RWTH Aachen
Media Computing Group
Prof. Dr. Jan Borchers

Human Computer Interaction
Proseminar SS 2007

Designregeln

Eduard Renz und Robert Morys

19.04.2007

Inhaltsverzeichnis

1. Einleitung

Die Human-Computer Interaction (HCI) als Teilgebiet der Informatik beschäftigt sich mit der benutzergerechten Gestaltung von interaktiven Systemen und ihren Mensch Maschine-Schnittstellen. Dabei sind bestimmte Regeln erforderlich, um die Benutzerfreundlichkeit dieser Systeme zu erhöhen.
Da viele Software- bzw. Hardwaredesinger nicht über das nötige psychologische und ergonomische Wissen verfügen, wurden bestimmte Richtlinien, Regeln und Standards festgelegt, die den Entwicklern beim Designprozess zur Verfügung stehen. Außerdem bilden sie eine Grundlage für die gemeinsame Arbeit mehrerer Entwickler an einem Projekt. Dadurch profitieren einerseits die Entwickler selbst, da ihnen die Arbeit erleichtert wird und zum anderen auch die Anwender, da die Benutzerfreundlichkeit steigt.
Das Ziel dieser Ausarbeitung ist es, einen Einblick in die Bestandteile solcher Regeln zu geben, ihre Wichtigkeit zu demonstrieren und Vor- bzw. Nachteile aufzuzeigen.

2. Regeln zur Steigerung der Benutzerfreundlichkeit

> „Benutzerfreundlichkeit [...] bezeichnet die vom Nutzer erlebte Nutzungsqualität bei der Interaktion mit einem System. Eine besonders einfache, zum Nutzer und seinen Aufgaben passende Bedienung wird dabei als benutzerfreundlich angesehen.“ [1]

2.1 Erlernbarkeit

Um neue und unerfahrene Benutzer nicht durch ein zu komplexes System abzuschrecken, sollte dieses einen einfachen Einstieg bieten und leicht erlernbar sein. Um das zu gewährleisten, sollten die im Folgenden beschriebenen Grundsätze beachtet werden.

2.1.1 Vorhersagbarkeit

Benutzerfreundliche Software hat die Eigenschaft, dass der Anwender immer vorhersagen kann, was als nächstes passiert. Diese Fähigkeit entsteht aus vergangenen Erfahrungen mit dem Programm. Zusätzlich sind viele Systeme so konzipiert, dass sie maximal vorhersehbar sind und ein Benutzer somit zu jedem Zeitpunkt weiß, zu welchem Resultat seine Handlung führen wird. Beispielsweise könnte man eine Installation eines beliebigen Programms auch dann fortführen, wenn nur das gewünschte Endresultat bekannt ist und man mitten in den Installationsprozess einsteigt.
Eine andere Form der Vorhersagbarkeit wird als *Operationssichtbarkeit* bezeichnet, welche das *Prinzip des Wiedererkennens statt Erinnerns* verfolgt. Das bedeutet, dass sich der Benutzer nicht alle relevanten Informationen merken muss, sondern diese vom System angezeigt bekommt. Des Weiteren impliziert *Operationssichtbarkeit*, dass dem Benutzer wichtige Optionen immer angezeigt werden und er nicht all diese im Hinterkopf behalten muss.

---

1 Artikel *Benutzerfreundlichkeit.* In: Wikipedia, Die freie Enzyklopädie. Bearbeitungsstand: 26. März 2007, 17:50 UTC. URL: http://de.wikipedia.org/w/index.php?title=Benutzerfreundlichkeit&oldid=29709251 (Abgerufen: 29. März 2007, 14:11 UTC)

