

# Human-Computer-Interaction im Web 2.0

Jochen Puff  
Seminar ViCCC  
17. November 2007

**Zusammenfassung**—Dieses Dokument behandelt diverse Aspekte der Mensch-Maschine-Kommunikation (Human-Computer-Interaction). Es wird auf Regeln, Richtlinien und Normen eingegangen, der Mensch als Akteur betrachtet, Dialoggestaltung im Allgemeinen diskutiert sowie auf Usability-Engineering und Test kurz eingegangen und schließlich eine Verbindung zum Web, speziell Web 2.0, hergestellt.

## I. EINLEITUNG

DIESE Seminararbeit ist im Rahmen des Studienprojekts A „ViCCC“ an der Universität Stuttgart, in Zusammenarbeit mit dem Institut für Visualisierung und Interaktive Systeme, sowie dem Institut für Baubetriebslehre, entstanden.

### A. Zielgruppe

Die Ausführungen richten sich in erster Linie an die Personen, die in direktem Zusammenhang mit dem Studienprojekt A „ViCCC“ stehen. Die durchführenden Personen, die Kunden und die Betreuer.

## II. BEGRIFFSABGRENZUNG

### A. Interaktives System

In der vorliegenden Seminararbeit wird von *interaktiven Systemen* die Rede sein. Dieser Begriff wird seminarspezifisch als Synonym für *Anwendung* verwendet. Allerdings sei an dieser Stelle darauf hingewiesen, dass ein interaktives System im Allgemeinen **keine** Anwendung ist. Eine Anwendung (die nicht vollständig autonom arbeitet) ist immer auch ein interaktives System, jedoch nicht umgekehrt.

Eine genaue Definition eines interaktiven Systems, mit all seinen Eigenheiten und Besonderheiten, kann in [9, Entwicklung interaktiver Systeme] gefunden werden.

### B. Web 2.0

Der Begriff Web 2.0 entzieht sich einer klaren Abgrenzung. Ursprünglich als Marketing-Begriff eingeführt, hat sich dieser Begriff im *Volksmund* für neuartige Technologien im Internet, sowie auch für Interaktionsarten und -ansätze im **WWW** eingebürgert.

1) *Grundprinzip*: Das Grundprinzip des Web 2.0 ist die zwischenmenschliche Interaktion und die direkte Rückkopplung auf Handlungen des Benutzers. Das Prinzip der zentralen Datenverwaltung gilt nicht mehr. Während die Speicherung der Daten weiterhin zentral funktioniert, ist die Manipulation der Daten auf viele Stellen verteilt. Benutzer haben die Möglichkeit auf die Entwicklung eines Web-Portals maßgeblichen Einfluss zu nehmen. Somit geht die Inhaltsverwaltung oft von der vierten Gewalt über zur fünften Gewalt.

2) *Technologische Aspekte*: Obwohl Web 2.0 eigentlich keine technologische Entwicklung beschreibt, werden bestimmte technologische Ansätze in direkter Verbindung mit dem Web 2.0 gesehen.

- **Ajax** bietet beispielsweise die Möglichkeit eine Web-Anwendung, im Normalfall eine Website, wie eine Desktop-Applikation zu handhaben. Darum gelten die im Folgenden genannten Aspekte fast uneingeschränkt auch für das Web 2.0.
- **RSS-Feeds** und deren Generika bieten die Möglichkeit zum schnellen, unkomplizierten und ständigen Datenaustausch zwischen zwei Teilnehmern. Meist ist dieser Austausch auf Basis von **XML**-Formaten aufgebaut.
- **Webservices**, welche für den Datenaustausch von Server und Client zuständig sind und sich auf drei definierte Standards begründen: **UDDI**, **WSDL** und **SOAP**.

Die im Vorigen genannten Technologien sind natürlich nur ein kleiner, wenn auch wichtiger, Ausschnitt aus dem reichhaltigen Angebot an Kommunikations- und Interaktionstechnologien.

3) *Gesellschaftliche Aspekte*: Der Schwerpunkt im Web 2.0 liegt in den gesellschaftlichen Aspekten. Jeder Nutzer einer Web-Applikation soll die Möglichkeit haben sie *intuitiv* und *direkt* zu bedienen, sie von *überall* zu erreichen, sie seinen Wünschen *anzupassen*, seine *Persönlichkeit* auszudrücken, sie mit anderen Mitbenutzern zu *teilen* und sie grundlegend und inhaltlich zu *verändern*, sofern die Rechteverwaltung dies zulässt. Wikis und *Social-Network*-Plattformen stellen beispielsweise diese Punkte sicher.

Auf die gesellschaftlichen Aspekte wird im Rahmen dieser Seminararbeit nicht explizit eingegangen.

### 4) Beispielanwendungen:

- Wikipedia<sup>1</sup>
- YouTube<sup>2</sup>
- Last.fm<sup>3</sup>
- studiVZ<sup>4</sup>

## III. REGELN, RICHTLINIEN UND NORMEN

Die Begriffe *Regeln*, *Richtlinien* und *Normen* bilden eine Hierarchie. An unterster Stelle stehen die Regeln. Sie sind kleine Fragmente, welche sich mit einem ebenfalls kleinen Teil des zu regelnden Bereichs befassen und diesen festlegen. Ein Beispiel für eine Regel wäre z.B. die Einhaltung der

<sup>1</sup><http://de.wikipedia.org>

<sup>2</sup><http://youtube.com>

<sup>3</sup><http://last.fm>

<sup>4</sup><http://studivz.de>

*Konsistenz.* Darauf, und auf weitere Regeln, wird in *Kapitel V-C auf Seite 4* näher eingegangen.

Aus den Regeln setzen sich die Richtlinien zusammen. Richtlinien bilden also ein Konglomerat an elementaren Richtlinienseiten für eine bestimmte Aufgabenstellung. Die Aufgabenstellung als solche ist in ihrer Thematik nicht festgelegt, jedoch begrenzen sich Richtlinien meist auf ein Arbeitsgebiet und wirken nicht themenübergreifend. Einige große Softwarefirmen haben ihre eigenen Richtlinien, auch *Styleguides* genannt. Sie regeln meist das Aussehen, die Anordnung, die Dialoglogik, die Farb-, Schrift- und Iconwahl, die Terminologie und die Arbeitsprozesse.

Die Normen, die den höchsten Platz in der Hierarchie einnehmen, haben den Anspruch, auf allgemeine Art und Weise, geltend für eine bestimmte Branche oder ihren Tätigkeitsbereich zu sein. Normen sind im Allgemeinen nicht sonderlich beliebt, da ihre Formulierung eher nüchtern und anweisend ist. Sie erleichtern das Arbeiten aber ungemein und sorgen für eine gewisse Kompatibilität zwischen den Produkten verschiedener Anbieter. Sie erleichtern somit dem Benutzer der Produkte die Benutzung verschiedener Produkte aus gleichen Bereichen.

#### A. Richtlinien

Zwei bekannte Richtlinien sind die *8 goldenen Regeln* von *Ben Shneiderman* und die *10 Usability-Heuristiken* von *Jakob Nielsen* [4, Grundlagen der Mensch-Computer-Interaktion]. Diese beiden Richtlinien überschneiden sich in sehr vielen Punkten, weshalb hier nur die Punkte selber genannt werden sollen. Eine nähere Betrachtung der Regeln ist in *Kapitel V auf der nächsten Seite* vorzufinden.

##### 1) 8 goldene Regeln nach Shneiderman:

- 1) Strive for consistency
- 2) Enable frequent users to use shortcuts
- 3) Offer informative feedback
- 4) Design dialog to yield closure
- 5) Offer simple error handling
- 6) Permit easy reversal of actions
- 7) Support internal locus of control
- 8) Reduce short-term memory load

##### 2) 10 Usability-Heuristiken nach Nielsen:

- 1) Visibility of system status
- 2) Match between system and the real world
- 3) User control and freedom
- 4) Consistency and standards
- 5) Error prevention
- 6) Recognition rather than recall
- 7) Flexibility and efficiency of use
- 8) Aesthetic and minimalist design
- 9) Help users recognize, diagnose, and recover from errors
- 10) Help and documentation

#### B. Beispiele für Normen

DIN EN ISO 9241

Ergonomie der Mensch-System-Interaktion.

DIN EN ISO 13407

Benutzer-orientierte Gestaltung interaktiver Systeme.

DIN 66234

Bildschirmarbeitsplätze, Grundsätze ergonomischer Dialoggestaltung.

