Die wichtigsten Schritte innerhalb dieses Prozesses sind die Benutzeranalyse, die Funktionsanalyse, das Design, Prototyping und Evaluation des Designs (des Prototyps), sowie schließlich die Implementation.




## Analysephase

Die Systemanalysephase erfolgt für Benutzerschnittstelle und Anwendungsfunktionalität gemeinsam. In dieser Phase werden im wesentlichen die Anforderungen an das System festgelegt. Die Aktivitäten innerhalb der Analysephase sind die Aufgabenanalyse, die Benutzerklassenanalyse, die Funktionanalyse, Produktvergleiche und die Festlegung von Usability Zielen.

### Aufgabenanalyse

Bei der Aufgabenanalyse werden jene Tätigkeiten identifiziert, die von Benutzern durchgeführt werden und die zum Aufgabenbereich des geplanten Systems gehören. Die Tätigkeiten werden analysiert, z.B. in Bezug auf die Frequenz, mit der sie durchgeführt werden (z.B. in einer Personalverwaltung: das Eintragen neuer Mitarbeiter wird nur selten durchgeführt, die Gehaltsabrechnung jedes Monat, Nachrichten an Mitarbeiter absenden dagegen muß jeden Tag durchgeführt werden), ihre Wichtigkeit für die Gesamtaufgabe und die Schwierigkeit, sie zu erlernen etc. und in Einzeltätigkeiten zerlegt. Zu diesem Zweck sollten Designer die künftigen Benutzer bei der Arbeit studieren. Nichts ist für die künftige Benutzerschnittstelle so wichtig wie ein Verständnis für die Benutzer und die Arbeitsvorgänge des Benutzers.

### Benutzerklassenanalyse

Bei der Benutzerklassenanalyse wird ein Modell vom künftigen oder potentiellen Benutzer entworfen. Es werden die Benutzerklassen und ihre Eigenschaften - ihr Verständnis für den Problembereich, ihr Wissen über Computer, allgemeine Eigenschaften (z.B. Rechts- oder Linkshänder) und Fähigkeiten (z.B. Beherrschen des Zehnfingersystems oder nicht) etc. analysiert. Wie detailliert dieses Bild vom künftigen Benutzer ist, hängt natürlich davon ab, auf welche Zielgruppe das Produkt ausgelegt ist, z.B. ob das Produkt innerhalb eines Unternehmens für eine bestimmte Tätigkeit verwendet werden soll oder für den freien Markt entwickelt wird als Produkt, für sehr viele (und damit sehr unterschiedliche) Benutzer.

## Funktionsanalyse

Bei der Funktionsanalyse wird die geplante Funktionalität des Produkts analysiert bzw. auch festgelegt. Liegt bereits ein Pflichtenheft vor, so dient dies als Grundlage. Idealerweise sollte allerdings die Aufgabenanalyse vor das Pflichtenheft gestellt werden, um mit der Funktionalität des Produkts jene Aufgaben abzudecken, die vom Benutzer auch tatsächlich durchgeführt bzw. benötigt werden. Für den Benutzerschnittstellendesigner ist es in dieser Phase wichtig, die gesamte geplante Funktionalität eines Produkts einzusehen und zu verstehen.

## Produktvergleiche

Die meisten Produkte die heute entwickelt werden sind nicht das erste Produkt in dieser Sparte und mit dieser Funktionalität. Meist gibt es Vorgängerprodukte, entweder aus dem eigenen Haus, weil ein Produkt für eine neue Plattform produziert werden soll, verbessert werden soll o.ä., oder ein Konkurrenzprodukt, das sich bereits auf dem Markt befindet. In diesem Fall sollte man unbedingt das Vorgängerprodukt studieren, Benutzertests mit dem Vorgängerprodukt durchführen und so Fehler, die man bei diesem Produkt findet (denn wenige sind fehlerlos) 8auf kostengünstige Art schon im vorhinein vermeiden. Konkurrenzprodukte können als Prototypen angesehen werden, aus denen man wertvolle Erkenntnisse für ein neues, besseres Produkt gewinnen kann.

