Didaktische Grundfragen der Informatik
Softwareergonomie

Marco Kuhrmann*
Universität Potsdam – WS02/03

14. Februar 2003

Zusammenfassung

Softwareergonomie befasst sich mit der Anpassung von interaktiven Computersystemen, insbesondere von interaktiven Softwaresystemen, an die Bedürfnisse von Benutzern bzgl. Nutzbarkeit, Anwendungsfreundlichkeit und Effizienz. Ausgehend vom aktuellen Stand der Technik treten an dieser Stelle dialogorientierte Anwendungssysteme in den Vordergrund. Ein Dialog, speziell eine Dialogmaske, stellt die Schnittstelle zwischen dem Menschen und dem Computer her. Sämtliche Aktionen sowie Interaktionen und Kommunkationsakte werden mithilfe dieser Schnittstelle durchgeführt.
Im Folgenden wird ein Überblick über das Gebiet der Softwareergonomie gegeben. Neben allgemeinen Gedanken und Prinzipien werden Normen und Richtlinien betrachtet. Die Ausführungen werden durch einen Exkurs in das Teilgebiet *User Interface Design* abgeschlossen.

1 Einleitung

Im vorliegenden Papier ist eine ausführliche Darstellung der Inhalte des begleitenden Vortrags zu finden. Diese soll nicht das gesprochene Wort ersetzen und auch nicht die gleiche Orientierung verfolgen, wie der Vortrag. Die angesprochenen Themen werden hier kompakt und in sich abgeschlossen dargestellt. Dabei wird versucht ein Höchstmaß an Objektivität und Sachlichkeit zu wahren.

Folgende Themengebiete werden im Rahmen dieser Arbeit betrachtet. Zunächst werden die Fragen „Was ist Softwareergonomie?" und „Warum Softwareergonomie?" aufgegriffen. Dabei werden Themen, wie Mensch-Maschine-Kommunikation, Normen und Richtlinien betrachtet und die Begriffe Information und Medium im hiesigen Kontext dargestellt. Die Arbeit wird durch einen kurzen Exkurs in das Gebiet des „User Interface Design" abgeschlossen. Dabei wird die Übertragung von Normen auf die Aspekte der UI-Gestaltung durchgeführt. Weiterhin werden Anregungen zum Design grafischer Benutzungsschnittstellen gegeben, sowie Empfehlungen bzgl. positiver und negativer Aspekte geäußert.

*mkuhr@cs.uni-potsdam.de

2 WAS IST SOFTWAREERGONOMIE?

Beispiel 2.1 *Ein unerfahrener Benutzer eines Softwaresystems hat ein Problem, welches er nicht selbstständig lösen kann (vgl. Abbildung 1). Für solche Fälle gibt es in der Regel eine Hotline, die Benutzern weiterhilft. Allerdings ist der hier betrachtete Nutzer so unerfahren, dass er mit den Anweisungen der Hotline nicht zurechtkommt. In diesem*

Abbildung 1: Szenarium (Quelle: `www.heise.de/ct`)

Fall liegt ein Kommunikationsproblem vor, da der Benutzer nicht das benötigte Wissen hat.
Moderne Softwaresysteme sollen ihre Anwender bzgl. der Benutzbarkeit des Systems unterstützen und solche Situationen vermeiden helfen.

Softwareergonomie bezeichnet ein interdisziplinäres Forschungs- und Arbeitsgebiet, welches sich mit Fragestellungen der Interaktion von Mensch und Maschine in

interaktiven Softwaresystemen und der Gestaltung von Schnittstellen zur Interaktion auseinandersetzt.

Softwareergonomie hat die Aufgabe, Benutzungsschnittstellen für die *Mensch-Maschine-Kommunikation* zu beschreiben und zu bewerten. Die Bewertungskriterien formulieren hierbei die Ziele:

- Einfach zu erlernende, intuitiv einsetzbare, fehlertolerante Computersysteme.
- Steigerung der Produktivität und Gebrauchstauglichkeit.
- Dadurch Steigerung der Akzeptanz von interaktiven Systemen in der Gesellschaft.

Diese Themen werden besonders durch die *Human Computer Interaction* (HCI, siehe [HCI02]) bearbeitet. Interaktive Softwaresysteme werden durch die HCI als soziotechnische Systeme verstanden. Auch der Terminus kontingentes System (vgl. [Döl02]) ist hierfür gebräuchlich. Weiterhin werden Empfehlungen für die Gestaltung und Realisierung von Benutzungsschnittstellen gegeben. Diese werden auch Bezug nehmend auf Benutzer, -aufgaben und Anwendungsbereiche bewertet. Ziel ist es zu erreichen, dass Softwaresysteme zur Interaktion geeignet sind, womit der Mensch selbst als Systemkomponente (siehe Abbildung 2) zu betrachten ist. Die HCI fasst nicht nur Teildisziplinen der Informatik zusammen, sondern ist ebenfalls Tätigkeitsbereich für die Psychologie, Linguistik, Betriebswirtschaft und Weitere. Softwareergonomie ist somit nicht klar abgrenzbar.

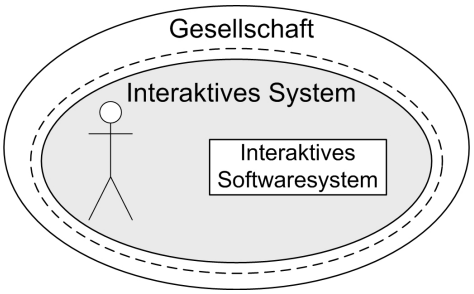

Abbildung 2: Der Mensch als Systemkomponente

3 WARUM SOFTWARERGONOMIE?

3.1 MENSCH-MASCHINE-KOMMUNIKATION

Mit dem Einführen des Menschen in die Betrachtung und dem Verstehen des Menschen als Systemkomponente ergeben sich viele Schwierigkeiten.

Es ist nun im Kontext von relativ gut vorhersagbaren Softwaresystemen eine erhebliche Unbekannte zu beachten. So sind nun Fragen der Automatisierung und Kommunikation auf völlig anderen Ebenen zu diskutieren. Ausgangspunkt hierfür ist die Feststellung, dass in interaktiven Softwaresystemen eine Mensch-Maschine-Kommunikation (Abbildung 3) stattfindet. Diese Kommunikation erfolgt für gewöhnlich durch eine Benutzungsschnittstelle, wie z. B. der Bildschirmarbeitsplatz. Dieser ist für den Menschen brauchbar zu gestalten, womit es hier schon zu ersten Komplikationen kommen kann. Hierfür seien z. B. Berührungsschwierigkeiten oder Akzeptanzprobleme genannt. Diese sind nicht generalisierbar und hängen von unterschiedlichen internen (menschbezogenen) und externen Faktoren ab.