#### IV. MENSCHEN IN DER INTERAKTION

In der *MCI* stellt der Mensch den zentralen Ankerpunkt der Interaktion dar. Er zeichnet sich durch seine Sinne, sein Gedächtnis und seine Rolle in der Gesellschaft aus. Diese Merkmale stellen harte Anforderungen an den Entwurf interaktiver Systeme. Ein System muss sich also den kognitiven Fähigkeiten des Menschen anpassen.

##### A. Wahrnehmung

1) *Visuelle Wahrnehmung:* Die wichtigste Aufnahmequelle des Menschen ist sein Auge. Die Informationsaufnahme wird von verschiedenen Faktoren beeinflusst. Beispielsweise spielt die *Helligkeit* eine wesentliche Rolle. Auch der (*Farb-*) *Kontrast* oder das *Sichtfeld* wirken auf die visuelle Wahrnehmung ein. Eine differenziertere Aufzählung an Faktoren kann in [11, Interaktive Systeme, Kapitel 2.1.1, S. 41] gefunden werden.

Probleme ergeben sich oft bei der Informationsverarbeitung aufgrund zu vieler visueller Reize und zu großer Informationsflut. Da alles, was über das Auge aufgenommen wird, auch verarbeitet werden muss, müssen interaktive Systeme sich an bestimmte Einschränkungen halten. Siehe hierzu auch *Kapitel V-C.9 auf Seite 6*. Auf die wichtigsten visuellen Merkmale, und wie man sie benutzergerecht anspricht, wird in *Kapitel V-C auf Seite 4* eingegangen.

2) *Auditive Wahrnehmung:* Nach der visuellen Wahrnehmung spielt in der *MCI* die auditive Wahrnehmung die größte Rolle. Mit Hilfe von Schall werden Töne erzeugt. Durch Variation der Schwingungen des Schalls können unterschiedlich hohe bzw. tiefe Töne erzeugt werden. Töne, und deren Art, können die Aufmerksamkeit auf bestimmte Systemteile lenken und beispielsweise Dringlichkeit signalisieren. Wie man Töne einsetzt, welche Vor- und Nachteile sie mit sich bringen wird in *Kapitel V-C.3 auf Seite 4* erläutert.

3) *Bewegungs- und Tastsinn:* Interaktion erfolgt über Bewegung auf oder mit Eingabegeräten. Die klassischen Eingabegeräte wie Maus und Tastatur nutzen diese Sinne. Zur Steuerung einer Maus muss die Position im Raum erkannt und genutzt werden, um eine Bewegung bzw. Reaktion des Zeigers zu erzeugen. Tastaturen nutzen Druckpunkte um eine erfolgte Eingabe zu bestätigen, da nicht zwingend alle Tastendrücke eine visuelle Ausgabe erzeugen.

Da diese beiden Sinne im Rahmen dieser Seminararbeit und im Web 2.0 nur eine geringe Rolle spielen, wird auch nicht weiter auf sie eingegangen. Nur wäre eine Aufzählung ohne sie, im Bezug auf interaktive Systeme, unvollständig.

##### B. Gedächtnis

Nachdem die Wahrnehmung im Allgemeinen für die Informationsaufnahme zuständig ist, sorgt das Gedächtnis dafür, dass die Informationen strukturiert, organisiert und reproduziert werden können. Man unterscheidet zwei grundlegende Teile des Gedächtnis:

- Arbeits- oder Kurzzeitgedächtnis und
- Langzeitgedächtnis

Das Arbeitsgedächtnis hat die Aufgabe eine kurzfristige Speicherung von Informationen zu garantieren. Seine Kapazität ist sehr limitiert. Es können 7 +/- 2 [8, Miller 1956] „*Chunks*“, also bedeutende Einträge, gespeichert werden. Es ist daher wichtig, interaktive Systeme so zu gestalten, dass die temporäre Aufnahme von Information an diese Beschränkung angepasst ist. Gestaltungsmerkmale und „Optimierungen“ für das Arbeitsgedächtnis werden in *Kapitel V-C.9 auf Seite 6* gegeben und diskutiert.

Im Gegensatz zum Arbeitsgedächtnis steht das Langzeitgedächtnis. Hier werden verschiedene Arten von Informationen für lange Zeit gespeichert. Die Kapazitätsgrenzen des Langzeitgedächtnis sind nicht bekannt, jedoch sehr groß. Durch mehrfache Anwendung ein und desselben Arbeitsschritts oder durch Einprägen bestimmter Formationen, Zeichen oder Terminologien wandern Informationen vom Kurzzeitgedächtnis ins Langzeitgedächtnis. Das Langzeitgedächtnis wirkt sich vor allem auf die Wahl von Metaphern und Idiomen, näher in *Kapitel VI auf Seite 7* beschrieben, aus.

### C. Individualität

Neben den allgemeinen menschlichen Faktoren spielen die individuellen Faktoren eine erhebliche Rolle in der Erstellung interaktiver Systeme. Individuen unterscheiden sich in ihrem Wesen, ihrem Verhalten, ihren Fertigkeiten, ihrem Wissen und ihren Kompetenzen.

1) *Fertigkeiten*: Fertigkeitserwerb und -anwendung gliedern sich in 3 weitestgehend autonome Teile nach [11, Stary 1994]:

- Die kognitive Phase
- Die assoziative Phase
- Die automatisierte Phase

Die *kognitive Phase* beschreibt den Wissenszuwachs von Fakten aus dem aktuellen Problem- bzw. Arbeitsbereich. Die Informationen sind hier starr vorgegeben und werden von den Nutzern eines interaktiven Systems im Normalfall mitgebracht. Wenn es sich um Lernsoftware handelt, so wird zumindest ein gewisser Grundstock an Fachwissen vom Benutzer mitgebracht, den das interaktive System aufgreifen, aufbereiten und zur weiteren Verständnisklärung nutzen kann.

In der *assoziativen Phase* wird das in der kognitiven Phase erworbene Wissen reproduziert und zur Problemlösung angewandt. Gerade in dieser Phase stellen sich primäre Aufgaben an ein interaktives System. Die Fehlerquote in dieser Phase ist verhältnismäßig hoch. Es muss gewährleistet werden, dass die Anwendung des erworbenen Wissens intuitiv und selbst erklärend erfolgen kann.

Routinetätigkeiten werden in der *automatisierten Phase* durchlaufen. Sie bilden den Hauptanteil der Tätigkeiten im „Tagesgeschäft“. Beispielsweise das Anlegen und Bearbeiten von Mitarbeitereinträgen in einer Personalabteilung. Die Fehlerquote in dieser Phase ist gering. Dafür stellen sich hier andere Anforderungen an das interaktive System, wie die komfortable und vor allem **schnelle** Erledigung der Aufgaben.

2) *Persona*: Das Konzept der *Persona* wurde 1998 von Alan Cooper [2, *The Inmates Are Running the Asylum*] vorgestellt. Eine *Persona* ist ein fiktiver Mensch, der die oben erwähnten individuellen Eigenschaften eines bestimmten Anwendertyps besitzt.

Personas werden benutzt um Fragen der Bedienbarkeit zu klären:

- „*Wie würde die fiktive Person X in dieser Situation reagieren?*“
- „*Warum hat Person X hier Vorteile gegenüber Person Y und wie kann man den Unterschied so gering wie möglich halten, ohne eine der Personen zusätzlich zu belasten?*“

Personas sind ein tragendes Element der Konzipierung von Benutzerschnittstellen, da man die Anforderungen einer Anwendergruppe so genau wie möglich abbilden muss.

Die einzigen Kosten, die Personas erzeugen, sind die Erstellungs- bzw. Findungskosten. Durch eine ausreichende Analyse des Kunden, seiner Prozesse und Angestellten, kurz der späteren Benutzer des Systems, können die Personas verhältnismäßig schnell gefunden werden. Der Nutzen übersteigt die Kosten bei weitem.

a) *Beispiel*: Die Sachbearbeiterin Schlibrovski, 39, in der Schadensabteilung einer Versicherung mit über zehn Jahren Berufserfahrung. Sie kennt ihre Kunden, die Abläufe und alle typischen Daten. Sie arbeitet nicht ungern am Computer, wenn der funktioniert. Daten, die auf dem offiziellen Wege nicht zugänglich, aber notwendig sind, beschafft sie sich aufgrund ihrer guten Kontakte.

Beispiel aus [4, *Grundlagen der Mensch-Computer-Interaktion*, Kapitel 14.1, S. 317].

## V. DIALOGE

### A. Definition

Ein Dialog definiert sich nicht ganz so offensichtlich wie im Allgemeinen vielleicht angenommen. Dialoge von interaktiven Systemen lassen sich am Besten im Vergleich mit den Dialogen zwischen zwei Menschen erklären. Es gibt einen *aktiven* und einen *passiven* Teilnehmer. Diese Rollen können tauschen, wenn der aktive Teilnehmer seine Interaktion, also seinen Satz, sein Thema, vorerst zu einem Abschluss gebracht hat. Der passive Teilnehmer muss dem aktiven Teilnehmer signalisieren, dass seine Ausführungen oder Interaktionen angekommen sind und darüber nachgedacht wird. Geschieht das nicht, verliert der aktive Teilnehmer die Motivation zur Dialogführung.