**„Verdammte Hufe, wer entwirft eigentlich diese Cockpits? Waschbären?“**

## Usability Ziele

Ebenso, wie für die Funktionalität eines Produkts Ziele gesetzt werden, können und sollen auch für die Usability Ziele festgelegt werden. Dies ist natürlich nicht einfach, da die Benutzbarkeit eines Produkts schwer zu messen ist. Im Hinblick darauf, daß bei Benutzertests mehrere Personen die Software testen kann man allerdings Werte festlegen, die im Durchschnitt von den Testpersonen erreicht werden sollten. Dadurch werden die Werte objektiviert. Welche Zielwerte hier festgelegt werden hängt vom jeweiligen Produkt ab. Für eine Kalendersoftware z.B. könnte ein Zielwert sein, daß, vorausgesetzt das Programm läuft bereits, Benutzer im Durchschnitt nicht länger als sieben Sekunden benötigen sollten, um einen neuen Termin einzutragen. Die frühe Festlegung von Usability Goals und ihre Überprüfung bei jeder Design-Prototyp-Evaluations-Iteration ermöglicht es festzustellen, ob die Benutzeroberfläche gegen ein verbessertes, benutzerfreundlicheres Design konvergiert.

# Design

In der Designphase wird das Aussehen der Benutzerschnittstelle entworfen. Die Designphase beinhaltet das Design der gesamten Mensch-Computer-Interaktion, also die Definition von Benutzeraktionen, Feedback auf Benutzeraktionen, Festlegung des Aussehens des Bildschirms, der Tätigkeiten des Benutzers, Repräsentation der Systemfunktionalität, Reihenfolgenüberlegungen, Zugriffsmechanismen sowie Design von Benutzerschnittstellenobjekten, Interaktionsstilen und Interaktionstechniken.

## Konzeptuelles Design und Metaphernwahl

Das konzeptuelle Design findet auf einer höheren Ebene statt und definiert Objekte und Operationen auf den Objekten. Gleichzeitig werden ein oder mehrere Metaphern, das sind Repräsentationen von Systemfunktionen wie z.B. ein Papierkorb am Bildschirm, der die Funktion „Löschen“ repräsentiert, gewählt. Die Auswahl von Konzepten und Metaphern für die geplante Benutzerschnittstelle ist der Knackpunkt jedes Designs. Alle weiteren (detaillierten) Designentscheidungen müssen auf diesen grundlegenden Konzepten und Metaphern aufbauen. Es ist daher wesentlich, daß für diese grundlegenden Entscheidungen auf den Erkenntnissen der gesamten Analysephase - bes. betreffend die Systemfunktionalität und das Benutzermodell - aufgebaut wird, um ein konsistentes und durchgängiges Konzept zu erhalten. Nicht immer paßt eine einzige Metapher für ein ganzes Softwaresystem. Dann muß entweder eine alternative Metapher gesucht werden oder die bestehende Metapher erweitert und ergänzt werden. z.B. ist ein Papierkorb am Bildschirm eigentlich schon eine Erweiterung an und für sich vielen Fenstersystemen zugrundeliegenden Schreibtischmetapher, da sich Papierkörbe üblicherweise nicht auf einem Schreibtisch befinden. Hier ist eine sorgfältige Abwägung und Auswahl notwendig, um einen verständlichen und konsistenten Gesamteindruck zu bieten.