2.1.2 Nachvollziehbarkeit

Durch die Fähigkeit des Anwenders die Folgen vergangener Aktionen auf die aktuelle Situation zu übertragen und aufgrund der Reaktion des Systems entwickelt der Benutzer ein mentales Modell. Die Basis für die Nachvollziehbarkeit bietet das *Prinzip der Ehrlichkeit*. Das heißt, dass wenn eine Aktion den Zustand des Systems verändert, dann muss dies dem Benutzer, vor allem bei wichtigen Informationen, nach Möglichkeit sofort mitgeteilt werden. Bei nicht sofort angezeigten Änderungen sollte zumindest die Möglichkeit bestehen den veränderten Zustand manuell anzufordern.
Ein Beispiel hierfür liefert der Vergleich zwischen grafischen Oberflächen (GUI) und Kommandozeileninterpretern (CLI). Bei GUIs werden Dateien durch Icons dargestellt und können per *Drag & Drop* einfach und intuitiv verschoben werden. Der Benutzer sieht somit sofort welche Veränderung seine Aktion bewirkt hat. Bei CLIs hingegen ist so eine Veränderung nicht sofort sichtbar. Der Benutzer muss die für das Verschieben einer Datei relevanten Informationen (Ordnername, Pfad etc.) explizit anfordern und danach sogar überprüfen ob die Datei tatsächlich verschoben und nicht kopiert wurde.
Ein weiteres Problem, das bei nicht sofortiger Anzeige der Veränderungen auftreten kann, kann man an der *Suchen & Ersetzen* - Funktion in einem Textverarbeitungsprogramm darstellen. Hat der Benutzer z.B. einen Fehler gemacht und möchte jedes Vorkommen von „der“ durch „das“ ersetzen, so wird auch beispielsweise „der“ im Wort „wieder“ ersetzt, was eigentlich nicht beabsichtigt war. Zeigt das Programm nicht alle Ersetzungen, beispielsweise durch markieren, an, so bleibt dieser Fehler wahrscheinlich unbemerkt.

2.1.3 Vertrautheit

Sowohl Erfahrungen aus dem Alltagsleben als auch das Wissen über andere Computersysteme helfen uns neue und unbekannte Systeme schneller und einfacher zu erlernen.
Vertrautheit bedeutet, dass der Anwender sein existierendes Wissen gut auf das neu zu Erlernende anwenden kann und eine Beziehung zwischen dem Bekannten und dem Neuen herstellen kann. Dies wird unter anderem durch die Verwendung von Metaphern wie etwa der Symbolisierung des Arbeitsplatzes im Büro durch den Desktop mit seinen Dokumenten und dem Papierkorb unterstützt.
Allein durch die Existenz von bestimmten, anderweitig bekannten Objekten kann das Verhalten des Anwenders beeinflusst werden. So weiß der Anwender z.B. beim Anblick einer halb-geöffneten Tür dass er durch einen Klick auf diesen Button das Programm verlassen kann. Geschickter Einsatz solcher Mittel kann dazu genutzt werden um beim Anwender vertraute Situationen herzustellen und ihm somit den Umgang mit dem System zu erleichtern.

2.1.4 Konsistenz

Eines der wichtigsten Prinzipien beim Design einer benutzerfreundlichen Oberfläche ist die Konsistenz. Grundsätzlich gehen Anwender von einem gleich bleibenden System aus. Konsistenz hilft dem Benutzer, sich durch invarianten Einsatz von bestimmten Mitteln in unbekannten Situationen besser zurechtzufinden. So ist es z.B. für den Benutzer angenehmer, wenn er beim Navigieren durch eine Webseite das Menü immer an der gleichen Stelle vorfindet.

Wie auch in dem Beispiel zu sehen ist, ist Konsistenz keine direkte Eigenschaft, sondern muss relativ zu etwas angewendet werden. Somit betrachtet man Konsistenz nicht im Allgemeinen sondern in Bezug auf Farbe, Positionierung, Benennung der Dateien, uvm.. Jedoch ist Konsistenz nicht immer von Vorteil. Betrachtet man etwa frühere Exemplare des Tastaturlayouts (ABCDEF-Anordnung), so war dieses zwar konsistent, hatte aber die ungleichmäßige Verteilung der Auslastung der zehn Finger als Nachteil. Deshalb hat sich letztendlich die QWERTY-Anordnung durchgesetzt.

2.1.5 Generalisierung

Unter Generalisierung versteht man die Verallgemeinerung von bereits vorhandenem Wissen um dieses in einer noch unbekannten, aber ähnlichen Situation einsetzen zu können. Damit versucht der Anwender sein Modell vom System zu erweitern und es gleichzeitig zu vervollständigen. Beispielsweise kann ein Benutzer, dem die *Copy & Paste* Funktion aus einem Textverarbeitungsprogramm bekannt ist, diese auch in anderen Programmen benutzen, sofern die Entwickler der Software auf das Prinzip der Generalisierung geachtet haben.