Übertragen wir dieses Konzept nun auf die **MCI**. Es gibt zwei Teilnehmer: Den *Benutzer* und das *System*. Der Benutzer (aktiv) agiert, aus eigener Intention, um das System (passiv) zu einer Handlung zu bewegen. Er erwartet eine Rückmeldung, eine Bestätigung. Andererseits kann das System (nun aktiv) auch eigenmächtig eine Frage an den Benutzer (jetzt passiv) stellen, um beispielsweise einen Bearbeitungsschritt auszuführen, der zusätzliche Information benötigt.

Ein Dialog ist also auch die *Kommunikation* zwischen System und Benutzer. Dabei sei erst einmal egal, von welcher Seite der Anstoß kam. Ein Dialog besteht aus *Interaktionselementen*, ihrer *Repräsentation*, ihrem *Verhalten* und ihrer

*Fähigkeit sich mitzuteilen.* Diese, und noch einige weitere, Eigenschaften werden oft als *Look & Feel* bezeichnet.

Es soll noch angemerkt sein, dass es eine kleine Abweichung zum Mensch-Mensch-Dialog gibt. Es ist beim Dialog eines interaktiven Systems **immer** Aufgabe des Systems dem Anwender *Rückmeldung* zu geben, sowie klar zu zeigen, was zu tun ist, **niemals** anders herum.

## B. Schnittstellen

Prinzipiell besteht eine Anwendung, im Bezug auf Benutzungsschnittstellen, aus drei Schichten. Sie überlagern sich zum Teil. Siehe dazu auch [5, Das IFIP-Modell für Benutzungsschnittstellen]. Diese Schichten sind die *Ein- und Ausgabeschnittstelle*, die *Dialogschnittstelle* und die *Werkzeugschnittstelle*. Im Folgenden wird nur auf die Dialogschnittstelle näher eingegangen.

An dieser Stelle sei noch einmal erwähnt, dass ein Dialog nicht nur ein Grafik-Widget aus einer Grafikkbibliothek (bspw. Java Swing, Trolltech Qt) ist. Bezüglich dieser Differenzierung sei auf *Kapitel V-A auf der vorherigen Seite* verwiesen.

## C. Die Dialogschnittstelle

Wie in den vorangegangenen Kapiteln mehrfach erwähnt, erfolgt nun eine Zusammenfassung grundlegender Regeln zur Gestaltung von Benutzerschnittstellen. Die jeweiligen Regeln sind zum Teil mit Beispielen versehen. Diese stellen jedoch kein „must-do“ dar, sondern sollen lediglich aufzeigen, in welcher Art man die Regeln umsetzen *könnte*.

Inhaltlich folgen die Punkte der *DIN 66234 Teil 8*, sowie den in *Kapitel III auf Seite 1* angesprochenen Richtlinien von *Ben Shneiderman* und *Jakob Nielsen*.

1) *Konsistenz*: Dialoge sollten sich systemweit konsistent verhalten. Bedienelemente, oder Elemente allgemein, sind einheitlich zu gestalten, wenn sie von der gleichen Art sind und/oder der gleichen Aufgabe dienen. Einheitliche Gestaltung bezieht sich hier auf:

- Farbe
- Größe
- Beschriftung
- Terminologie
- Aufgabe
- Interaktionsmöglichkeit
- Verhalten
- Reihenfolge
- Position

Die Konsistenz bezieht sich auch auf die Abfolge, in der manche Tätigkeiten abgehandelt werden. Diese Reihenfolge darf sich nicht von einem Dialog zum anderen unterscheiden. Benutzer greifen auf erlerntes Wissen zurück und sind oft hilflos, wenn sich ihr bisheriger Arbeitsprozess eines Problems nicht auf das gleiche Problem, an anderer Stelle, anwenden lässt.

**Generell gilt:** Sorge dafür, dass Gleiches Gleiches tut und sich gleich verhält.

2) *Unterschiedliche Ansprüche*: Verschieden erfahrene Benutzer haben verschiedene Bedienansprüche. Die Erfahrung mit dem momentanen Arbeitsprogramm, wie auch mit anderen Programmen allgemein, spielt hier eine Rolle. Der unerfahrene Benutzer erwartet klar strukturierte Menüs und führt (fast) alle Interaktion mit intuitiven Bediengeräten wie der Maus durch. Der erfahrene Benutzer möchte schneller arbeiten. Anstatt immer wieder durch Menüeinträge zu klicken erwartet er Tastaturkürzel, Mnemonics und Makros.

**Generell gilt:** Biete alle Aktionen in Menüs an. Führe die Möglichkeit ein, häufig benutzte Aktionen stark abzukürzen.

3) *Rückkopplung*: Ein System muss dem Benutzer deutlich machen, dass eine Aktion, die er angestoßen hat, auch wirklich beim System angekommen ist. Gib es keine Rückmeldung, wird der Benutzer dazu verleitet, Aktionen sofort zu wiederholen, auch wenn im Hintergrund bereits die vorherige Aktion bearbeitet wird.

Wie in *Kapitel IV-A auf Seite 2* bereits erwähnt, gibt es visuelle wie auch auditive Wahrnehmung. Dies sind die beiden wichtigsten Reizsysteme für Rückmeldungen.

a) *Visuelle Rückkopplung*: In den meisten Fällen interagiert der Benutzer mit Bedienelementen. Ein Button, der gedrückt wird, bekommt häufig durch Schattierung einen dreidimensionalen „Eindrückeffekt“. Wird der Aktionsfokus in eine Textarea oder ein Textfield gelegt, so kann beispielsweise ein Rahmen um das Eingabeelement erscheinen. Weitere Beispiele sind das semi-transparente Mitführen von Icons beim *Drag & Drop* oder die Anzeige, wie lang eine Aktion dauert. Kopier-, Synchronisations- und sonstige Aktionen, die nicht in „einem Schritt“ erledigt sind, erfordern zwingend eine Fortschrittsanzeige. Sei es in Form einer Prozentanzeige oder einfach nur eines animierten Rades, welches Berechnung symbolisiert.

b) *Auditive Rückkopplung*: Die auditive Wahrnehmung sollte in der Rückkopplung eine eher untergeordnete Position einnehmen. Rückmeldungen auf kritische Aktionen, wie das Löschen von Datensätzen oder eine andere irreversible Aktion, können sowohl visuell als auch akustisch untermalt werden. Jedoch ist darauf zu achten, dass nicht jede kritische Aktion den gleichen Ton produziert. Ein Warnhinweis darf nicht gleich wie ein kritischer Fehler klingen.

c) *Wie reagiert das Web 2.0?:* Systeme, die eine direkte Rückkopplung auf Aktionen geben, nennt man *direkt manipulative Systeme*. Im Vergleich zum Web 1.0, welches erst nach dem Neuladen in einer Webseite auf eine Aktion reagieren kann, bietet das Web 2.0 die Möglichkeit direkt zu reagieren. Dies kann beispielsweise durch *Ajax* erreicht werden, womit im Hintergrund Informationen aus einer Datenquelle „live“ nachgeladen werden.

**Generell gilt:** Erzeuge *passende* Rückmeldungen auf alle Aktionen, bei denen der Benutzer mit dem System interagiert.