## Detailliertes Design

In der Phase des detaillierten Designs erfolgen, basierend auf dem konzeptuellen Modell und der gewählten Metapher, Aktivitäten wie z.B. die genaue Formulierung von Systemnachrichten an den Benutzer, Auswahl von Interaktionstechniken wie z.B. Menüs, Dialogboxen, Buttons, graphisches und verhaltensmäßiges Design von Bildschirmobjekten, Feedback auf Benutzeraktionen, wie z.B.: der Benutzer zieht ein Dokument zum Papierkorb, um es zu löschen: das Dokument verschwindet mit einem zischenden Geräusch oder der Papierkorb flackert auf oder der Papierkorb zeigt an einem Indikator die Füllmenge an oder eine Dialogbox erscheint mit einer Frage, ob das Dokument tatsächlich gelöscht werden soll oder .. etc. etc.

# Methoden für die Designphase

Für das Design selbst gibt es eigentlich keine Methoden. Ähnlich wie in anderen Bereichen ist Design etwas sehr Subjektives, hat allerdings bei der Entwicklung von Benutzerschnittstellen weniger mit künstlerischen Aspekten zu tun als mit psychologischen und ergonomischen. Wenn an dieser Stelle zwar kaum vorgefertigten Methoden für das Design selbst angeboten werden können, so doch Richtlinien und Standards. Richtlinien sind allgemein (und damit wiederum wenig pragmatisch) formulierte „Weisunge“ für das Design basierend auf Erkenntnissen aus der Psychologie. Als Standards werden momentan die einzelnen Fenstersystemstandards bezeichnet, die verschiedene Gestaltungsvorschriften enthalten, welche garantieren sollen, daß innerhalb eines Fenstersystems entwickelte Software möglichst konsistent ist. Nun wird sich so manche(r) fragen, wodurch sich also ein guter Designer auszeichnet, wenn es anscheinend kaum erlernbare Methoden gibt und Kunst damit auch nichts zu tun hat. Abgesehen von den Richtlinien und Standards, die Designer im Hinterkopf haben sollten, ist hier wohl die Erfahrung mit vielen Softwaresystemen, Benutzerschnittstellen und Designprojekten das größte Potential.

Als generelle Vorgangsweise innerhalb der Designphase sollen hier noch zwei Schlagworte, nämlich das des parallelen Designs und das des partizipativen Designs vorgestellt werden.

## Paralleles Design

Beim Parallelen Design arbeiten mehrere Designer gleichzeitig und unabhängig voneinander am selben Designprojekt. Dadurch beeinflussen sie einander nicht gegenseitig und es entstehen unabhängige Entwürfe. Aus den unterschiedlichen Alternativen kann sodann die beste gewählt werden oder es können verschiedene Ideen zu einer einzigen verschmolzen werden.

## Partizipatives Design

Beim, in letzter Zeit viel diskutierte Ansatz des Partizipativen Designs wird vorgeschlagen, Benutzer am Design teilhaben zu lassen. Es soll sich dabei um eine Zusammenarbeit handeln, bei der auch die Benutzer Designideen beitragen, alle Designentscheidungen mit Benutzern diskutiert werden und die Benutzer auch mit entscheidungsberechtigt sind. Dadurch soll erreicht werden, daß Korrekturen möglichst frühzeitig vorgenommen werden können bzw. durch die Mitarbeit des Benutzers gar nicht erst notwendig sind. Einzelne Benutzer sollten nicht zu lange am Design mitarbeiten, sondern ausgetauscht werden, da sie sich bald in der Rolle des Entwicklers sehen und schließlich nicht mehr repräsentativ für die typische Benutzergruppe sind.

## Prototyping

Prototyping wird in vielen industriellen Bereichen (Autoindustrie, Film, Architektur etc.) als Vorgangsweise bei der Entwicklung angewendet. Schon Alfred Hitchcock entdeckte das Prototyping - für die Filmindustrie: Zunächst testete er die Reaktionen auf Geschichten, die er bei Cocktailparties zum besten gab. Danach experimentierte er mit verschiedenen Abwandlungen der Story, die auf Kommentaren des Publikums basierten. *Psycho* war schließlich ein Ergebnis dieser Technik.