2.2 Flexibilität

Flexibilität stellt die Anzahl der Möglichkeiten mit denen der Anwender und das System miteinander kommunizieren können dar.

2.2.1 Initiative

In bestimmten Situationen ist es vorteilhaft, wenn der Benutzer die Initiative über das System übernehmen kann, in anderen wiederum ist es sinnvoll wenn das System den Anwender einschränkt. Man unterscheidet also zwei Formen der Dialogführung:
Zum einen *benutzerbevorrechtigt*, was bedeutet, dass der Anwender dem System vorgibt, welche Aktionen es ausführen soll und zum anderen s*ystembevorrechtigt* bei welcher die Handlungsfreiheit des Anwenders einschränkt wird. In diesem Fall leitet das System den Benutzer welcher nur auf dessen Anfragen reagieren muss.
Aus der Sicht des Anwenders unterstützt die benutzerbevorrechtigte Form der Dialogführung die Flexibilität des Systems wohingegen die systemberechtigte Form diese unterbindet. Im Allgemeinen will man die Vereinnahmung des Benutzers durch das System vermeiden. Dies ist in bestimmten Situationen jedoch nicht möglich, wenn etwa die Sicherheit des Systems gefährdet werden würde. Betrachtet man beispielsweise das Sicherheitskonzept von UNIX Systemen in denen Administrations- und Benutzerkonten strikt voneinander getrennt werden, dann sieht man, dass die Benutzerfreiheiten zugunsten der Sicherheit eingeschränkt werden.
Bei zu viel Flexibilität hingegen kann der Benutzer die Übersicht verlieren, da es passieren kann, dass er noch zu erledigende Aufgaben beiseite legt um immer wieder neue zu starten und sich somit in einem Dschungel von Aufgaben im System verirrt.

2.2.2 Aufgabenverteilung

Damit ein System möglichst effizient arbeitet, sollte es dem Benutzer und dem System möglich sein, Aufgaben an den jeweils anderen abzugeben, damit so eine optimale Zusammenarbeit zwischen PC und Anwender entsteht.
Ein gutes Beispiel dazu bietet die Rechtschreibprüfung in einem Textverarbeitungsprogramm. Hierbei kann der Anwender diese Aufgabe dem Programm überlassen, solange es um Wörter geht, die auch im Wörterbuch zu finden sind. Bei Namen ist das Programm jedoch überfordert und gibt die Aufgabe der Korrektur an den Benutzer zurück.
Somit sieht man, dass sich durch dieses Prinzip Mensch und Maschine optimal ergänzen, so dass der Benutzer die fundamentalen Aufgaben an den Computer abgeben kann um so Zeit zu sparen. Andererseits ist der Computer mit Hilfe des Menschen in der Lage Aufgaben, die seine Fähigkeiten überschreiten, zu bewältigen.

2.2.3 Ersetzbarkeit

Bei der Ersetzbarkeit geht es darum, dass der Benutzer einen beliebigen Wert gegen einen anderen austauschen kann und das System ihm dies so einfach wie möglich gestaltet. Nehmen wir als Beispiel ein Bildbearbeitungsprogramm mit dem der Benutzer eine Visitenkarte erstellen möchte und ihm die Größenangabe nur im Millimetermaß bekannt ist. Würde das Programm jedoch nur inch-Eingaben akzeptieren, so ist ein mancher Benutzer überfordert, da er den Wert nicht umrechnen kann.
Ersetzbarkeit kann man aber nicht nur auf die Eingabe sondern auch auf die Ausgabe beziehen. So kann es von Vorteil für den Anwender sein, wenn ihm verschiedene Grafiken zur Ausgabe zur Verfügung stehen und er je nach Bedarf zwischen diesen hin und her wechseln kann, sei es ein Graph, Diagramm, Tabelle oder ähnliches.
Somit gewinnt ein System an Benutzerfreundlichkeit, wenn es dem Anwender möglichst viele Varianten bietet, ihm unnötige Kopfarbeit abnimmt und es ihm ermöglicht verschiedene Werte mit Leichtigkeit auszutauschen.