4) *Abgeschlossenheit und Simplizität*: Die wesentliche Aufgabe von interaktiven Systemen liegt darin, Aufgaben zu bewältigen, oder wenigstens zu erleichtern. Um dies zu garantieren, müssen Aktionen, die zusammen einen geschlossenen Arbeitsablauf bilden, auch als solcher zu erkennen sein. Ein Beispiel aus dem Alltag wäre das Konfigurieren der Ausstattung eines Autos. Es muss die Innenausstattung, die Reifen,

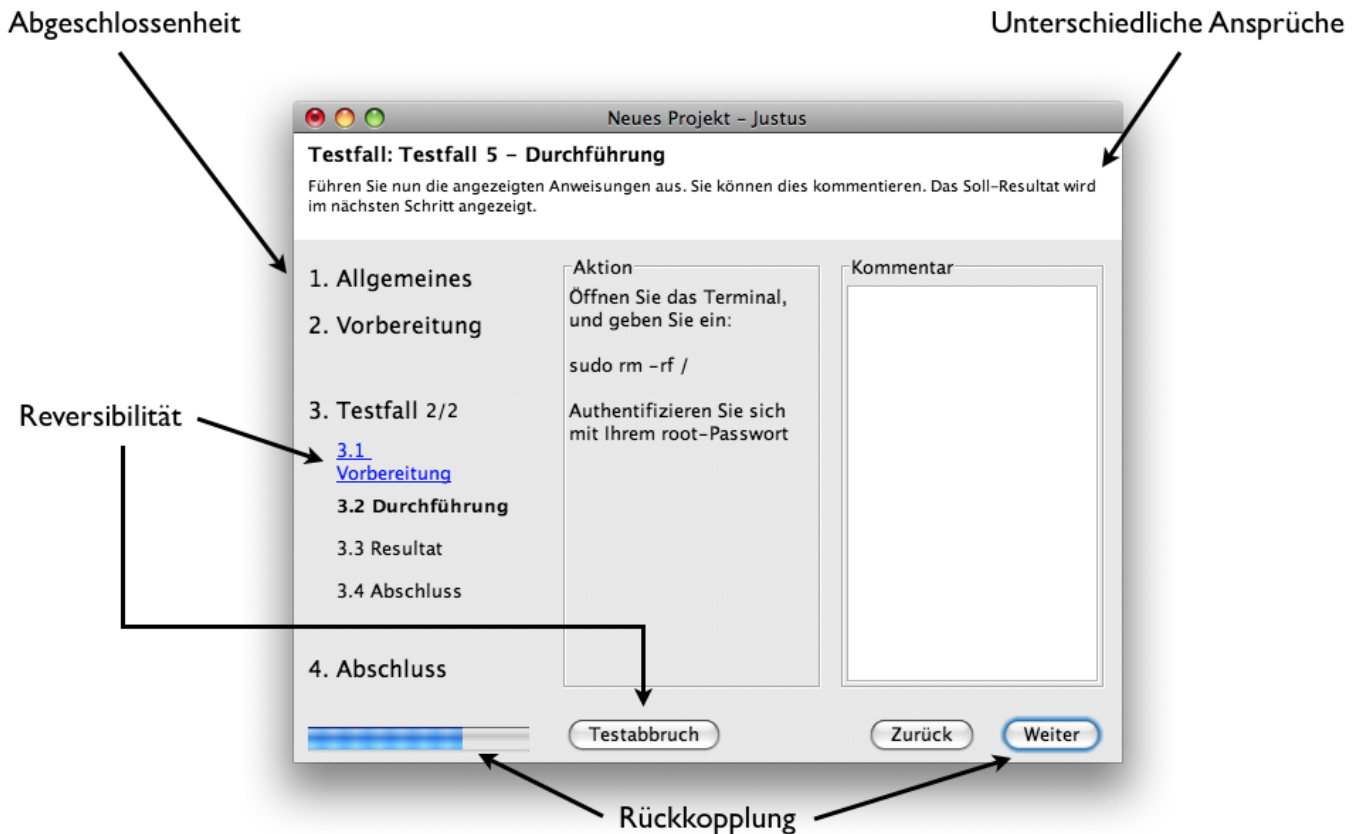


Fig. 1. Beispiel guter Dialoggestaltung  
Quelle: Justus<sup>a</sup>

<sup>a</sup><http://justus.tigris.org>

die Felgen, die Motorisierung und was an einem Auto sonst noch konfigurierbar ist, konfiguriert werden.

Es muss klar zu erkennen sein, welcher Konfigurationsschritt auf den Nächsten folgt und wann ein Ende zu erwarten ist. Wichtig dabei ist auch, dass die Abfolge natürlich bzw. domänenangemessen geschieht. Ein Programm zur sequentiellen Abarbeitung einer Norm hat dies auch in der angegebenen Reihenfolge zu erledigen.

a) *Wie reagiert das Web 2.0?*: Geschlossene Arbeitsabläufe können im Web 2.0, wie auch im Web 1.0, ohne Weiteres umgesetzt werden. Jedoch bietet sich im Web 2.0 die Möglichkeit, flüssiger zu arbeiten. Es muss zwischen Bearbeitungsschritten nicht auf Antworten vom Server gewartet werden. Eine „Live-Evaluierung“ ist möglich. Dies erleichtert unter anderem Fortschrittsanzeigen und ermöglicht ein zusammenhängendes Bearbeiten eines gruppierten Arbeitsablaufes.

**Generell gilt:** Gruppiere *zusammengehörige* Schritte in abgeschlossenen Arbeitsabläufen.

5) *Fehlermeldungen*: Wann immer Fehler auftreten, müssen diese an den Benutzer weitergegeben werden, damit dieser **entsprechend** reagieren kann. Offensichtlich scheint, dass eine Fehlermeldung daher den Fehler genau beschreiben muss, den Auftrittsort approximieren und, wenn möglich, sogar Lösungsvorschläge anbieten sollte. **Niemals**, oder nur in eindeutigen und sehr gut begründeten Ausnahmefällen, aber eigenmächtig

korrigieren sollte.

**Generell gilt:** Erzeuge *konstruktive* Fehlermeldungen für *alle* auftretenden Fehler.

6) *Fehlervermeidung*: Es sollte erst gar nicht zu Fehlern kommen. Der Benutzer sollte so gut „gepolstert“ wie möglich durch das Programm geführt werden. Die klassischen, auch in jeder Literatur zur **MCI** zu findenden, Beispiele dafür sind:

- Ein Datum nicht eingeben lassen, sondern mit Auswahlboxen den Tag, den Monat und das Jahr auswählen lassen. Sind solche Auswahlboxen nicht erwünscht oder möglich, hat zumindest das anzugebende Format in dem entsprechenden Eingabefeld zu stehen.
- Das „Ausgrauen“ von momentan nicht anwendbaren Menübefehlen.

**Generell gilt:** Vermeide Fehler durch angegebene Formate oder vordefinierte Auswahlmöglichkeiten.

7) *Reversibilität*: Benutzer machen Fehler. Völlig unabhängig davon wie erfahren sie sind. Auch wenn kein Fehler vorliegt, so muss die gemachte Eingabe oder Interaktion nicht zwingend das gewesen sein, was der Benutzer gewollt hat.

Betrachtet man das Ausprobieren einer unbekanntem Anwendung oder Systems, so stellt man fest, dass die meisten Eingaben und Interaktionen am Anfang dazu dienen Möglichkeiten auszuloten. Dieser Sachverhalt wirkt sich nicht nur auf das Einarbeiten in ein unbekanntes System aus. Auch

Aktionen, die selten von Benutzern durchgeführt werden, produzieren oft unerwartete Ergebnisse. Sie sollten deshalb keine endgültigen Resultate bringen.

Üblich ist ein *Undo-Redo*-Mechanismus, um unerwartete Eingaben und Interaktionen rückgängig zu machen und wieder herzustellen. Die Zahl der zu revidierenden/herzustellenden Aktionen sollte, wenn möglich, unbeschränkt sein (natürlich sind die Speichergrenzen zu beachten). Auch sollten angestoßene Aktionen, die eine Folge von Interaktionen vom Benutzer erfordern, oder eine lange Zeit in Anspruch nehmen, abgebrochen werden können.

**Generell gilt:** Biete die Möglichkeit, Aktionen rückgängig zu machen und Prozesse abubrechen.

8) *Benutzerzentrierung:* Wie bereits in der Einleitung zu diesem Kapitel erwähnt, unterscheidet man passive und aktive Teilnehmer. Präziser auf die **MCI** abgestimmt kann von system-, benutzergesteuerten sowie von hybriden Dialogen gesprochen werden. In den meisten Anwendungen und Systemen findet man hybride Dialoge, die als benutzergesteuerte Dialoge „getarnt“ sind, vor.

Ein hybrider Dialog wechselt die aktiven und passiven Rollen durch. Einmal ist der Benutzer aktiv, dann wieder das System. Im Allgemeinen wird versucht dem Benutzer die Entscheidungen, soweit möglich, zu überlassen. Ist dies nicht möglich, so sollte versucht werden, den Nutzer mit einzubeziehen und/oder ihn glauben zu lassen, die Kontrolle zu haben. Kein Mensch, und das liegt in seiner Natur begründet, lässt sich gern den Wind aus den Segeln nehmen. Er gibt gerne Anweisungen, lässt sich aber wenig sagen.

**Generell gilt:** Gib dem Benutzer das Gefühl, der Mittelpunkt des Systems zu sein.

9) *Mentalbelastung:* Im *Kapitel IV-B auf Seite 2* wurde die nur sehr beschränkte Aufnahmefähigkeit des menschlichen Kurzzeitgedächtnisses erwähnt. Eine unmittelbare Folge ist, Aktionen und Aktionsgruppen so simpel wie möglich zu halten. Unmengen an Einstellungsmöglichkeiten und Informationen für eine Aufgabe bereitzustellen ist eher hinderlich als förderlich. Es ist offensichtlich, dass hier ein Kompromiss zwischen der Gruppierung von Aktionen zu Abläufen und dem Informationsangebot gefunden werden muss.