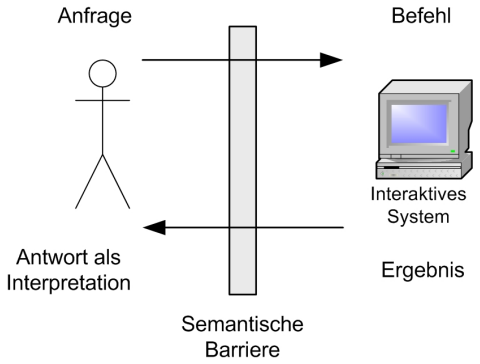

Abbildung 3: Mensch-Maschine-Kommunikation

Beispiel 3.1 (Akzeptanz von Softwaresystemen)
Softwaresysteme werden nicht akzeptiert, wenn ihre Qualität schlecht ist, d. h. dass Aufgaben mit einem solchen System nur kompliziert und unbefriedigend erledigt werden können.
Auch Rationalisierungsängste wirken sich negativ auf das Akzeptanzverhalten von Benutzern aus. Sie sehen in Softwaresystemen meist mehr Konkurrenz als Unterstützung.

Beispiel 3.2 (Erterne Faktoren) *Die Qualität eines Softwaresystems kann noch so gut sein, wenn es in einer Umgebung eingesetzt wird, in der ein nicht zumutbarer Lärmpegel herrscht oder die Helligkeitsverhältnisse nicht akzeptabel sind. In Folge dessen wird es nur zögernd eingesetzt.*

Auch diese Faktoren hängen wieder im Wesentlichen vom persönlichen Empfinden des Benutzers ab. Offensichtlich kann man aufgrund dieser Feststellungen die Aussage treffen, dass es keine optimale Schnittstelle zwischen Mensch und Maschine gibt.

3.2 INFORMATION UND MEDIEN

Der Austausch von Informationen stellt den Kern der Mensch-Maschine-Kommunikation dar. Jedoch soll der

Begriff Information hier nicht diskutiert werden, da er selbst nicht definiert ist. Vielmehr wird betrachtet, wie Informationen ausgetauscht werden. Der Austausch findet immer in einer interpretierbaren Form statt. Es genügt daher, sich auf den Begriff Daten und ihre strukturierte Repräsentation zu konzentrieren. Beide Begriffe werden im Folgenden der Einfachheit halber synonym verwendet.

Im Informationsaustausch finden sich neben der Ungeklärtheit des Informationsbegriffs noch eine Reihe weiterer Probleme. Zunächst ist das Problem der Interpretation von Informationen zu nennen. Auf beiden Seiten (Mensch und Maschine) wird Information interpretiert. Dabei unterscheiden sich mentales und elektronisches Verarbeitungsmodell erheblich. Dies betrifft sowohl die Repräsentation der Daten sowie deren Erfassung und Verarbeitung. Verwiesen sei an dieser Stelle noch einmal auf Abbildung 3, welche diesen Umstand durch eine „semantische Barriere" symbolisiert. Das Überwinden dieser Barriere ist nur durch beiderseitige Interpretation der Informationen möglich.

Der Unterschied wird deutlich, wenn man betrachtet, wie ein Computer Daten erfasst, interpretiert und verarbeitet. Dies geschieht immer in binärer Form.

Menschen hingegen verwenden 5 Sinne zur Wahrnehmung. Dies sind Sehen, Hören, Riechen, Schmecken und Tasten. Informationen werden zunächst durch externe Reize (Umweltreize) erfasst. Sie werden durch komplexe elektrochemische Prozesse im Nervensystem verarbeitet. Dies wird durch ein s.g. Phasenmodell dargestellt, welches aus Interpretation, Speicherung, Externalisierung und Anwendung besteht. Dieser Vorgang beschreibt im Wesentlichen das Lernen und Interagieren eines Menschen mit seiner Umwelt (vgl. Abbildung 4).
Jedoch ist das so erworbene Wissen nicht perfekt. Informationen und Daten werden i. A. durch Modellbildung strukturiert, organisiert und abgelegt. Hierbei kommen unterschiedliche Bereiche des Gedächtnisses zum Tragen. Je nach Speicherort und Zustand des Wissens (aktiv/inaktiv) erhält man unterschiedliche Qualität bei der Reproduktion des Erlernten. Die Reproduktion von Wissen ist immer fehlerbehaftet. Sie wird durch unterschiedliche Zugangspfade angestoßen. Meistens werden Informationen nicht wiedergegeben, sondern interpoliert. Auch Ähnlichkeiten führen zu Assoziationen und zur Übertragung bekannten Wissens auf neue Sachverhalte[1]. Hinzu kommt, dass die Gedächtnisleistung des Menschen nicht immer konstant ist. Alle diese Faktoren lassen sich nicht generalisieren, sondern sind individuell.

Aus diesem Grund ist der Mensch als Komponente des interaktiven Systems Partizipant unterschiedlichen Gewichts. Obwohl die Anforderungen und Erwartungen, die Computersysteme an ihre Nutzer stellen konstant sind, stellt die Interaktion selbst ein Problem dar. Dieses Problem ist im Individuum des Nutzers zu suchen.

Jeder Benutzer stellt an das System unterschiedliche Anforderungen und tritt diesem mit unterschiedlichen Erwartungen gegenüber. Aufgrund dieser Situation ist es ratsam Benutzer zu kategorisieren. Dies kann anhand unterschiedlicher Kriterien erfolgen.

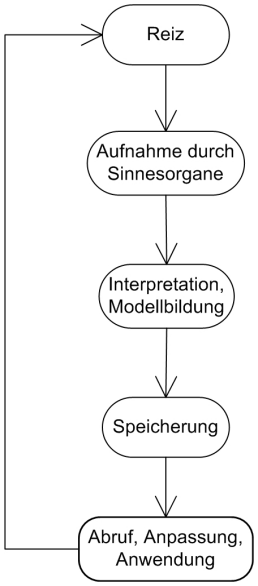

Abbildung 4: Menschliche Informationsverarbeitung

- Verhalten des Benutzers,
- Fertigkeiten des Benutzers,
- Neigungen des Benutzers,
- Wissen des Benutzers,
- Lebensalter,
- Tätigkeitsfelder, Rollen in Unternehmen und Weitere.

Die Festlegung von Benutzergruppen hat wesentliche Auswirkungen auf die Akzeptanz gegenüber Softwaresystemen. Folgende Szenarien verdeutlichen dies.