2.2.4 Anpassbarkeit

Eine große Stärke von Computersoftware ist ihre Anpassungsfähigkeit. Darunter versteht man die Eigenschaft der Benutzeroberfläche sich zu verändern oder die Möglichkeit bereitzustellen verändert zu werden. Die Veränderungen können dabei entweder vom System oder vom Benutzer durchgeführt werden. Der Anwender kann dabei ein unterschiedliches Maß an Freiraum für die Modifikationen erhalten. Angefangen bei oberflächlichen Veränderungen die nur die optische Gestaltung beeinflussen und die Struktur der Anwendung unberührt lassen, über Makros die es dem Benutzer erlauben die Bearbeitung häufig durchgeführter Aufgaben zu beschleunigen bis hin zur Integration von Programmiersprachen in die Oberfläche (siehe UNIX).
Beispiele hierfür sind auch Programme die den Anwender beim Start nach seinem Wissensstand fragen und entsprechend die Oberfläche anpassen.
Der Anwender spielt hier bei der Anpassung also eine direkte Rolle. Im Gegensatz dazu nimmt der Benutzer bei Veränderungen die direkt durch das System vorgenommen werden eine passive Rolle ein.
Die Grundlage für die Veränderung der Benutzeroberfläche durch das System ist das

Wissen des Systems über den Benutzer. Die Software beobachtet, welche Aktionen der Anwender besonders oft ausführt und versucht zusätzlich mithilfe des Wissens über den Erfahrungsstand des Anwenders die Oberfläche dementsprechend anzupassen.

2.3 Robustheit

Systeme die die Zusammenarbeit zwischen Mensch und Maschine fördern und dadurch das Erfüllen von Aufgaben ermöglichen, werden als robust bezeichnet.

2.3.1 Beobachtbarkeit

Der Benutzer sollte immer die Möglichkeit haben, Informationen über den Status des Systems bzw. bestimmter Teilbereiche dieses zu erhalten.
Dies wird durch die Prinzipien der *Erforschbarkeit*, *Standards*, *Erreichbarkeit*, *Persistenz* und *Operationssichtbarkeit* (siehe 2.1.1) gewährleistet.
Durch die *Erforschbarkeit* wird es dem Benutzer ermöglicht zusätzliche Informationen über das System abzurufen, die standardmäßig nicht auf der Benutzeroberfläche angezeigt werden können. Man kann sich dies am Beispiel eines Dateibrowser klarmachen: So werden bei einem Solchen standardmäßig nur wenige, ausgewählte Informationen, wie z.B. Dateigröße oder Erstellungsdatum, angezeigt. Der Benutzer hat jedoch die Möglichkeit bei Bedarf zusätzliche Attribute wie Autor oder Berechtigungen abzurufen.
Standards bieten dem Benutzer einerseits eine Basis, worauf er seine Arbeit aufbauen, zum anderen aber auch Leitwerte nach denen er sich richten kann, was somit das Verständnis des Benutzers über das System fördert. Außerdem werden dadurch Fehler vermieden und dem Benutzer die Arbeit erleichtert. So existiert bei einem Bildbearbeitungsprogramm z.B. eine DINA4 Vorlage, so dass der Anwender nicht die Dimension eines solchen Formats auswendig kennen muss.
Erreichbarkeit bedeutet, dass der Benutzer zu jedem Zeitpunkt zu anderen aufgabenrelevanten Zuständen gelangen kann. Unterschiedliche Ausprägungen der Erreichbarkeit wirken sich auf die Flexibilität der Software aus.
Mit Hilfe der *Persistenz* werden dem Benutzer bestimmte Informationen über längere Zeit angezeigt. Dabei gilt für verschiedene Ausgabekanäle, dass sie eine differenzierte Ausprägung an Persistenz aufweisen. So sind auditive Signale, wie z.B. ein Piepton beim Erhalt einer Email nicht persistent wohingegen visuelle Signale, wie etwa ein dauerhaft angezeigtes Icon, das über den Eingang neuer Emails informiert, persistent.

2.3.2 Wiederherstellbarkeit

Jeder Mensch macht Fehler, die er im Nachhinein am Liebsten wieder rückgängig machen würde. Im Gegensatz zum Alltagsleben kann ein Computersystem ihm diese Möglichkeit bieten. So kann er beispielsweise einen falsch gezeichneten Strich durch ein einfaches klicken auf „Rückgängig“ verschwinden lassen oder eine aus Versehen gelöschte Textpassage wiederherstellen.