Das klassische Beispiel dafür sind Menüs, die eher breit als tief sein sollten. Obwohl sich hier die Präferenzen von erfahrenen und unerfahrenen Benutzern unterscheiden, könnte man folgende Werte verwenden: *Nicht mehr als 5 bis 7 Menüpunkte auf einer Ebene und nicht mehr als 2 bis 3 Menüebenen in der Tiefe.*

**Generell gilt:** Überfordere den Benutzer nicht mit unnötigen Informationen und stelle die Aufgabe in den Vordergrund.

10) *Fachorientierung:* Anwendungen oder Systeme dienen dazu, Aufgaben zu erledigen. Diese Aufgaben kommen meist aus Fachbereichen, aus denen der Entwickler keine Erfahrungswerte schöpfen kann. Trotzdem muss die Anwendung oder das System an die eventuell standardisierten Arbeitsabläufe dieser Branche angepasst sein. Auch die verwendete Terminologie, ja sogar das Layout von Eingabemasken, hat der Fachorientierung zu folgen.

**Generell gilt:** Drücke dich wie der Benutzer aus und unterstütze seinen Fachbereich.

11) *Psychologie:* Über die im Vorhergehenden allgemein gültigen Regeln hinaus gibt es weniger greif- bzw. messbare Anforderungen.

Die *Zufriedenheit* eines Benutzers ist ein maßgeblicher Faktor dafür, wie produktiv mit einem System gearbeitet werden kann. Besteht schon vor dem Beginn der Arbeit ein Unbehagen dem System gegenüber, so wird die Arbeit zum Frust beim Benutzer führen und die Produktivität erheblich senken. Ausschlaggebend für eine Benutzung, die Freude bereitet, ist beispielsweise das Aussehen einer Anwendung. Genügt diese höheren ästhetischen Ansprüchen, steigert das erheblich die Arbeitslust.

#### D. Hilfesysteme

Auch wenn alle in *Kapitel V-C auf Seite 4* genannten Regeln befolgt werden, so wird es immer dazu kommen, dass ein Benutzer nicht weiter weiss. In diesem Fall muss ein *Hilfesystem* eingreifen. Ein Hilfesystem dient dazu, einen Systemzustand (nicht zwangsweise den aktuellen) zu erklären und die Entscheidungsmöglichkeiten (sowie deren Folgen) aufzuzeigen. Für Hilfesysteme gelten im Allgemeinen die gleichen Gestaltungsgrundsätze wie für jedes andere interaktive System.

Eine erste Unterteilung von Hilfesystemen kann vorgenommen werden, indem die Initiative betrachtet wird. Liegt die Initiative beim Benutzer, so spricht man von einem *passiven* Hilfesystem. Ansonsten spricht man von einem *aktiven* Hilfesystem.

1) *Passive Hilfesysteme:* Praktisch jedes zur Zeit vorhandene Hilfesystem ist passiv. Gründe dafür sind in erster Linie die Erst- und Wartbarkeit. Wenn hier von passiven Hilfesystemen gesprochen wird sei erwähnt, dass kein klassisches Handbuch in dem Sinne gemeint ist. Es wird nur die digitale bzw. virtuelle Hilfe angesprochen. Für physisch vorhandene Handbücher gelten andere Gestaltungsgrundsätze, auf die hier nicht eingegangen wird.

Passive Hilfesysteme werden vom Benutzer angefordert. Eine systemweit definierte Aktion, beispielsweise das Drücken der Taste *Hilfe* beim Apple Macintosh, oder ein im Dialog verankerter *Button* mit einem Fragezeichen als Piktogramm, dient zum Aufruf des Hilfesystems. Das Hilfesystem selbst besteht meist aus einer Fülle an Informationen, die durch eine Struktur angeordnet und meist auch miteinander verbunden sind. Um diese Informationskette zu erzeugen wird ein Einstiegspunkt benötigt. Dieser kann entweder eine generische „Startseite“ sein oder sich *kontextabhängig*, durch den Ort oder Zeitpunkt des Aufrufs, definieren.

Trotz der vorhandenen Strukturierung muss gewährleistet sein, dass man sich in der Informationsflut passiver Hilfesysteme zurechtfindet. Ein viel gewählter Ansatz dafür ist die aus dem **WWW** bekannte Technik des Hypertexts. Durch Querverbindungen zwischen den Informationen kann auf eventuell inhaltlich ähnliche Teile der Hilfe verwiesen werden. Diese Art der Zugänglichkeit reicht aber nicht aus. Eine Suche erleichtert das Auffinden von Informationen erheblich. Anstatt sich durch Unmengen an Informationen zu wühlen, kann gezielt nach Problemen gesucht werden. Auch der Einsatz natürlicher Sprache hilft bei der Informationsfindung ungemein,

da Informationen besser aufgenommen werden können und intuitiver gesucht werden kann.

2) *Aktive Hilfesysteme*: Den Gegensatz bilden aktive Hilfesysteme. Sie agieren mehr oder minder autonom. Um überhaupt wissen zu können ob ein Benutzer Hilfe braucht, muss das Hilfesystem die Benutzeraktionen verfolgen, auftretende Ereignisse und Reaktionen darauf betrachten und analysieren sowie die Situation, in der der Benutzer sein könnte, richtig einschätzen.

Es ist in vielen Fällen unmöglich, wirklich vorhersagen zu können, aus welchen Gründen ein Fehler aufgetreten ist und wie bzw. ob die Hilfe nun agieren soll. Prinzipiell können die aktiven Systeme nur zuverlässig auf syntaktische und lexikalische Eingabefehler reagieren. Dem System ist bekannt, in welchem Zustand es sich befindet und welche Eingaben zulässig sind. Pragmatische und semantische Fehler können schlechter behandelt werden, da sie Informationen über die Intention des Benutzers benötigen. Pragmatische Fehler wären beispielsweise die unnötig lange Aneinanderkettung von Aktionen um ein Ziel zu erreichen, das auch leichter erreicht werden kann. Ob der Benutzer aber Gründe für diese lange Aktionskette hatte (Intention), kann dem System nicht bekannt sein.

Aus genannten Eigenschaften ergibt sich offensichtlich, dass aktive Systeme eine ganze Reihe an Heuristiken und Annahmen über den Benutzer treffen müssen. Dies ist meist nicht zuverlässig möglich. Darum sind aktive Hilfesysteme eher die Ausnahme als die Regel.

## VI. METAPHORIK UND IDIOME

Eine Anforderung an ein System ist meist, die reale Welt so genau (oder idealisiert abstrahiert) wie möglich abzubilden. Diese Abbildung kann in verschiedener Weise geschehen. Eine rein faktbezogene Abbildung (bspw. für Ingenieure) nennt man *implementierungsbezogene Schnittstelle*. Diese Schnittstellenart verbindet ihre Bedienung mit ihrer technischen Umsetzung. So kann durch die Benutzung der Schnittstelle in machen Fällen auf eine bestimmte „Berechnung“ geschlossen werden. Ein Beispiel dafür wäre eine Schnittstelle zur Interpolation. Anhand der Parameter und Werte die man angeben muss, sowie der Ausgabe, kann man auf das Interpolationsverfahren schließen. Diese Benutzerschnittstellen benötigen Hintergrundwissen zur effektiven und effizienten Bedienung.

Zwei weitere Verfahren der Benutzerschnittstellengestaltung, die auf Schlussfolgerungen und der Erinnerung basieren, werden im Folgenden vorgestellt.

### A. Metaphern

Der Mensch sieht sich im Leben ständig mit Metaphern konfrontiert. Man unterscheidet zwischen diversen Arten von Metaphern. In Bezug auf die Benutzerschnittstellengestaltung wird hier nur die visuelle Metapher diskutiert.

Metaphern basieren auf Intuition, also der schnellen Erkennung und Verarbeitung von Informationen, gefolgt von Assoziationen mit Bekanntem. Hieraus ergibt sich das erste Problem mit Metaphern. Derjenige, der Metaphern verwendet (Benutzerschnittstelle/Schnittstellendesigner), geht davon aus,

dass sein Interaktionspartner (Benutzer) ein gewisses Vorwissen mit sich bringt. Oft, aber mit Sicherheit nicht immer, ist dieses Vorwissen vorhanden.