Beispiel 3.3 (Verhalten des Benutzers) *Softwaresysteme sollten auf Menschen mit cholerischen oder spastischen Neigungen eher beruhigend wirken. Dazu muss ein Softwaresystem derart konstruiert sein, dass Konfliktpunkte nach Möglichkeit gemieden werden. Dies erfordert ein äußert fehlertolerantes System.*

Beispiel 3.4 (Fertigkeiten des Benutzers) *Ein Softwaresystem muss berücksichtigen, dass nicht alle potentiellen Benutzer über die gleichen Fähigkeiten und Fertigkeiten verfügen. Dies können z. B. geistige oder körperliche Behinderungen sein, die Ursache dafür sind, dass die Software nicht in vorhergesehener Weise genutzt werden kann. Ein Beispiel hierfür wäre ein Bildschirmarbeitsplatz für einen Blinden.*

[1] Von diesen Vorgängen machen z. B. Metaphern Gebrauch.

Beispiel 3.5 (Tätigkeitsfelder, Rollen in Unternehmen) *Ein Softwaresystem muss an verschiedene Benutzerrollen anpassbar sein. Ein Beispiel hierfür ist Software aus dem Unternehmenseinsatz. Administratoren warten die Software und die Angestellten mit ihr. Dies ist eine strikte Aufgabentrennung innerhalb der Unternehmensstruktur, die durch ein Softwaresystem nicht ignoriert werden darf. Software ist nur dann erfolgreich, wenn sie an Strukturen und Prozesse von Unternehmen anpassbar ist. Software mit entgegengesetzer Zielstellung (das Unternehmen passt sich der Software an) ist meistens nicht akzeptiert worden und gescheitert.*

Das ignorieren der menschlichen Komponente bei der Konstruktion von interaktiven Softwaresystemen kann gravierende Folgen haben. Dies können sowohl psychologische, gesundheitliche oder betriebswirtschaftliche Folgen sein. Softwaresysteme, die unter Auslassung des Menschen konzipiert wurden sind meist „Selbstzweck". Sie werden nicht oder nur widerwillig genutzt, was die Produktivität und Effizienz senkt. Somit hätten Softwaresysteme ihren eigentlichen Sinn verfehlt. Die Auswirkungen auf den Benutzer sind ebenso gravierend. Ist die Software nicht klar in ihrer Handhabung, zu kompliziert oder fehlerträchtig, wird sie entweder nicht verwendet oder sie wird falsch verwendet. Hierbei ist der letzte Punkt offensichtlich der schwerwiegendere, da der Benutzer mit der Zeit kein Maß mehr für die Qualität der Software hat. Mit zunehmender Leidensfähigkeit der Benutzer sinkt die Qualität der Software, da sie dem Benutzer letztendlich egal ist.

3.3 Strukturierung und Repräsentation von Information

Wie bereits weiter oben festgestellt wurde verläuft der Informationsaustausch zwischen Mensch und Maschine über Daten in verschiedenen Repräsentationsformen. Die Repräsentationsformen lassen sich grob wie folgt Kategorisieren:

- Text (alphanumerische Darstellung)
- Bilder (Fotos, Malereien)
- Grafiken (Diagramme, Skizzen)
- Audioinformationen (Musik, Sprache)
- Film (Videos)
- Gestiken (haptische Informationen, wie Mimiken)

Diese Repräsentationsklassen besitzen i. A. eine Struktur, die es erleichtert, die syntaktische Repräsentation mit Semantik zu versehen. Die Vorgehensweise der Strukturierung kommt dem menschlichen Denken (Bildung mentaler Modelle) sehr entgegen. Gebräuchliche Strukturen, die sich auf o. g. Repräsentationsformen anwenden lassen sind:

- Hierarchische Strukturen (Bäume)
- Sequentielle Strukturen (Listen)
- Tabellarische Strukturen (Matrizen)
- Graphenstrukturen
- Karten (geodätische, geographische, thematische Karten)
- Hypertextsysteme

Das Hypertextsystem ist hierbei die komplexeste Struktur, da sie mehrdimensionale Graphen erzeugt und somit mehrdimensionale Informationsräume (*Hyperspace*) aufspannt (Abbildung 5). Die verknüpften Informationen können thematisch verschieden sein und auf unterschiedlichen Repräsentationsformen basieren. Hierfür sei als Beispiel ein Web Portal genannt, welches über Tabellen, Texte und Grafiken kombiniert und strukturiert und durch Mechanismen des Hypertextsystems auf Audio und Videoinformationen verweist.

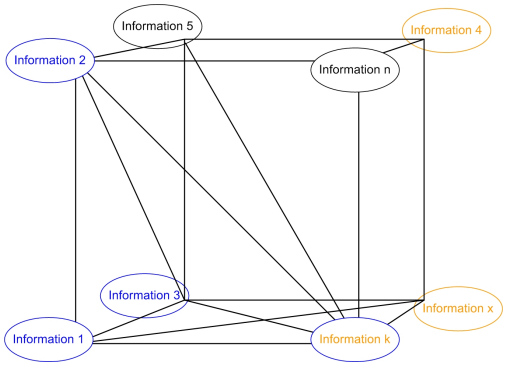

Abbildung 5: Hypertextsysteme als mehrdimensionale Informationsstruktur

3.4 Der Medienbegriff

Information ist nicht korpuskular, so dass sie stets ein Trägermedium benötigt. Dieses Trägermedium enthält Informationen und macht diese zugänglich. Das Medium bildet, im Sinne der Systemtheorie, einen Kanal zwischen einer Quelle und einem Empfänger. Medien können nach bestimmten Schemata klassifiziert werden. Die gebräuchlichste Klassifikation ist eine Einteilung in

- Klassische Medien und
- Neue Medien,

wobei man zusätzlich noch Primärmedien, Sekundärmedien etc. unterscheidet. Unter klassischen Medien sind z. B. das Buch, das Radio oder auch die Zeitung einzuordnen. Zu den neuen Medien zählen z. B. Softwaresysteme

oder Internet basierte Dienste. Diesen Zuordnungen ist zu entnehmen, dass Medien selbst eine Evolution durchlaufen können. Als Beispiel hierfür sei das Buch genannt, welches in seiner ursprünglichen Form den klassischen Medien zuzuordnen ist. In Form des E-Books ist es nun den neuen Medien zuzuordnen. Weiterhin ist zu bemerken, dass Medien anscheinend nur eine beschränkte Lebensdauer haben. In der Literatur findet man häufig den Begriff der Verdrängung.

Beispiel 3.6 (Verdrängung) *Heutzutage werden keine Höhlenmalereien mehr angefertigt. Die Audiokassette ist zum größten Teil von der CD verdrängt worden.*

3.5 Konsequenzen für das Systemdesign

Aus den obigen Ausführen lassen sich direkt Konsequenzen für ein ergonomisches Systemdesign ableiten.