2.3.3 Antwortverhalten

Das Antwortverhalten bezeichnet das Maß an Kommunikation zwischen dem System und dem Benutzer. Die Antwortzeit ist definiert als die Zeit, die das System für das Anzeigen einer Zustandsänderung benötigt wobei möglichst kurze Reaktionszeiten erwünscht sind. Falls eine sofortige Reaktion des Systems nicht möglich ist, dann sollte dieses dem Benutzer mitteilen, dass es seine Anfrage erhalten hat und diese gerade bearbeitet. Dies kann z.B. durch die Transformation des Mauszeigers in eine Sanduhr erreicht werden. Des Weiteren sollte das Antwortverhalten zeitlich konsistent sein um den Benutzer nicht unnötig zu verwirren. So sollte etwa ein Menü nach einem Mausklick immer nach der gleichen Zeitspanne erscheinen.

2.3.4 Aufgabenerfüllung

Eine wichtige Frage ist, in wie fern ein System eine gewünschte Aufgabe erfüllen kann und wie weit dies mit den Vorstellungen des Benutzers übereinstimmt. Dabei ist es vorteilhaft, wenn das System allgemein einsetzbar ist und es dem Benutzer ermöglicht selbstständig neue Aufgaben, für die das System nicht direkt ausgelegt worden ist, zu lösen.
Betrachtet man etwa ein Datenbankmanagementsystem (z.B. Microsoft Access), welches die Entwicklung von Datenbankanwendungen ermöglicht, so sieht man, dass damit eine Vielzahl von unterschiedlichen Aufgaben gelöst werden kann, obwohl die Software nicht direkt für all diese konzipiert wurde.

2.4 Bewertung der Benutzerfreundlichkeit

Die Benutzerfreundlichkeit eines Systems ist teilweise eine subjektive Beurteilung. Weitgehend aber durch Konsistenz in der Bedienung, einprägsame und unterscheidbare Symbolik und eine für die Benutzergruppen passende Begriffsnutzung zu erreichen. Während der gesamten Entwicklungsphase eines Systems sollte die Benutzerfreundlichkeit stetig von nicht in den Designprozess eingebundenen Personen getestet werden. Das heißt, dass der Designer das System nicht nur nach seinen Vorstellungen entwickeln sollte, sondern auch Rückmeldungen von Benutzern einholen muss oder diesen sogar in den Entwicklungsprozess als Co-Designer mit einbezieht (siehe 2.5 Anwenderbezogenes Design).
Wenn es um die Bewertung eines Systems geht, können Fragen, wie „Kann das Produkt die gewünschte Aufgabe erfüllen?“ oder auch „Wie leicht kann man als neuer Benutzer effektiv mit dem System interagieren?“ hilfreich sein. Außerdem tragen zu der Bewertung noch Eigenschaften wie Funktionalität, das Erscheinungsbild und auch die physikalischen Attribute bei. Jedoch sollte man sich darüber im Klaren sein, dass es keine allgemeinen Schemata gibt, um die Benutzerfreundlichkeit explizit zu messen.
Die Missachtung von Prinzipien zur Steigerung der Benutzerfreundlichkeit hat weitreichende Auswirkungen, da der Benutzer von dem Produkt abgeschreckt werden würde und zukünftig die Marke meiden würde.

Quelle: (3) – siehe Literaturverzeichnis

2.5 Anwenderbezogenes Design

Wenn beim Designprozess Benutzer der Zielgruppe explizit eingebunden werden, dann spricht man von *anwenderbezogenem Design.* So kann z.B. ein Kind sehr hilfreich bei der Entwicklung von Software für junge Altersgruppen sein.
Ein solches Vorgehen bringt einige Vorteile mit sich, da es hilft, die Erwartungen und das Maß an Zufriedenheit der Benutzergruppen zu erhöhen. Außerdem fällt es dem Anwender so leichter, sich mit dem Produkt zu identifizieren. Durch die Zusammenarbeit entstehen ggf. mehr kreative Ideen die sonst unbeachtet geblieben wären.
Diese Vorteile haben natürlich ihren Preis da mit ihnen hohe Kosten und ein hoher Zeitverbrauch verbunden sind. Außerdem ist es empfehlenswert für die Auswertung der Benutzerrückmeldungen Spezialisten (z.B. Psychologen) einzusetzen, was Sichtweisen wiederum einschränkt. Schwierigkeiten können auch beim Auswerten und Umsetzen der von den Benutzern gesammelten Daten entstehen. Bei der Einbeziehung der Benutzer könnte außerdem das Endprodukt zu spezifisch werden.