Metaphern in der Benutzerschnittstellengestaltung arbeiten meist mit visuellen Reizen. Sehen wir das Bild eines Einkaufswagens (Piktogramm) in einem Online-Shop, verbinden wir das oft mit dem Einkaufen im Supermarkt. Man kann den Wagen mit Produkten füllen, geht dann zur Kasse und bezahlt. Ein ganz normaler und alltäglicher Ablauf.

Wirklich? **Nein.**

Metaphern sind nicht allgemein gültig. Das Vorwissen, dass ein Einkaufswagen sich mit Produkten füllen lässt, ist zwar bei den meisten Menschen vorhanden, aber sicher nicht bei allen. Ein Mensch, der sein Leben lang nur auf Bazaren eingekauft hat, wird das Konzept des Einkaufswagens nicht kennen. Er kann also auch mit dem Piktogramm keine Assoziation finden. Metaphern unterstützen den wissenden Benutzer stark, aber lassen den unwissenden Benutzer komplett im Dunkeln tapen.

Metaphern sind also oft unpassend und missachten wissend die Herkunft, den Kulturkreis und das Wissen von Benutzern. Sie sollten deshalb mit Vorsicht eingesetzt werden, auch wenn sie als das beliebteste Mittel von Benutzerschnittstellendesignern gelten. Auch die Mehrdeutigkeit ist ein Problem. Das Piktogramm einer Musiknote kann die Änderung der Tonlautstärke oder die Wahl eines Musikstücks (und noch viele weitere Dinge) bedeuten.

Neben diesen *lokalen* Metaphern gibt es noch *globale Metaphern*. Eine globale Metapher beeinflusst die komplette Arbeitsweise mit einem System. Das wohl beste Beispiel dafür ist die *Desktop-Metapher*. Fast jedes grafisch orientierte Betriebssystem benutzt den Desktop. Der Benutzer assoziiert damit das Platzieren und Verschieben von Dokumenten. Das Anlegen von Strukturen und Ordern, in denen die Dokumente abgelegt werden können. Wird eine solche globale Metapher, aus beispielsweise vorangehend genannten Gründen, nicht verstanden, ist die Benutzung des gesamten Systems, und nicht nur eines lokalen Bereichs, gefährdet.

**Generell gilt:** Metaphern können unterstützen, gefährden aber die grenzüberschreitende Benutzung von Systemen in unterschiedlichen Bevölkerungsgruppen.

### B. Idiome

Idiome stellen, im Gegensatz zu Metaphern, Erfahrungswerte dar, die auf einem Lerneffekt basieren. Auch auf instinktivem Verhalten lassen sich Idiome aufbauen. Der Zusammenhang zwischen Intuition und Instinkt sollte an dieser Stelle klar sein und wird als bekannt angenommen.

Aktionen, die Menschen ausführen, bleiben in Erinnerung. Diese Erinnerung wird genutzt, um weitere Aktionen zu steuern und zu interpretieren. Der Vorteil gegenüber Metaphern ist, dass man keine Verbindung zwischen realer und abgebildeter Welt herstellen muss. Die Schnittstellen-Idiome arbeiten grundsätzlich nur in der abgebildeten Welt. Es wird versucht, den Benutzer an eine Konvention heranzuführen und diese dann im weiteren Verlauf der Benutzung wieder aufzugreifen. Gute Idiome müssen nur einmal erlernt werden und bleiben

dem Benutzer ein Leben lang (im Idealfall), dank der großen Kapazität des Langzeitgedächtnis, in Erinnerung.

Ein Beispiel für ein Idiom ist das Schachteln von Ordnern. Mit Ordnern sind hier die aus der *Desktop*-Metapher bekannten Ordner gemeint. Es ist nicht möglich, Ordner im richtigen Leben zu schachteln, darum besteht hier auch kein metaphorischer Zusammenhang. Hat man jedoch einmal erlernt, dass in der abgebildeten Welt das Schachteln von Ordnern zulässig ist, so wird man dies ständig anwenden. Ein gutes Idiom.

An dieser Stelle zeigt sich auch die Abgrenzung zwischen Metapher und Idiom, die nicht immer intuitiv klar ist. Der *Desktop*, im Beispiel mit dem Schachteln von Ordnern, ist eine Metapher. Ein Teil der Metapher, nämlich das Schachteln selber, ist ein Idiom. Metaphern und Idiome können also nahtlos ineinander greifen.

**Generell gilt:** Alle Idiome müssen erlernt werden, gute aber nur ein Mal.

## VII. USABILITY-ENGINEERING

Das *Usability-Engineering* ist ein Teilgebiet des Software-Engineering. Somit unterliegt auch das Usability-Engineering den gleichen Grundsätzen und ähnlichen Vorgehensweisen wie das Software-Engineering. Die Entwicklung von Benutzerschnittstellen orientiert sich an Prozess- und Vorgehensmodellen.

Da der Schwerpunkt des Seminars nicht auf den Engineering-Prozessen liegt, soll hier nur kurz vorgestellt werden, was die drei gebräuchlichsten Modelle des Usability-Engineering sind. Verwiesen sei bei diesen Entwicklungsprozessen auch auf die Personas in *Kapitel IV-C.2 auf Seite 3*, da sie der Dreh- und Angelpunkt des Benutzerschnittstellendesigns sind.

### A. Quantity- und Quality-Engineering

Usability-Engineering lässt sich auf zwei Ästen sehen. Zum Einen dem *Quality-Engineering*, welches sich damit beschäftigt die nicht messbaren Anforderungen an eine Software zu erfüllen. Darunter fällt beispielsweise die Benutzerzufriedenheit. Das *Quantity-Engineering*, zum Anderen, befasst sich mit den messbaren, durch Metriken erfassbaren, Anforderungen an ein System, wie zum Beispiel noch zumutbare Antwortzeiten.

### B. Vorgehensmodelle

1) *Wasserfall-Modell:* Das *Wasserfall-Modell* [10, Royce 1970] beschreibt den (teilweise) rückgekoppelten, strukturierten Ablauf von Projektphasen. Durch seine Simplizität und weite Verbreitung findet es heutzutage noch große Verwendung, obwohl seine Korrektheit in Fachkreisen immer wieder scharf diskutiert wird.

Eine genaue Auflistung der Phasen ist nicht möglich, da das Wasserfall-Modell nur eine grobe Struktur, aber keine genaue Phasenabfolge angibt. Ein Beispiel für das Modell kann in [4, Grundlagen der Mensch-Computer-Interaktion, Kapitel 14.1.1, S. 311] gefunden werden.

2) *Spiral-Modell:* Das *Spiral-Modell* [1, Boehm 1988] definiert sich als iterativer Prozess, der immer wieder 4 Phasen durchläuft:

- 1) Zielfestlegung (Analyse)
- 2) Alternativenauswahl und Evaluierung (Entwurf)
- 3) Realisierung (Implementierung)
- 4) Test und Planung der nächsten Iteration (Abschluss)

Vorteil dieses Modells ist, dass ein Kunde oder auch Testbenutzer immer wieder Rückmeldung über die Entwicklung des System bekommt. Dadurch werden Fehler schneller erkannt und behoben. Eine genauere Beschreibung des Modells kann in einschlägiger Literatur gefunden werden.

3) *Prototyping:* Eng verwandt mit dem Spiral-Modell ist das *Prototyping*. Nach jedem Iterationsschritt im Spiral-Modell entsteht ein Prototyp. Dieser reicht von einer groben Beschreibung von Abläufen bis hin zu deren konkreter Implementierung. Im Bezug auf Benutzerschnittstellen findet man in Prototypen auch oft sogenannte *Mock-Ups*, also Schnittstellen ohne Funktion dahinter.

Auch hier finden sich zahlreiche Anwendungs- und Fallbeispiele in der einschlägigen Literatur.

## VIII. USABILITY-TEST

Das Wort *Engineering* lässt vermuten, dass sich Ergebnisse gut messen lassen. Wenn dies zwar auf die klassischen Ingenieursdisziplinen zutrifft, so aber nicht auf das Software-Engineering und noch viel weniger auf das Usability-Engineering. Aufgrund dieser Tatsache haben sich im Laufe der Zeit unterschiedliche Kategorien [4, Grundlagen der Mensch-Computer-Interaktion, Kapitel 14.5, S. 318] herauskristallisiert.

### A. Testansätze

Die folgende Auflistung an Testverfahren soll einen Überblick geben, welche Tests wann sinnvoll sind. Eine umfassende Analyse von Testverfahren, deren Auswertung und ihrem Nutzen würde den Rahmen des Seminars sprengen. An dieser Stelle sei auch auf *Kapitel VII-A* verwiesen.