- Der Mensch als Systemkomponente muss berücksichtigt werden. Die schließt eine Benutzerkategorisierung ein.
- Das interaktive System muss den Ansprüchen des Benutzers entsprechend konzipiert sein. Dies betrifft die Form der Darstellung von Information und die Form der Eingabe von Informationen über entsprechende Eingabemedien, wie z. B. eine Tastatur.
- Die Kontrollfunktion des Menschen muss gewahrt werden, obwohl das Softwaresystem mächtiger sein kann.
- Das Softwaresystem muss dem Benutzer ein angemessenes Maß an Hilfestellung bieten, sodass er seine Aufgaben damit effektiv lösen kann.
- Die Benutzerführung durch ein Dialogsystem sollte benutzergerecht sein und sollte in ihrer Gestaltung anpassbar sein.

Die Um- und Durchsetzung dieser Konsequenzen soll durch Normen und Richtlinien gewährleistet werden. Drei konkrete Normen sind hier z.B. die ISO 9241 (vgl. [ISO01a], [ISO01b]), die DIN 66234 und die Bildschirmarbeitsplatzverordnung (BildschArbV). Die Inhalte dieser Normen werden im Rahmen der Betrachtungen zum User Interface Design aufgeführt.

4 Schwerpunkt - User Interface Design

Im Folgenden wird der Schwerpunkt auf den Entwurf und die Gestaltung ergonomischer Benutzeroberflächen gelegt. Hierbei werden noch einmal rückblickend die Aspekte der Mensch-Maschine-Kommunikation aufgegriffen, die Umsetzung der Normen für ergonomische Benutzungsschnittstellen gezeigt und einige Do's und Don'ts zum User Interface Design aufgeführt.

Beispiel 4.1 (Ein schlechtes Userinterface) *Das hier gezeigte Userinterface (Abbildung 6) zeigt einige gravierende Mängel auf. Diese sollten auch dem ungeübten Leser sofort ins Auge fallen. Klare Verstöße gegen sämtliche Regeln der Farbenlehre und Typografie kennzeichnen diese Webseite. So ist zunächst das gewählte Farbschema (Blau, Rot, Gelb) zu bemängeln. Es ist klar ersichtlich, dass gelber Text auf weißem Hintergrund kaum zu erkennen und somit defacto nicht lesbar ist. Des Weiteren ist ein massiver optischer Bruch zwischen blauen Gruppierungselementen und rotem Text zu bemerken. Weiterhin fällt auf, dass diese Seite keine klar erkennbare Struktur besitzt, obwohl sie kaum Inhalt enthält.*
Ein solches UI-Design sollte vermieden werden.

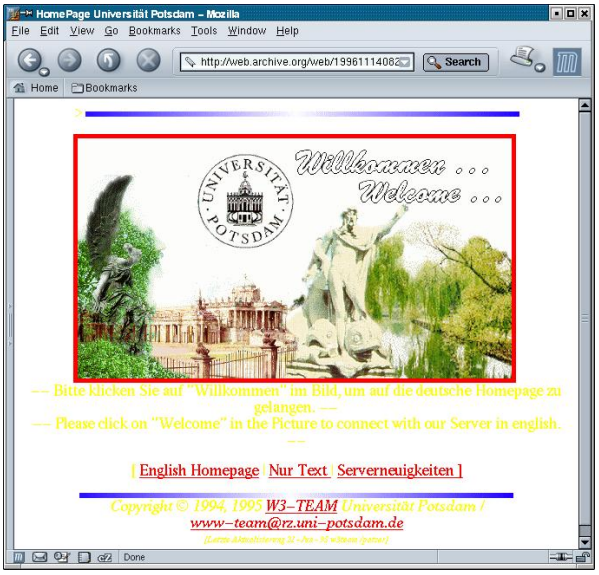

Abbildung 6: Webseite der Unversiät Potsdam von 1996

Als zentraler Punkt des Folgenden Abschnitts ist die Frage *„Was ist eine ergonomische Benutzerschnittstelle?"* zu sehen. Wie bereits weiter oben festgestellt wurde liegt das eigentliche Problem bei der Festlegung von Ergonomierkriterien bei der Systemkomponente Mensch. Ergonomisches Empfinden ist ein subjektiver Eindruck eines einzelnen Individuums. Dieses Empfinden ist abhängig von Aspekten, wie Alter, Vorlieben etc.

Es existieren Ansätze, dieses Problem durch Formalisierung beherrschbar zu machen. Hierbei zielen unterschiedliche Ansätze auf unterschiedliche Ebenen der Interaktion. Einige Ansätze verfolgen durch *verhaltensorientierte* Modellierung (Modellierung aus Benutzersicht) die Definition von Benutzungsschnittstellen in komplexen Regelsyste-

men (*GOMS*[2]). Andere verfolgen das Ziel, Verhalten mit Mitteln der theoretischen Informatik zu erfassen, wie z.B. die *TAG*[3] oder *UAN*[4]. Die verhaltensorientierte Modellierung zielt hauptsächlich auf den Benutzer und dessen Verhalten ab. Es wird versucht die semantische und pragmatische Ebene der Interaktion zu erfassen. Von daher wird sie für reale Systeme sehr schnell hochgradig komplex und somit praktisch nicht anwendbar, da sich menschliche Individuen nicht formalisieren lassen.
Auf der anderen Seite existieren Ansätze, die eine Konstruktion aus Systemsicht vornehmen (syntaktische und lexikalische Ebene der Interaktion). Die *konstruktionsorientierten* Ansätze stellen den Anwender zunächst in seiner Priorität zurück. Dies ist von Vorteil für das Softwaresystem, da es beherrschbar bleibt, unter Umständen wird jedoch ein System erzeugt, das praktisch nicht anwendbar ist, da der Mensch ignoriert wurde. Einige Techniken der konstruktionsorientierten Modellierung sind algebraische Spezifikationen, die praktisch nicht anwendbar sind, Zustandsdiagramme[5] und „*Example Based Programming*". Hiervon haben sich für den praktischen Einsatz die letzten beiden Ansätze als durchaus tauglich erwiesen.

Die softwaretheoretischen Ansätze haben sich mittlerweile bewährt. So existieren bspw. Ansätze, die auf dem Seeheim-Modell[6] basieren, welches ein Vorschlag für ein allgemeines Schichtenmodell ist. Ausprägungen dieses Modells sind z. B. das *MVC-Design Pattern* (vgl. [GHJV94], [SG96]) oder die heute sehr verbreiteten mehrschichtigen Anwendungssysteme (N-Tier Application, Client-Server Architekturen; vgl. Abbildung 7). Der Vorteil dieser Anwendungsklassen ist, dass die Benutzungsschnittstelle logisch und meist auch physisch von anderen Teilen der Anwendung entkoppelt ist. Somit kann sie relativ eigenständig entwickelt und auf die Bedürfnisse der Anwender zugeschnitten werden. In diesem Kontext ist dies natürlich von besonderem Interesse, da die Benutzungsschnittstelle vom eigentlichen Programmsystem getrennt betrachtet werden kann. Generell sollten Softwarespezifika eine sehr geringe Rolle beim UI-Design spielen.