Beim Prototyping wird mindestens ein Prototyp des Endprodukts erstellt, von Benutzern und Designern getestet und, falls Verbesserungswünsche existieren, entsprechend geändert. Beim Rapid Prototyping (mittlerweile meist nur als Prototyping bezeichnet, da *langsames* Prototyping ohnehin wenig Sinn hat) wird davon ausgegangen, daß der Prototyp so schnell erstellt werden kann, daß genug Zeit bleibt, um auch grundlegende Veränderungen, die nach der Evaluation gewünscht werden, durchzuführen. Ist die Erstellung von Prototypen schnell genug, so werden mehrere Prototypingzyklen möglich, in denen der Prototyp iterativ verbessert und verfeinert wird.

Auch aus der Entwicklung interaktiver Systeme ist Prototyping mittlerweile als zentrales Element der meisten Entwicklungszyklen nicht mehr wegzudenken. Die Idee des Prototyping ist angesichts der schlechten Erfahrungen entstanden, die mit konventionellen Methoden gemacht wurden. Wenn keine Prototypen (in diesem Zusammenhang auch als Dialogsimulation oder ausführbare Spezifikation bezeichnet) erzeugt werden, so müssen sehr viele Designentscheidungen (komplexere Benutzerschnittstellen bestehen aus Tausenden von Details) getroffen werden, ohne die Möglichkeit, diese visuell darzustellen. Dies erschwert die Kommunikation innerhalb des Designteams, das aus Experten verschiedener Fachbereiche (Psychologie, Ergonomie, Informatik, Visuelles Design etc.) besteht, und auch die Kommunikation zwischen Kunden, künftigen Benutzern, Managern und Entwicklern. Benutzer und Manager können oft erst angesichts eines lauffähigen Systems die Anforderungen an ein solches System deutlich machen. Ist das Design schließlich implementiert, dann ist es zu spät, da zu aufwendig, um Änderungen vorzunehmen. Auch Entwickler und Designer haben oft Probleme mit konventionellen Entwicklungszyklen. Sie lassen sich nur ungern in die Abläufe pressen, die der konventionelle Softwarelebenszyklus ihnen vorschreibt. Statt dessen vermeiden sie gerne Spezifikationen und arbeiten lieber bottom-up als top-down.

Neben Kommunikations- und methodischen Problemen existiert auch nach wie vor das Designproblem an sich. Ebensowenig wie Methoden, die automatisch zum Ziel führen, gibt es konkrete Kriterien und Richtlinien für qualitativ gute Benutzerschnittstellen an denen sich der Designer orientieren könnte. In der Literatur sind zwar eine Vielzahl psychologisch fundierter Kriterien zu finden, diese sind jedoch nicht konkret genug und können nur als grobe Richtlinien verwendet werden. Schließlich ist es der „menschliche Faktor" des Endprodukts, der entscheidet und dieser ist nach wie vor schwer zu erfassen. Es ist daher im Moment noch nicht gezielt möglich, in einem Durchgang die beste Benutzerschnittstelle für ein Problem zu finden.

Für die spezielle Situation der Entwicklung von Benutzerschnittstellen, bei der Spezialisten aus verschiedenen Bereichen zusammenarbeiten, ist daher iteratives Prototyping die idealste Vorgangsweise. Prototyping hilft dabei, die Kommunikation zwischen den Anwendungsprogrammierern auf der einen Seite, die die Notwendigkeit von benutzerorientierter Entwicklung nicht immer verstehen, und den Designern der Benutzerschnittstelle, die ihrerseits wenig über implementationstechnische Anforderungen und Einschränkungen wissen, zu erleichtern.

Vordringlichster Zweck des Prototyping ist die Herstellung eines Systems, das so gut wie (mit dem verwendeten Werkzeug) möglich, die Benutzerschnittstelle simuliert, sodaß der gewählte Designansatz von Designern und Testpersonen evaluiert werden kann. Je ähnlicher der Prototyp der intendierten Benutzerschnittstelle ist, desto sinnvoller ist natürlich das Ergebnis der Evaluation.