Quelle: (2) – siehe Literaturverzeichnis

3. Standards

Standards sind Design-Regeln die von verschiedenen Institutionen veröffentlicht werden um den Entwicklern von Soft- und Hardware bestimmte Vorgaben zu geben. Dabei beruhen Hardware Standards eher auf physiologischem Verständnis wohingegen Software Standards psychologisch begründet sind.
Ein gutes Beispiel eines solchen Standards liefert ein Auszug aus dem ISO[2] Standard 9241, welcher darüber informiert woran Benutzerfreundlichkeit sowohl in Bezug auf Soft- als auch Hardware, gemessen wird:

> Benutzerfreundlichkeit – Die Effektivität, Effizienz und Zufriedenheit mit der ein beliebiger Benutzer ein beliebiges Ziel in einer bestimmten Umgebung erreichen kann.
>
> Effektivität – Die Genauigkeit und Vollständigkeit mit der ein beliebiger Benutzer ein bestimmtes Ziel in einer bestimmten Umgebung erreichen kann.
>
> Effizienz – Die investierte Arbeit in Bezug zu der Genauigkeit und Vollständigkeit mit der das Ziel erreicht wird.
>
> Zufriedenheit – Der Komfort und die Akzeptanz die das System dem Benutzer und anderen Leute die davon Gebrauch machen, bietet.

Ein Vorteil solcher Standards ist, dass sie eine große Gemeinschaft von Entwicklern dazu veranlassen, bestimmte Regeln einzuhalten um so ein einheitliches System zu schaffen. Viele dieser Regeln, vor allem im Bereich der Software, stellen nur eine Anregung und keine Pflichtvorgabe dar.

---

2 ISO steht für „International Organization for Standardization“ und ist eine der bekanntesten Institutionen, die Design Standards veröffentlichen

4. Richtlinien

Da der Bereich der HCI noch sehr neu ist, enthält auch die Theorie dahinter noch Lücken, wodurch es an Durchsetzungsvermögen von Standards mangelt. Deshalb sind viele Regeln in diesem Bereich eher Vorschläge und Richtlinien nach denen sich ein Designer richten kann, aber nicht muss. Diese Richtlinien werden auch als *Stilratgeber* bezeichnet und können dabei unterschiedlich genau und konkret sein.
Abstrakte Richtlinien spiegeln dabei eher eine Philosophie oder Einstellung wieder und sind unabhängig von konkreten Soft- oder Hardwareimplementierungen. Solche Richtlinien beziehen sich eher auf die zuvor vorgestellten Werte wie Erlernbarkeit, Robustheit und Flexibilität.
Dagegen sind konkrete Richtlinien auf ein bestimmtes System bezogen, stellen eine genaue Umsetzung der abstrakten Richtlinien dar und sind meistens nicht allgemein verwendbar.
Gut sichtbar wird dies am Beispiel der *Nomen-Verb* - Richtlinie bei Apple. Dabei soll alles auf dem Prinzip „mit welchem Objekt“ (Nomen) „soll welche Aktion durchgeführt werden?“ (Verb) beruhen. Da dieses Prinzip in allen Bereichen einer Software angewendet werden muss, ist es konsistent. Ein weiterer Auszug aus der abstrakten Richtlinie von Apple besagt, dass der Anwender und nicht der Computer die Initiative und Kontrolle über alle Aktionen haben soll, da dies die Flexibilität eines Systems fördert.
Eher konkrete Richtlinien besagen, dass große Menüs in kleine Gruppen zusammengefasst werden sollen um dem Benutzer das Finden von Einträgen zu erleichtern. Dabei wird als Ausgangspunkt die Tatsache genommen, dass es für Menschen einfacher ist etwas wieder zu erkennen anstatt sich an etwas zu erinnern. Durch die Gruppierung wird diese Tatsache in den Designprozess mit eingebunden.
Zentraler Bestandteil vieler allgemein gehaltener Richtlinien sind die *Dialogstile,* welche die Kommunikation des Benutzers mit dem System kategorisieren. Diese können nach Bedarf auch gemischt werden um eine Steigerung der Benutzerfreundlichkeit zu erreichen. Im folgenden ist eine Übersicht über die von „Smith & Mosier“ aufgeführten Dialogstile gegeben:

- Frage und Antwort
- Formularausfüllung
- Menüauswahl
- Funktionstasten
- Kommandozeilensprachen
- Anfragesprachen
- Natürliche Sprache
- Grafische Auswahl

5. Goldene Regeln

Zuvor vorgestellte Prinzipien und Richtlinien unterstützen den Designer dabei, benutzerfreundliche Systeme zu entwerfen. Dabei muss der Designer jedoch entweder selber die Prinzipien für den aktuellen Fall neu interpretieren oder Einschränkungen durch die Richtlinien hinnehmen. Eine Kombination der beiden Prinzipien stellen die so genannten *Goldenen Regeln* dar, die dem Designer eine Hilfestellung bieten sollen. Sie

können als eine Art Checkliste angesehen werden und können teilweise auch für die Evaluierung verwendet werden.
Einige der wichtigsten und am meisten verwendeten *Goldenen Regeln* sind die von Shneiderman und Norman, die eine Art Zusammenfassung der zuvor vorgestellten Prinzipien darstellen.

5.1 Shneidermans acht goldene Regeln des Interface Designs

1. **Konsistenz**
   Beschreibung: Siehe 2.1.4.
   Beispiel: Benenne Aktionen immer gleich und verwende nicht für das Löschen eines Objekts verschiedene Bezeichnungen wie „Entfernen“, „Löschen“, „In den Papierkorb legen“ usw..
2. **Shortcuts & Hotkeys**
   Beschreibung: Gib erfahrenen Benutzern die Möglichkeit Aktionen schneller auszuführen.
   Beispiel: Strg + C zum Kopieren und Strg + V zum Einfügen.
3. **Feedback**
   Beschreibung: Gib dem Benutzer angebrachte Informationen, so dass dieser sich ein genaues Bild des Systemzustands machen kann. Einfache Aktionen erfordern nur wenige Informationen, komplexe/weitreichende dagegen mehr.
   Beispiel: Das Kopieren einer Datei erzeugt wenig Feedback wohingegen das Installieren eines Programms viel Feedback erzeugt.
4. **Dialoge**
   Beschreibung: Zeige dem Benutzer, wenn er eine Aufgabe abgeschlossen hat.
   Beispiel: Download beendet.
5. **Fehlerbehandlung und Fehlerbearbeitung**
   Beschreibung: Benutzer sollten davon abgehalten werden, Fehler zu machen und falls diese doch auftreten, dann sollten Informationen zur Verfügung gestellt werden wie der aufgetretene Fehler behoben werden kann.
   Beispiel: „Problemberichte und Lösungen“ in Windows Vista.
6. **Wiederherstellungsmöglichkeiten**
   Beschreibung: Benutzer kann immer zu einem vorigen Zustand des Systems zurückkehren und hat somit weniger Scheu dieses zu erforschen.
   Beispiel: Systemwiederherstellung von Windows.
7. **Kontrollierbarkeit**
   Beschreibung: Der Benutzer sollte die Kontrolle über das System haben.

8. **Gedächtnisleistung minimieren**
   Beschreibung: Versuche, alles so einfach wie möglich zu halten und dem Benutzer Gedächtnisleistung abzunehmen.
   Beispiel: Emailnotifikation, Wizards.

Diese Regeln können natürlich nicht überall angewendet werden, bieten aber oft eine gute Orientierung.

5.2 Normans sieben Prinzipien zur Vereinfachung von Aufgaben

1. **Erfahrungen und Wissen benutzen**
   Wissen aus dem Alltagsleben sollte nach Möglichkeit auch auf das System übertragbar sein.
2. **Aufgabenvereinfachung**
   Aufgaben sollten einfach gehalten werden um komplexe Lösungen zu vermeiden. Hilfreich sind dabei zusätzliche Informationen, Feedback und Automatisierung. Dabei darf dem Benutzer jedoch nicht die Kontrolle entrissen werden.
3. **Sichtbarkeit**
   Es sollte klar zu erkennen sein, was das System leisten kann, wie man bestimmte Aufgaben lösen kann und welchen Effekte bestimmte Aktionen haben.
4. **Abbildung**
   Die Absichten und Ziele des Benutzers sollten sich in der Steuerung des Systems widerspiegeln. Dabei müssen z.B. kleine Veränderungen auch kleine Auswirkungen haben.
5. **Grenzen**
   Natürliche und künstliche Grenzen sollten so eingesetzt werden, dass es unmöglich ist, etwas anderes als die richtige Handlung in der richtigen Reihenfolge durchzuführen. Der Benutzer sollte also durch die Aufgabe geleitet werden.
6. **Design für Fehler**
   Mögliche Fehler des Benutzers sollten im System berücksichtigt werden. Zusätzlich sollten Wiederherstellungsoptionen angeboten werden.
7. **Standardisierung**
   Falls eine natürliche Abbildung nicht möglich ist, dann sollte das System soweit standardisiert werden, dass der Benutzer Abläufe nur einmal lernen muss.