4.1 UI-Entwurf

Beim Entwurf der Benutzungsschnittstelle gibt es nur eine allgemeingültige Regel: „*Schickes Design ist Geschmacksache.*" In der Tat ist es gerade wegen der Unberechenbarkeit der Komponente Mensch nicht möglich, ein optimales UI zu entwerfen und zu gestalten. Vielmehr ist das UI-Design meist eine Kombination aus Normen, physiologischen und psychologischen Aspekten, Erfahrungen, Typografie etc. Für die weiteren Betrachtungen werden ausschließlich dialogorientierte Anwendungen herangezogen. Diese können durch eine abgrenzbare Funktionalität charakterisiert werden. Die Nutzung solcher Systeme und die Interaktion erfolgt durch eine grafische Benutzeroberfläche. Als problematisch hierbei erweist sich die Tatsache, dass man GUI's[7] nicht ohne Weiteres wie den Rest eines Softwaresystems entwerfen kann. Außerdem sind GUI's sehr aufwendig in der Entwicklung. Deshalb wird die Konstruktion von GUI's von den meisten Entwicklern als Last empfunden und daher vernachlässigt. Die daraus resultierenden Konsequenzen liegen auf der Hand. Die durch mehrschichtige Architekturen postulierte Trennung zwischen UI und dem Rest des Systems kann sich bei der Entwicklung nachteilig auswirken. Viel gravierender ist jedoch die Tatsache, dass der Softwareentwickler hier zum Designer wird, während der Designer in Hinblick auf das Gesamtprojekt zum Modellierer wird. Dieser erzwungene Rollentausch kann sich ebenfalls nachteilig auswirken, sodass hier eine massive Fehlerquelle zu identifizieren ist. Es können auf diese Weise „Fehlentwicklungen" am Benutzer vorgenommen werden, sodass das Softwaresystem u.U. nur als Selbstzweck existiert.

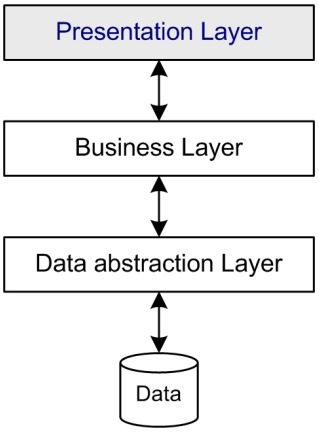

Abbildung 7: Mehrschichtige Anwendungsarchitektur

Häufig auftretende, technikgetriebene Fehler („*Sünden am Benutzer*") sind z.B. die Überfrachtung der Schnittstelle, eine unstrukturierte Schnittstelle, eine unsinnige Schnittstelle oder eine für ein bestimmtes Umfeld entworfene Schnittstelle.

Zur Lösung dieser Probleme empfiehlt es sich, die Normen und Richtlinien für grafische Benutzungsschnittstellen zu beachten und beim Entwurf zu berücksichtigen.

4.2 UI-Design Richtlinien

Die Entwicklung von GUI's ist zunächst von einigen pragmatischen Fragen getrieben:

- Für wen ist das GUI?

[2] GOMS–Goal Operator Method Selection Rules (Card, Moran und Newel, 1983)
[3] TAG–Task Action Grammar (Payne, 1985)
[4] UAN–User Action Grammar (Hartson, 1990)
[5] Parnas, 1969 und Jacob, 1985. Mittlerweile sind Zustandsdiagramme im Rahmen der UML ein gebräuchliches Modellierungswerkzeug.
[6] Benannt nach gleichnamiger Konferenz 1985
[7] GUI-Graphical User Interface

- Was soll dem Benutzer zur Verfügung gestellt werden?
- Welche Art Hilfe und wie viel ist notwendig?
- Ist das GUI vollständig und übersichtlich?

Diese pragmatischen Fragen führen durch den intuitiven Bereich der Benutzerkategorisierung und GUI-Modellierung. Sie werden unter Beachtung der Arbeits- und Vorgehensweise von Anwendern gestellt. Benutzer erschließen eine Anwendung durch die Benutzeroberfläche, da diese ihre einzige Möglichkeit ist, mit der Software zu interagieren. Dabei sollte die Software den Anwender unterstützen. Hierbei gibt es z. B. die Möglichkeit, Hilfestellung anzubieten. Die Funktionalität der Software soll also klar sein und durch den Nutzer erfasst werden können. Die Oberfläche soll einen Anwender nicht verwirren. Hierfür ist es ratsam, dass es einem Anwender nur zugemutet wird, auch wirklich nur notwendige Operationen auszuführen und nur notwendige Entscheidungen treffen zu müssen. Es sollte sich somit die Frage stellen, ob die GUI mit Features überfrachtet ist und ob wirklich alle Funktionen notwendig sind.

Ein weiterer Aspekt wird unter dem Begriff *„Design for Extremes"* (vgl. [Lel02]) betrachtet. Eine GUI sollte auch für „Randgruppen" benutzbar sein, d. h. auch für Anwender, die nicht den üblichen Benutzerprofilen entsprechen. Dies sind z. B. Analphabeten oder behinderte Anwender.

Beispiel 4.2 (Anwendung von Metaphern) *Ein Beipiel, wie dem Menschen unter Verwendung von Richtlinien und seines Erfahrungsschatzes der Zugang zu einem Softwaresystem erleichtert werden kann, ist die Anwendung von Metaphern. Metaphern bilden einen Zugang zum mentalen Weltbild des Anwenders und stellen Benzüge zu realen Sachverhalten dar. Abbildung 8 zeigt die s. g. Desktop Metapher. Sie bildet eine realen Schreibtisch auf einen virtuellen ab. Rechts unten ist bspw. ein Papierkorb zu sehen, der ebenfalls einer Metapher entspricht. In der realen Welt kann man ein Dokument, z. B. einen Brief, in den Papierkorb werfen. Damit ist er vorerst „gelöscht", kann jedoch jederzeit wieder aus dem Papierkorb herausgenommen werden. Erst beim leeren des Papierkorbs ist der Brief/das Dokument entgültig vernichtet.*

Beispiel 4.3 (Gefahren von Metaphern) *Die Anwendung von Metaphern hat zwei Seiten. Einerseits wird dem Anwender ein intuitiver Zugang zum System gewährt, da er Erfahrungen aus dem alltäglichen Leben auf das interaktive System überführen kann. Auf der anderen Seite hat die Anwendung von Metaphern natürlich auch Nachteile. Hierfür sei die „Ampel Metapher" genannt. Sie suggeriert, dass eine grüne Farbe positive Auswirkungen hat, eine rote negative. Dem entsprechend wäre es logisch, Bejaungen grün zu färben, Verneinungen rot.*
In dialogorientierten Anwendungen kann dies gravierende Konsequenzen haben, wenn man z. B. die Frage nach dem Löschen wichtiger Dateien wie folgt stellt: „Wollen Sie alle Dateien löschen?". Ein Anwender wird wahrscheinlich intuitiv auf JA drücken, da er eine falsche Assoziation mit der Farbe grün herstellt. Dieses Beispiel zeigt, dass die Anwendung von Metaphern mit Bedacht erfolgen sollte.
Eine weitere Gefahr ergibt sich aus der in Abbildung 8 gezeigten Desktop-Metapher. So ist der Menüpunkt „Beenden" im s. g. Startmenü zu finden. Man muss also auf „Start" klicken, um das System auszuschalten. Diese Anordnung ist fragwürdig, da sie in gewissem Maße widersprüchlich ist.