Die Verwendbarkeit von Prototypen geht aber weit über den ursprünglichen Zweck der Evaluation hinaus. Prototyping ist eng mit der Spezifikation von Benutzerschnittstellen verbunden. Einerseits können Spezifikations- und Repräsentationsmethoden dazu verwendet werden, um zu spezifizieren, wie der Prototyp aussehen soll, besonders, wenn die Entwicklung des Prototyps nicht von den Designern, sondern von Entwicklern oder Spezialisten vorgenommen wird. Nicht jedes Prototypingtool ist einfach bedienbar, manchmal ist auch Programmieren erforderlich.

Andererseits kann der Prototyp selbst als Spezifikation angesehen werden. Wenn Designer mit Hilfe eines leicht bedienbaren Werkzeugs einen Prototypen erzeugen, dann stellt dieser die beste, weil lauffähige, Spezifikationsmöglichkeit dar, die es gibt.

Manche Theoretiker verlangen sogar, daß Designer die Prototypen selbst herstellen, da sie dadurch ein besseres Verständnis für Designrichtlinien erlangen und beim Prototyping neue Ideen bekommen. Manchmal wird ein Prototyp sogar nur zum Zwecke der Spezifikation erstellt. Funktionsfähige Prototypen sind das bestmögliche Kommunikationsmittel über die Benutzerschnittstelle. Im Vergleich zu jeder Sprache, die eine Benutzerschnittstelle beschreibt, schneidet der Prototyp besser ab, was Verständlichkeit betrifft. Er ermöglicht ein intuitives, schnelles Erfassen der Konzepte einer Benutzerschnittstelle. Es muß für die Erklärung der Darstellungen und deren Funktionalität nicht der Umweg über eine meist komplizierte Sprache gegangen werden, die vom Betrachter oder Leser erst erlernt werden muß. Prototypen können herkömmliche Spezifikationen, die dazu gedacht sind, Programmierern Information über die Benutzerschnittstelle zu vermitteln zumindest zum Teil (abhängig von ihrer Vollständigkeit) ersetzen. Für die Entwickler hat Prototyping den Vorteil, daß sie mit Hilfe der Prototypen besser abschätzen können, welchen Aufwand die Implementation der geforderten Funktionalitäten notwendig machen wird.

*"…not because we don't know enough yet, but because in a design domain we can never know enough."*

## Schwierigkeiten und Risiken

Neben den Vorteilen, die Prototyping aufweist, entstehen aber auch einige Schwierigkeiten bei der Entwicklung, die allerdings durch Anwendung großer Sorgfalt vermieden werden können. Zunächst muß sichergestellt sein, daß alle Beteiligten mit der Vorgangsweise des Prototyping einverstanden sind. Die Möglichkeit, einen Prototyp zu verbessern hängt schließlich vom Willen und Können der künftigen Benutzer ab, entsprechende konstruktive Kritik zu liefern. Systementwickler nehmen die Entwicklung von Prototypen z.T. nicht wirklich ernst und neigen zu Ungenauigkeiten mit dem Argument, daß sie kein „richtiges" Produkt produzieren. Manager sehen Prototyping z.T. als Verschwendung von Ressourcen. Bei großen Systemen kann auch der Prototyp sehr groß werden. Wird er dann noch immer als eine Art Spielzeug angesehen, kann dies zu Problemen beim Prototyping führen. Geeignete

Methoden und Werkzeuge können hier Abhilfe schaffen. Ernsthafte Probleme können auch auftreten, wenn die Beteiligten den Prototyp als Endprodukt ansehen. So schlagen manche Experten vor, möglichst viele Funktionen hinzuzufügen, auch, wenn diese nicht für das Endprodukt geplant waren, um während der Prototypingzyklen auch die Anforderungen nochmals zu testen und eventuell zu revidieren. Andere dagegen warnen davor, da dadurch von Benutzerseite zu hohe Erwartungen erzeugt werden, die möglicherweise angesichts des Endprodukts enttäuscht werden müssen.