1) *Benutzertest:* Diese Testkategorie verfolgt das Ziel, so gut wie möglich auf die tatsächlichen Alltagsanforderungen an die Software zu testen. Grundvoraussetzung für den Test sind Benutzer, die später auch wirklich mit dieser speziellen Software arbeiten werden, sowie die fertige Software, oder ein reifer Prototyp. Ein realer, nicht simulierter Benutzer findet Fehler in Arbeitsabläufen, Inkonsistenzen in der Terminologie oder einfach unhandliche und unintuitive Bedienung. Der Schwerpunkt des Benutzertests liegt allerdings bei der Simulation von alltäglichen Problemen wie Ablenkung und Fehlinformation. Keine Persona kann diese Testqualitäten erfüllen.

2) *Expertentest:* Im Gegensatz zum Benutzertest werden im *Expertentest* erfahrene Benutzer als Tester eingesetzt. Diese reagieren weniger sensibel auf Probleme wie fehlende Datensätze oder andere im Alltag häufig vorkommende Probleme. Sie wissen, wie man damit umgehen kann. Ihre Stärke liegt in der präzisen Analyse von Arbeitsabläufen und der peniblen Kontrolle was die Einhaltung von Abläufen und Fachstandards angeht.

3) *Quantitativer Test: Quantitative Tests* beschreiben die am besten testbaren Eigenschaften. Es lassen sich leicht Metriken erheben und Statistiken führen. Die Erhebung und Auswertung kann automatisiert erfolgen. Somit ergibt sich relativ wenig Aufwand für verhältnismäßig viel Ergebnis. Allerdings schränkt sich der testbare Bereich auf eher triviale Teile, wie die Anzahl an Mausklicks, um eine Aktion zu verrichten, ein.

4) *Qualitativer Test:* Dieser Test beschäftigt sich mit den qualitativen, nicht messbaren Aspekten von Benutzerschnittstellen. Darum ist dies auch der am schwersten zu erfassende Test. Am besten lässt er sich mit einem Freitext abhandeln. Vorgegebene Fragebögen wie beispielsweise der *ISONORM-Fragebogen* versuchen die qualitative Analyse in eine Skala zu fassen.

## IX. INTERAKTION IM WEB

Die Entwicklung von Anwendungen im Internet, speziell von Websites, unterliegt im Allgemeinen den Gestaltungsgrundsätzen und Regeln die bisher vorgestellt wurden. Natürlich gibt es einige Unterschiede und Besonderheiten, die im Folgenden erläutert werden.

### A. Nutzergruppen

Anwendungen werden für Benutzer erstellt. Im Falle der Desktop-Anwendung kann man die Benutzergruppen recht präzise abschätzen. Dies ist im Internet nicht möglich. Websites dienen in erster Linie der Informationsvermittlung. Wer sich für eine Information interessiert, kann dem Entwickler nicht bekannt sein. Das Prinzip der Persona hat also keine sonderliche Auswirkung auf die Gestaltung. Es gibt einfach zu viele.

Mit dem Web 2.0 kommen zu den informationsbezogenen Aspekten noch Weitere hinzu. Wie in *Kapitel II-B.3 auf Seite 1* bereits erwähnt sind dies die gesellschaftlichen Aspekte. Sie spielen in den sogenannten *Social-Network-Plattformen* eine große Rolle. Hier ist stark von der Art der gesellschaftlichen Interaktion abhängig, ob Benutzergruppen oder Personas bestimmt werden können.

Eine Plattform wie *studiVZ*<sup>5</sup>, die sich stark auf Studenten konzentriert, kann annehmen, dass die meisten ihrer Nutzer, wenn auch nicht alle, Studenten sind. Die wenigen Ausreißer sind nicht von Bedeutung. Betrachtet man hingegen die Musik-Plattform *Last.fm*<sup>6</sup>, so werden dort alle gesellschaftlichen Schichten und Musikgeschmäcker repräsentiert. Es ist, trotz des Web 2.0, keine Nutzergruppenbestimmung möglich.

### B. Spezielle Strukturierung

Ähnlich wie bei den in *Kapitel V-D auf Seite 6* erwähnten Hilfesystemen werden Informationen auf Websites gegliedert und strukturiert. Da eine Website kein Menü im Sinne einer Desktop-Anwendung anbietet, auch mit Mnemonics, Tastenkürzeln und Makros nur bedingt umgehen kann, wird oft eine hierarchische Navigation verwendet. Die klassischen Formen dieser Navigationen sind horizontal, beispielsweise bei *Amazon*<sup>7</sup>, zu finden, oder vertikal, wie die Produktkategorien auf

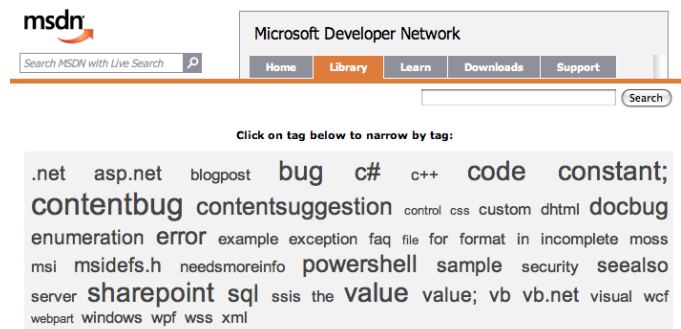


Fig. 2. „Tag-Cloud“  
Quelle: MSDN<sup>d</sup>

<sup>d</sup><http://msdn2.microsoft.com>

eBay<sup>8</sup>.

Beim Anlegen einer Struktur sind diverse Dinge zu beachten. Der Kulturkreis eines Benutzers spielt eine wesentliche Rolle. Im europäischen Kulturkreis wird von links nach rechts gelesen. Eine Navigation wird auf der linken Seite (vertikale Navigation) und oben (horizontale Navigation) erwartet. In asiatisch angehauchten Kulturen wird eine vertikale Navigation auf der rechten Seite vermutet. Auch muss sich der Benutzer immer im Klaren sein, wo er sich befindet. Durch das schnelle und häufige Aktivieren von Hyperlinks (siehe nächster Abschnitt), kann schnell der Überblick verloren gehen. Dann ist eine *Orientierungshilfe* in Form einer Pfadangabe, auch „Bread crumbs“ genannt, (*Startseite > Artikel > Was bringt das Web 2.0?*) angebracht.

Informationen werden mit *Hyperlinks* vernetzt. Sie bieten die Möglichkeit, schnell durch zusammenhängende Informationen zu blättern. Diese klassische Form der Informationsfindung wurde im Web 2.0 um das Prinzip der *Tags* erweitert. Informationen können Kategorien, den Tags, zugewiesen werden. Eine Information kann mehreren Tags angehören. Zusammen mit der Information findet sich meist ein „Pool“ an Tags, die auf die Information zutreffen. Desto zutreffender der Tag, desto deutlicher wird er hervorgehoben. Im Normalfall geschieht dies mit unterschiedlicher Schriftgröße und Schriftvariante. Als Beispiel für Tags kann *Last.fm*<sup>9</sup> genannt werden. Dort werden Musikstücke verschiedenen Musikstilen (Tags) zugeordnet. Die Tags selber sind auch Hyperlinks, die auf Übersichtsseiten mit zu den Tags gehörenden Informationen verlinken.

Um Informationen leichter zugänglich zu machen, wird neben den Hyperlinks und den Tags im Normalfall noch eine (Volltext-) Suche angeboten. Der Unterschied zwischen Web 1.0 und Web 2.0 liegt darin, dass die Suchanfrage im Web 2.0 im Hintergrund verläuft. Somit ist es beispielsweise möglich, während dem Tippen des Suchbegriffes schon Treffer anzubieten. „Google Suggest“<sup>10</sup> verwirklicht eine solche „Echtzeitsuche“.

<sup>5</sup><http://studivz.de>

<sup>6</sup><http://last.fm>

<sup>7</sup><http://amazon.de>

<sup>8</sup><http://ebay.de>

<sup>9</sup><http://last.fm>

<sup>10</sup><http://www.google.com/webhp?complete=1&hl=en>



Fig. 3. Beispiel einer Echtzeitsuche  
Quelle: Google Suggest<sup>11</sup>

<sup>11</sup><http://www.google.com/webhp?complete=1&hl=en>

Ein weiteres Problem von Websites ist die Informationsfülle. Oft sind Artikel mehrere Seiten lang. Eine Lösung für eine klare Struktur wäre Rollbalken (Scrollbars) anzubieten. Wo vertikale Rollbalken meist noch toleriert werden, dort verstoßen horizontale Rollbalken gegen grundlegende Gestaltungsregeln. Aber auch vertikale Rollbalken werden bei zu langen Texten unübersichtlich. Da die Informationen auf Websites oft linear sind, oder linearisiert werden können, bietet sich ein Blätterkonzept, ähnlich dem in einem Buch, an. Hyperlinks verweisen von einer „Seite“ des Textes zur nächsten. Auch hier kann im Web 2.0 der Benutzungsfluss erhalten bleiben. Beim Betrachten einer Seite kann bereits die nächste Seite geladen werden und beim Aktivieren des Hyperlinks muss sie nur noch angezeigt werden.