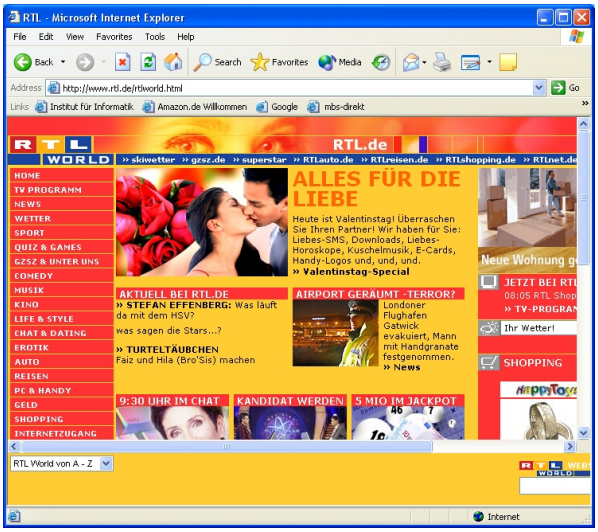

Abbildung 9: Multimedial überfrachtetes WebUI von RTL-World (`www.rtl.de`)

Diese gefühlsmäßigen und pragmatischen Aspekte werden zu einem großen Teil durch Normen und Richtlinien erfasst. Einige wichtige werden im Anschluss gezeigt, wobei zunächst der DIN-Begriff aufgeführt wird, welcher vom korrespondierenden ISO-Begriff in Klammern ergänzt wird. Von besonderm Interesse sind hier nicht die gesamten Normen, sondern nur Teile. Dies sind die ISO 9241, Teil 10, die DIN 66234, Teil 8 und die BildschArbV, Anhang zu § 4, Ziffer 20-22.

- Aufgabenorientiertheit (*suitability for the task*)
  Der Benutzer sollte nur mit Informationen versorgt werden, die er für die Bewältigung seiner Aufgaben auch benötigt. Insbesondere eine Reizüberflutung durch multiple Darstellungen von Informationen sollte vermieden werden. Dies senkt die Produktivität und irritiert den Anwender. (vgl. Abbildung 9 – Iritation/-Überfrachtung)
- Selbstbeschreibungsfähigkeit
  (*self-descriptiveness*)
  Das Anwendungsystem sollte den Benutzer dahingehend unterstützen, dass er die Funktionalität entweder durch Interpolation vorhandenen Wissens oder

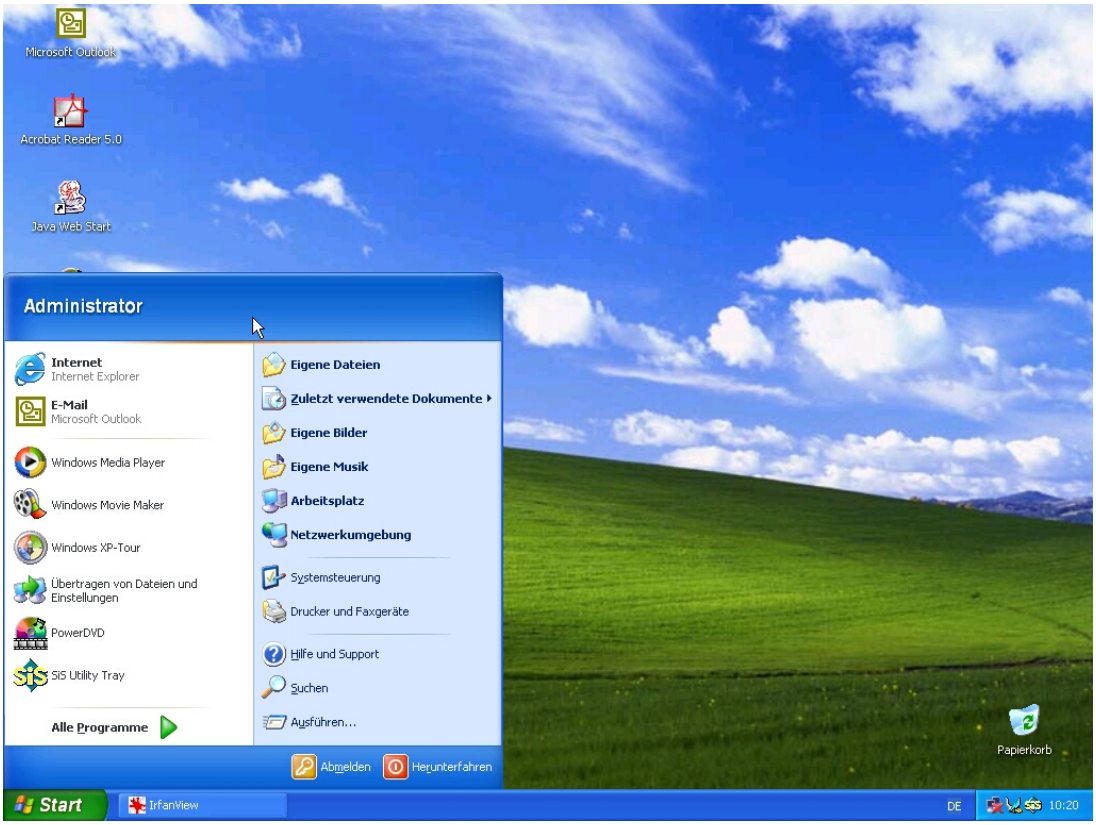

Abbildung 8: Erleichterung des Zugangs zur Software durch Anwendung von Metaphern

durch Interpretation der angebotenen Informationen erschließen kann. Hierfür empfehlen sich homogene Darstellungsformen, wie z. B. einheitliche Menüstrukturen oder Sinnbilddarstellungen (Icons). Auch die Verwendung von Metaphern ist angemessen.

- Steuerbarkeit (*controllability*)
  Der Anwender sollte stets vollständige Kontrolle über das System haben. Dies bedeutet, dass Aktion unterbrochen oder abgebrochen werden können. Bei sequentiellen Aktionssequenzen muss eine Wiederholbarkeit gewährleistet sein.