Die Begeisterung von Benutzern und Entwicklern an weiterem Prototyping kann schwinden, wenn sie einen akzeptablen, lauffähigen Prototypen sehen. Aber auch das Gegenteil kann - besonders beim revolutionären Prototyping ein Problem - eintreten. Entwickler oder Designer möchten einen Prototypen immer weiter bis zur Perfektion verbessern. Um den Prototypingprozeß nicht endlos werden zu lassen, muß hier als Ziel gesetzt werden es „gut genug“ zu machen, eine heikle Managementaufgabe.

## Evaluation

Die Evaluation ist die zentrale Aktivität bei der Entwicklung eines interaktiven Systems und innerhalb des Prototyping-Zyklus. Ohne Evaluation ist es unmöglich zu wissen, ob das System die Anforderungen und die, in der Analysephase festgelegten Usability Ziele erfüllt. Evaluation bedeutet das Sammeln von Daten über Verwendbarkeit eines Designs oder eines Produktes bei einer bestimmten Gruppe von Benutzern für eine bestimmte Aktivität innerhalb einer bestimmten Umgebung oder eines Arbeitskontextes.

Das Design Team wird durch die aus der Evaluation gewonnenen Daten während der gesamten Entwicklung darüber informiert, wie gut das vorgeschlagene Design auf die Bedürfnisse der Benutzer, auf die Tätigkeiten, die ausgeführt werden müssen und auf die Umgebung abgestimmt ist.

Verschiedene Formen von Evaluations-Feedback sind in den verschiedenen Stufen der Systementwicklung notwendig. In den früheren Aktivitäten genügt es, Ideen zu testen und auszuprobieren. Somit sind hier eher informale Evaluationsmethoden ausreichend. Zu späteren Zeitpunkten der Systementwicklung (vor allem, wenn bereits ausführbare Prototypen vorliegen) können aufwendigere, formaler orientierte Benutzertests angewendet werden.

Als grundlegendes Prinzip gilt das ständige Sammeln von Evaluationsdaten. Es ist sowohl immer möglich, als auch immer notwendig, eine bestimmte Form der Evaluation durchzuführen. Die verwendeten Evaluationsmethoden müssen je nach der entsprechenden Situation ausgewählt und angepaßt werden.

Als generelle Regel gilt, daß jede Art des Benutzertests besser ist als der Verzicht auf Benutzertests und daß die Tests so früh wie möglich durchgeführt werden sollten, damit die gewonnenen Erkenntnisse auch tatsächlich umgesetzt werden können. Auch in kleinerem Rahmen und mit weniger Testpersonen durchgeführte - und damit kostengünstigere - Tests können oft schon die gröbsten Usability-Mängel beseitigen.

## Neue Rollen

Da sich die Entwicklung von Benutzerschnittstellen vom traditionellen Wasserfallentwicklungsmodell unterscheidet, entstanden auch neue Rollen für die Beteiligten im Entwicklungsprozeß. Das entstandene Spezialgebiet bringt neue Arbeitsplatzbeschreibungen für Spezialisten hervor. Solange Benutzerschnittstelle und Anwendungssystem nicht als zwei eigenständige Komponenten eines Softwaresystems anerkannt waren gab es auch keine Notwendigkeit einer Trennung der Aufgabengebiete. Die Programmierer des Systems waren auch verantwortlich für Entwurf und Implementation der Benutzerschnittstelle. In der Praxis ist das auch heute noch teilweise der Fall, da sich der Bereich der Benutzerschnittstelle als eigenständiges Gebiet noch nicht überall durchgesetzt hat. Gerade diese beiden Aufgabengebiete sollten aber unbedingt getrennten Rollen zugeschrieben werden. Eines der größten Probleme beim User Interface Design ist nämlich jenes, daß es für Experten unmöglich ist, sich in den Zustand eines Anfängers zurückzuversetzen. Je mehr man über ein System bereits weiß, desto schwieriger ist es somit, einen neuen Benutzer zu verstehen. Besteht - für größere Projekte oder in größeren Unternehmen - die Möglichkeit, die Tätigkeiten noch weiter aufzuteilen, dann ergeben sich die Positionen des Designers, des Erzeugers von Prototypen, Testplaner, Techniker für die Unterstützung bei Tests, etc.