### C. Gestaltungsmerkmale

Wie bereits erwähnt, werden die klassischen Bedienelemente wie Buttons, Scrollbars und Textfields um *Hyperlinks* erweitert. Wohingegen die klassischen Bedienelemente vom *Graphic-Toolkit* des Browsers bereitgestellt werden, im Normalfall also nicht verändert werden sollten, sind Hyperlinks normaler Text. Dieser kann auf vielfältige Weise vom Entwickler angepasst werden. Die Gestaltungsregel der **Konsistenz** spielt hier eine wichtige Rolle. Hyperlinks sollten klar erkennbar sein, aber vor allem immer gleich innerhalb der Website aussehen. Eventuell kann auch eine Unterscheidung zwischen internen und externen (bspw. mit einem Pfeil versehenen) Hyperlinks getroffen werden.

Formulare, die meist nur aus klassischen Bedienelementen bestehen, werden im Web 2.0 um Funktionalität erweitert. Eingaben können zur Eingabezeit evaluiert werden und der Benutzer bekommt sofort eine Rückmeldung über die Korrektheit der Eingabe. Auch kann ein Formular nur abgesendet werden, wenn alle benötigten Felder, die natürlich hervorgehoben werden müssen, ausgefüllt sind. **Achtung:** Diese Evaluierung geschieht auf der Clientseite und ersetzt **nicht** eine zusätzliche Überprüfung beim Server. Sie dient nur der Bequemlichkeit der Benutzers.

Weiter kann im Web 2.0 normaler Text, der kein Hyperlink

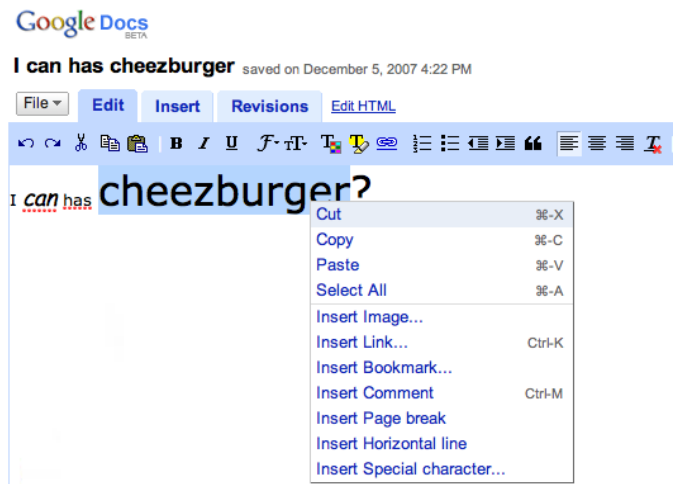


Fig. 4. Google Docs  
Quelle: Google Text & Tabellen<sup>12</sup>

<sup>12</sup><http://www.google.com/google-d-s/intl/de/tour1.html>

ist, zum Interaktionsobjekt werden. Man könnte sich ein Kontextmenü, das nicht vom Browser bestimmt wird, also nicht dessen Funktionen anbietet, vorstellen. Dieses Kontextmenü könnte Bezug auf selektierten Text nehmen und diesen an die Suche auf der Website weitergeben, um Querbezüge herzustellen. Dieses Konzept, und noch einige Weitere, die aus dem Desktop-Bereich entnommen sind, werden genutzt um Office-Anwendungen komplett in Websites auszulagern. „Google Text & Tabellen“<sup>11</sup> ist eine solche Office-Anwendung, die vollständig im Browser läuft und auf gängigen Desktop-Interaktionen wie Kontextmenüs, Clipboards etc. basiert.

### D. Technische Faktoren

Die Entwicklung für Websites zieht eigene Techniken mit sich, die hauptsächlich in diesem Bereich genutzt werden. Die Auszeichnungssprache **HTML**, sowie die Gestaltungssprache **CSS** werden hauptsächlich in der Web-1.0- und Web-2.0-Entwicklung eingesetzt, auch wenn man sich eine Verwendung in anderen Bereichen vorstellen kann. Der Mac OS X **IRC-Client Colloquy**<sup>12</sup> nutzt beispielsweise **CSS**, um seine Chat-Fenster zu gestalten.

**JavaScript** wird im Web 1.0 wie auch im Web 2.0 eingesetzt. In Verbindung mit **XML** entsteht jedoch die Möglichkeit „live“ auf Aktionen und Ereignisse zu reagieren, d.h. eine **HTTP**-Anfrage zu stellen ohne die Website im Browser sichtbar neu laden zu müssen. Diese Technologie nennt man **Ajax**, früher auch **XMLHttpRequest**. Genau genommen, die technische Seite von Web 2.0 betrachtend, wird fast nur mit Javascript und **XML** gearbeitet. Diese Kombination bietet eine Unmenge an Möglichkeiten, eine Website wie eine Desktop-Anwendung reagieren zu lassen. Die meisten anderen Techniken wie **RSS** und Konsorten beziehen sich eher auf gesellschaftliche wie auf bedientechnische Aspekte.

<sup>11</sup><http://www.google.com/google-d-s/intl/de/tour1.html>

<sup>12</sup><http://colloquy.info>

## ANHANG I AKRONYM-ÜBERSICHT

**Ajax** Asynchronous JavaScript and XML  
**CSS** Cascading Style Sheets  
**HTML** Hypertext Markup Language  
**HTTP** Hypertext Transfer Protocol  
**IRC** Internet Relay Chat  
**MCI** Mensch-Computer-Interaktion  
**RSS** Really Simple Syndication  
**SOAP** Simple Object Access Protocol  
**UDDI** Universal Description, Discovery and Integration  
**WSDL** Web Services Description Language  
**WWW** World Wide Web  
**XML** eXtensible Markup Language

## ANHANG II BEGRIFFSERLÄUTERUNGEN

### vierte Gewalt

Neben den drei ursprünglichen Gewalten (Legislative, Exekutive, Judikative) gilt die vierte Gewalt als informeller Begriff für die Presse bzw. die Medien.

### fünfte Gewalt

Die Begriffsanalogie ergibt sich wie bei der [vierten Gewalt](#), nur dass hier der Lobbyismus, oder bezogen auf Web 2.0, das Prinzip der gemeinschaftlichen Einflussnahme einer Interessengruppe, gemeint ist.

## ANHANG III FARBERKLÄRUNGEN

- **Dokument-interne [Akronym-Verweise](#) und [Inhaltsverzeichnis-Verweise](#).**
- **Dokument-interne [Begriffsverweise](#).**
- **Dokument-externe [Hyperlinks](#).**

## LITERATUR

- [1] BOEHM, Barry W.: A Spiral Model of Software Development and Enhancement. In: *IEEE Computer* 21 (1988), Mai, S. 61–72
- [2] COOPER, Alan: *The Inmates Are Running the Asylum*. Sams, 1998
- [3] COOPER, Alan und Robert R.: *About Face 2.0: The essentials of interaction design*. Hoboken, NJ : Wiley & Sons, 2003
- [4] DAHM, Markus: *Grundlagen der Mensch-Computer-Interaktion*. München, Bayern : Pearson Studium, 2005
- [5] DZIDA, W.: Das IFIP-Modell fuer Benutzerschnittstellen. In: *Office Management* 31 (1983), April, S. 6–8
- [6] FOUNDATION, Wikimedia: [http://de.wikipedia.org/wiki/Web\\_2.0](http://de.wikipedia.org/wiki/Web_2.0). In: <http://de.wikipedia.org> (2007), 17. November
- [7] HERCZEG, Michael: *Software-Ergonomie: Grundlagen der Mensch-Computer-Interaktion*. Upper Saddle River, NJ : Addison-Wesley, 1994
- [8] MILLER, G.A.: The magical number seven, plus or minus two: Some limits on our capacity for processing information. In: *Psychological Review* 63 (1956), November, S. 81–97
- [9] PREIM, Bernhard: *Entwicklung interaktiver Systeme*. Berlin : Springer, 1999
- [10] ROYCE, Winston: Managing the Development of Large Software Systems: Concepts and Techniques. In: *Psychological Review* (1970), November
- [11] STARY, Christian: *Interaktive Systeme*. Wiesbaden, Hessen : Vieweg Verlagsgesellschaft, 1999