- Erwartungskonformität
  (*confirmity with user expections*)
  Eine Software soll den Erwartungen seiner Anwender genügen, d. h. dass ein Textverarbeitungsprogramm auch Textverarbeitung und Druckfunktionen unterstützt. Mit jeder Klasse von Anwendungssystemen werden grundlegende und standardisierte Funktionen assoziiert, welche enthalten sein müssen.

- Fehlerrobustheit (*error tolerance*)
  Softwaresysteme müssen Fehleingaben und Fehlverhalten der Anwender gegenüber tolerant reagieren. Hierbei ist es unagebracht, einen Vorgang durch eine Fehlermeldung entgültig zu beenden. Vielmehr ist es ratsam Korrekturfunktionalität vorzusehen, sodass z. B. Fehleingaben im Nachhinein korrigiert werden können.

- Individualisierbarkeit (*ability of individualization*)
  Software sollte auf bestimmte Bedürfnisse verschiedener Anwendergruppen anpassbar sein. Dies betrifft z. B. ein Weglassen von Unterstützung für geübte Anwender, während ungeübten Benutzern ein erweitertes Hilfeangebot zu Verfügung steht. Auch die Fähigkeit von Software mehrere Sprachen zu unterstützen ist diesem Punkt zuzuordnen.

- Erlernbarkeit (*learnability*)
  Erlernbarkeit kann z. B. durch Tutorials, die einer Software beiliegen, umgesetzt werden. Auch eine einfache, Wizzard gestützte Umgebung kann durch intuitiv zu nutzende Funktionalität einen positiven Lerneinfluss haben.

- Übersichtlichkeit
  Ein Anwender muss immer sofort erfassen können, in welchem Teil der Software er sich befindet, welche Aufgaben er an dieser Stelle erledigen kann und welche Aktionen er ausführen muss, um andere Funktionalitäten nutzen zu können. Dies ist bspw. im Hyperspace oft nicht der Fall, man spricht in diesem Fall vom „*Lost in Hyperspace*"-Syndrom.

Die Gestaltungsprinzipien sind nicht orthogonal und ergeben sich teils auch aus dem Systementwurf (benutzerorientiert, aufgabenorientiert). Die zu beachtenden Effekte im Umgang mit GUI's sind, dass bei jeder Nutzung von Software Lerneffekte zu verzeichnen sind. Diese gestatten es bei effektiver Ausnutzung, den Umgang mit der Software zu vereinfachen und effizienter zu gestalten.

Beispiel 4.4 (Software im Learning-Mode)
Verschiedene Softwaresysteme bieten einen s. g. Learning-Mode an, in dem sich die Software auf den Anwender einstellen kann (Vorlieben und Gewohnheiten feststellen etc.). Der Learning-Mode kann dann deaktiviert werden, wenn die Arbeitsweise des Systems den Vorstellungen des Anwenders entspricht.

Auch hierzu existieren Prinzipien, die jedoch großteils nicht aus Teilgebieten der Informatik entwickelt wurden, sondern ihren Ursprung in anderen Wissenschaftsdisziplinen haben. Beispielhaft zu nennen sind hier die Lenkung der Aufmerksamkeit, Hilfestellung (*Assistence*), Erfolgs- und Fortschrittsgefühl (*Satisfaction*) und Weitere.

Auch die Nutzung der menschlichen Denkweise, insbesondere der Art der Wissensaktivierung, kann vorteilhaft genutzt werden. Hierzu werden meist *Style Guides*, z. B. der CUA-Standard[8] verwendet, die firmenspezifische Software homogen erscheinen lassen (vgl. auch [UI 02b], [UI 02a], [MS 02]).

Beispiel 4.5 (Style Guides) *Der Aufbau von Menü und Toolbars in den Office-Produkten der Firma Microsoft ist homogen gestaltet. Dies umfasst Menüstrukturen und Aussehen von Piktogrammen, sodass ähnliche Funktionalität in unterschiedlichen Anwendungen durch Überführung ähnlichen Wissens ermittelt werden kann.*

Beispiel 4.6 (Style Guides und Metaphern) *In direkter Fortsetzung zu Beispiel 4.2 und Abbildung 8 soll nun der Arbeitsplatz (Abbildung 10) betrachtet werden. Der Gedanke des virtuellen Schreibtisches wird konsequent fortgesetzt. An einem Arbeitsplatz werden nicht nur Dokumente hantiert, sondern es stehen auch physische Medien zur Interaktion und Organisation zur Verfügung. Der Arbeitsplatz stellt einen Zugang zu diesen dar. Wie Abbildung 10 zu entnehmen ist, werden hier bereits Elemente eines Hypertextsystems verarbeitet, indem man kontextsensitive Verknüpfungen zu möglicherweise relevanten, ähnlichen Informationsträgern (virtuelle, physische) aufbaut.*
Ebenfalls werden Elemente gezeigt, welche die Rolle von Style Guides verdeutlichen. Hier sind z. B. Menüstrucktуren oder Piktogramme zu nennen, die ein einheitliches Erscheinungsbild der GUI bestimmen. Weiterhin ist hier bereits zu sehen, dass Sinnbilder in der Toolbar Semantik tragen, z. B. die Navigationsbuttons (Vor, Zurück) oder die

[8] Common User Access, [Dr.92]

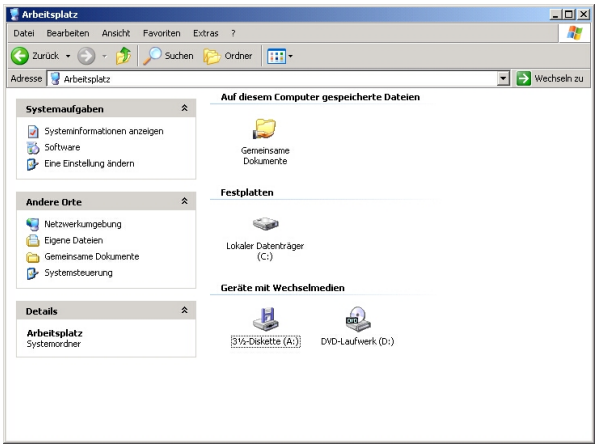

Abbildung 10: Weitergehende Anwendung der Desktop-Metapher

Lupe zum Suchen. Die konsequente Umsetzung dieses Erscheinungsbildes im gesamten System erleichtert den Umgang mit diesem und den Wiedererkennungswert einzelner Funktionalitäten.