## Auswirkungen auf die Organisation

Die Änderung der Schwerpunkte des Softwarelebenszyklus bringt natürlich auch für jene Organisationen Veränderungen mit sich, die Software herstellen, aber auch jedes andere Produkt mit Benutzerschnittstelle, wie Multimediapräsentationen, WWW (Internet), Kioske und Selbstbedienungsautomaten - oder möchten Sie ein Handbuch lesen müssen, bevor Sie einen Bankomaten bedienen können? (apropos: können Sie einen Fahrscheinautomaten bedienen? Oder können Sie sich vorstellen, daß ein Besucher in Österreich das schafft? ;-) ) -, Videorekorder, Waschmaschinen etc. etc.

Durch die Änderung von Abläufen ändern sich Rollenbeschreibungen, Abteilungen und ganze Organisationen. Daß dies mit ein Grund ist, daß Usability nur langsam ihren Einzug in alltägliche Überlegungen nimmt scheint offensichtlich und es wäre auch völlig falsch, Organisationen in großem Maßstab rasch zu verändern. Langsam aber sicher sollte allerdings der Gedanke an die Usability und damit an den Endbenutzer einen zentraleren Stellenwert einnehmen, Gesetze hin oder her.

## Beim Landevorgang

Pentium1 *Hey, sind wir nicht schon zu tief??*
Pentium2 *Ach wo, nach meinen Berechnungen sind wir noch zu hoch!*
Pentium3 *Wo sind wir eigentlich??*
Pentium4 *Zu hoch stimmt, wir können also noch tiefer.*

Pentium1 *Und ich sage ZU tief!!*
Pentium2 *Zu hoch, zu hoch, zu hoch!!!*
Pentium3 *Ich spiel' jetzt 'ne Runde Pac Man ...*
Pentium4 *Also runter mit der Kiste.*

Pentium1 *Laut Radar streifen wir schon fast die Bäume.*
Pentium2 *Glaubst Du nun dem Radar, oder unseren Berechnungen.*
Pentium3 *X-Wing wäre mir ja lieber ...*
Pentium4 *Also 3 ... eh ... 2 gegen 1. Wir gehen tiefer!*

Pentium1 *Das sage ich alles Billy Boy!!*
Pentium2 *Hey, der Pilot will die Kiste hochziehen, der Idiot!*
Pentium3 *Wenn ich groß bin, will ich 'mal ein Motorola werden!*
Pentium4 *Immer diese unzuverlässigen Menschen! Manuelle Steuerung aus!*

*Wir übernehmen jetzt!!*

Pentium1 *- Schmoll -*
Pentium2 *Hey, was scheppert denn da so laut??*
Pentium3 *Ich bin ein rosa Hochzeitskuchen tüdeldideldumm ...*
Pentium4 *Seit wann gibt es 500 Meter hohe Bäume? So ein Mi....*

KAWUMMMMMMMMMMMMMMMMM!!!

Später in den Nachrichten :

Am Abend stürzte ein Airbus beim Landevorgang ab. Nach der Ursache wird noch geforscht. Eine Auswertung der BlackBox gab bisher keine gezielten Hinweise, jedoch läßt der häufiger vorkommende Satz „Rosa Hochzeitskuchen“ darauf schließen .....

## Und noch einer

Bisher hieß es immer: Computer machen keine Fehler. Dank modernster Hardware wurde jetzt auch dieses Manko beseitigt ...