6. HCI Muster

Bei Standards und Richtlinien geht es darum, den Designern ein Konzept vorzugeben, das aufgrund von Forschung und Wissen aus z.B. dem Bereich der Psychologie oder Ergonomie hervorgeht. Bei Mustern wiederum geht es darum, dass man sich auf Lösungen bezieht die sich in der Vergangenheit als erfolgreich bewährt haben.
Dabei sind solche Muster nicht als Regeln oder Anweisungen anzusehen, wie genau etwas gestaltet werden soll, sondern eher als gute Beispiele, wobei dem Entwickler bei der Einbindung alle Freiheiten gelassen werden. Begleitet wird ein HCI Muster meist von einer kurzen Erklärung, welche klarmacht, warum sich das Muster in dieser Form bewährt hat und worin dessen Ursprünge liegen.
Außerdem haben HCI Muster den Vorteil, dass sie im Gegensatz zu Richtlinien miteinander in Verbindung stehen. Das erlaubt den Designern sich beim Entwickeln neuer Muster auf bereits vorhandene zu beziehen, was eine fortwährende Verbesserung und Weiterentwicklung zur Folge hat. Ein weiterer Vorteil besteht in ihrer einfachen Verständlichkeit wodurch sie sowohl vom unerfahrenen Webdesigner bis hin zum professionellen Entwicklerteam benutzt werden können. Ihre unkomplizierte Sprache bildet eine Kommunikationsgrundlage für verschiedene Parteien wie z.B. Benutzer, Entwickler, Designer usw..

Um sich dieses Prinzip zu verdeutlichen, betrachte man folgendes Beispiel eines Musters:

Muster: „Ermögliche mehrfache Dokumentablage“

Situation: Dokumente müssen oft in mehreren Ausführungen vorhanden sein und es sollte möglich sein, Referenzen auf diese zu erstellen. Im Büroleben müssen dafür Kopien des Dokumentes erstellt und in die verschiedenen Orte/Kategorien einsortiert werden.

Problem: Das Dokument ist während des Kopiervorgangs nicht verfügbar und kann dabei ggf. sogar verloren gehen.

Lösung: In einem Programm kann dieses Problem einfach gelöst werden, indem man einen „Kopieren“- bzw. „Verknüpfung erstellen“-Button zur Verfügung stellt.

7. Zusammenfassung

Design Regeln im Bereich der HCI bieten Systementwicklern eine hilfreiche Arbeitsbasis und sind vielseitig einsetzbar. Beachtet man diese Regeln konsequent erleichtern sie einem die Erstellung benutzerfreundlicher Systeme.
Dabei kann man sich einerseits an theoretische Richtlinien, oder andererseits auch an praxisnahe Muster halten, wobei beide Methoden ihre Vor- und Nachteile haben.
Da der Bereich der HCI noch relativ neu und unerforscht ist, werden Designregeln auch in Zukunft immer weiterentwickelt und durch die wachsende Priorität von Benutzerfreundlichkeit an Bedeutung gewinnen.

8. Literaturverzeichnis

Literarische Quellen

1. A. Dix, J. Finlay, G. D. Abowd, R. Beale: Human Computer Interaction, Prentice Hall, 3. Auflage, 2003

2. Abras, C., Maloney-Krichmar, D., Preece, J. (2004) User-Centered Design. In Bainbridge, W. Encyclopedia of Human-Computer Interaction. *User-Centered Design.* Thousand Oaks: Sage Publications.

3. Niamh McNamara, Jurek Kirakowski (2006) Functionality, Usability, and User Experience: Three Areas of Concern. ACM

4. Richard N Griffiths, Lyn Pemberton (2005) University of Brighton, Brighton, UK. Don’t Write Guidelines - Write Patterns!