5 Do's 'n' Don'ts beim User Interface Design

Für das Design einer Benutzungsschnittstelle gibt es keine Checkliste, die man abarbeiten kann und als Ergebnis eine ergonomische UI hat. Vielmehr lassen sich praktisch anwendbare Aussagen formulieren, die eine Mischung der o. g. pragmatischen Fragen und der Normen und Richlinien darstellt. Diese sollen im Folgenden aufgezeigt werden. Eine Benutzungsschnittstelle ist gut designt, wenn:

- Sie das macht, was der Benutzer will.
- Sie verhält sich so, wie es der Benutzer erwartet.
- Sie stellt nur sinnvolle und benötigte Operationen zur Verfügung.
- „Real-World"-Metaphern werden sinnvoll eingesetzt.
- Sie passt sich konsistent in das verwendete Basissystem ein, ohne einen optischen Bruch zu erzeugen.
- Sie führt und unterstützt den Benutzer ohne ihn zu bevormunden.

Diese Punkte sind formal nicht herleitbar, ergeben sich jedoch aus intuitiven Bedürfnissen des Menschen bei der Arbeit mit der Maschine. Der letzte Punkt ist hierbei besonders interessant. Ein System, dass den Benuzter führt und ihn unterstützt, erfährt eine sehr hohe Akzeptanz. Das selbe System wird jedoch nicht mehr akzeptiert, wenn es dem

Benutzer das Gefühl der Machtlosigkeit und des Ausgeliefertseins vermittelt. Dies ist ein sehr schmaler Grad zwischen brauchbaren und unbrauchbaren Softwaresystemen.

Beispiel 5.1 (Ein gutes Userinterface) *Ein gutes UI ist z. B. die in Webseite von „Heise Online"*[9] *(Abbildung 11). Zum Vergleich sei hier auf Abbildung 6 verwiesen. Hier er-*

Abbildung 11: Webseite von Heise Online

warten einen Benutzer eine klare Struktur („Corporate"-Schema, mehrere Onlineangebote unter einem Dach) und gut abgestimmte Farbschemata. Der Informationsgehalt ist ungleich höher als in Abbildung 6, ist jedoch weitaus leichter erfassbar.

Eine Benutzungsschnittstelle ist ergonomisch, wenn man folgende generellen Fehler beachtet und diese weites gehend eliminieren kann.

- Anwender lesen nicht, eigentlich NIE.
- Anwender lesen generell keine Mitteilungen in Dialogen, wenn diese mehr als eine Zeile umfassen.
- Anwender fühlen sich durch viele Rückfragen des Systems in Form von Dialogen genervt.
- Profis fühlen sich durch übereifrige Assistenten oft bevormundet und beleidigt.
- Tastaturfanatiker hassen die Maus!
- Mausfanatiker hassen die Tastatur!
- Beim Entwurf sollte man nicht kreativ sein – *„schlicht ist schick"*.

[9] http://www.heise.de

6 Kann man Softwareergonomie lehren?

Dies ist die Frage, die man sich zum Ende dieser Arbeit stellen sollte. Prinzipiell kann man, unter Beachtung aller hier angerissenen Aspekte, die Frage zunächst verneien. Man muss jedoch an dieser Stelle differenzieren. Softwareergonomie, insbesondere ergonomische Userinterfaces, ist ein so komplexes und schwieriges Gebiet, da hier der Mensch als große Unbekannte berücksichtigt werden muss. Die Frage ist nun, wie kann man ein solches Gebiet unter Berücksichtigung des Menschen lehren kann? Die Problematik liegt hier viel weniger in der Information oder deren Repräsentation, sondern vielmehr in der Individualität. Diese kann man aber nicht ohne Einschränkung erfassen, denn man kann es nicht allen recht machen.

Interessant für eine Lehre der Softwareergonomie sind die Aspekte, die einerseits im interdisziplinären Tätigkeitsbereich der HCI zu finden sind, andererseits aber auch die technischen Aspekte des Softwareengineering.
Im interdisziplinären Bereich zeigen sich interessante Perspektiven im Bereich der Physiologie, Psychologie und Kongnitionswissenschaften. Diese können das fachübergreifende Verständnis der Informatik positiv beeinflussen, sodass Software nicht mehr nur rein betriebswirtschaftlichen Prozessen genügt, sondern auch den Ansprüchen der Anwender mit einem vertieften Verständnis für ihre Bedürfnisse entgegenkommt. Dies ist im Erachten des Autors ein wesentlicher und notwendiger Schritt in der Evolution moderner Softwaresysteme.
Vom softwaretechnologischen Aspekt bieten sich ebenfalls interessante Teilgebiete der HCI für eine Vermittlung in der Lehre an. Interessant ist die Bewertung von Softwaresystemen nach ergonomischer Qualität. Hierfür müssen Metriken und Methodiken entwickelt, angewendet und vermittelt werden, die es gestatten softwareergonomische Aspekte bereits beim Entwurf von Softwaresystemen abzuschätzen und umzusetzen.
Die Konsequenz ist, dass Softwareengineering die Softwareergonomie in der Lehre berücksichtigen muss. Dies hat eine Öffnung und ein Verständnis für andere Fachgebiete zur Folge, sodass Software über kurz oder lang nicht mehr als technologischer Selbstzweck verstanden wird.

Literatur

[Döl02] DÖLLNER, J., PROF.DR.: *Benutzerschnittstellen*. Skript zur Vorlesung, 2002.

[Dr.92] DR. MANDEL: *Object-Oriented Interface Design: IBM Common User Access Guidelines*. IBM Corporation, 1992.

[GHJV94] GAMMA, E., R. HELM, R. JOHNSON und J. VLISSIDES: *Design Patterns. Elements of*

Reusable Object-Oriented Software. Addison-Wesley, 1994.

[HCI02] *Übersicht über die Arbeiten der HCI*. Online, `http://www.tau-web.de/hci`, 2002.

[ISO01a] *ISO 9241 Status*. Online, `http://www.system-concepts.com/stds/status.html`, 2001.

[ISO01b] *ISO 9241*. Online, `http://www.iso-standards-international.com/iso9241-kit9.htm`, 2001.

[Lel02] LELIĆ, ŠENAJ (Herausgeber): *UI Design für Einsteiger - Erstellung effektiver Anwendungsoberflächen*, msdn Tech Talk, 2002.

[MS 02] *Microsoft Research Group UI Design*. Online, `http://www.research.microsoft.com/research/ui`, 2002.

[SG96] SHAW, M. und D. GARLAN: *Software Architecture - Perspectives on an emerging discipline*. Prentice Hall, 1996.

[UI 02a] *Allgeimeine Informationen und Links zu Style Guides*. Online, `http://www.stcsig.org/usability/newsletter/0104-style.html`, 2002.

[UI 02b] *Allgeimeine Informationen und Links zu UI Standards*. Online, `http://www.isii.com/ui_design.html`, 2002.

[Wir] WIRTH, STEFAN: *Ergonomie und Gesundheit am Bildschirmarbeitsplatz*. Softwareergonomische Gestaltung von Bildschirmmasken und Computerprogrammen.