Proseminar: Human Computer Interaction
Lehrstuhl für Medieninformatik X
RWTH Aachen - SS 2006

Design Language

Designsprachen und ihre Bedeutung für heutige Mensch-Maschine Schnittstellen

Negah Nabbi – 248136
Patrick Zimmer – 250602
Studiengang: Informatik Dipl.

Kapitel 1 - Designlanguages - Designsprachen

1.1 Einleitung

1.2 Designsprache im Vergleich zum klassischen Verständnis der Sprache

1.3 Alltäglicher Gebrauch von Designsprachen

1.4 Designsprache und ihre Bedeutung für Objekte und ihre Benutzer

1.5 Die Struktur einer Designsprache

1.6 Nutzen der Designsprache

- **1.6.1 Interpretation: Designsprachen und Lernen**
- **1.6.2 Produktion: Designspracehn in Unternehmen**
- **1.6.3 Assimilation: Designsprachen und bedeutsame Erfindungen**

1.7 Entwicklung einer Designsprache

- **1.7.1 Charakterisierung**
- **1.7.2 Neupositionierung**
- **1.7.3 Entwicklung und Demonstration**
- **1.7.4 Evaluation**
- **1.7.5 Evolution**

1.8 Bedeutung der Designsprache für aktuelle Interfaces

Kapitel 2 - Die Macintosh Human Interface Guidelines

2.1 Einführung und Übersicht – Mac vs. Anti-Mac

2.2 Die Prinzipien des Mac und Gründe, sie zu brechen

- **2.2.1 Metaphern**
- **2.2.2 Direktzugriff**
- **2.2.3 Zeiger**
- **2.2.4 Konsistenz**
- **2.2.5 WYSIWYG**
- **2.2.6 Benutzerkontrolle**
- **2.2.7 Informationskontrolle**
- **2.2.8 Absicherung und Sicherheitsbestätigung**
- **2.2.9 Stabile Umgebung**
- **2.2.10 Einheitsdesign**
- **2.2.11 Zustandslosigkeit**

2.3 Im Ergebnis: Das Anti-Mac Interface und seine Prinzipien

- **2.3.1 Eine zentrale Rolle der Sprache**
- **2.3.2 Eine detailliertere Repräsentation von Objekten**
- **2.3.3 Ein Ausdrucksstärkeres Interface**
- **2.3.4 Ein hoher Benutzeranspruch**
- **2.3.5 Verteilte Kontrolle**

2.4 Schlusswort

1.1 Einleitung

Der Begriff der Designsprache lässt sich auf John Rheinfrank und Shelly Evanson zurückführen. John Rheinfrank war Entwickler bei Xerox und hatte im Jahr 1992 die Aufgabe, die komplette Serie der Kopiergeräte neu zu designen. Seine Richtlinien waren:
1. Verstehen, wie Menschen in alltäglicher Umgebung mit den Maschinen umgehen
2. Das Gerät wird als Medium betrachtet, über welches der Entwickler/Designer mit dem Nutzer Kommuniziert
Von der bekannten Grafikdesignerin Shelly Evanson kommt ebenfalls die Ansicht, dass die Designsprache Mittel zur Kommunikation zwischen Benutzer und Entwickler darstellt. Wir sprechen im folgenden von Artefakten, mit denen wir genau die Objekte meinen, auf die eine Designsprache ihre Anwendung findet.

1.2 Designsprache im Vergleich zum klassischen Verständnis der Sprache

Sprache ist für uns nicht nur ein Werkzeug, dass wir benutzen. Sprache ist ein relevanter Teil unseres Lebens. Sie dient nicht nur zum Informationsaustausch, sondern auch zum Ausdruck von Beziehungen und Gefühlen. Genau wie wir die Worte unserer Sprache benutzen um Sätze und Aussagen zu formulieren, benutzen wir in der Designsprache Grundelemente, aus denen wir gewisse Regelmäßigkeiten in der Gestaltung und im Umgang mit unserer Umwelt ableiten. Genau wie bei der natürlichen Sprache ist die Designsprache schon untrennbar in unser Leben integriert; ob es niedergeschriebene Regeln gibt ist dabei irrelevant. Nahezu in allen Bereichen unseres Lebens kommen Designsprachen mehr oder weniger bewusst zur Anwendung.
Wird eine Designsprache konsequent entwickelt, verstanden und angewandt, ergeben sich viele Vorteile in den entsprechenden Umgebungen und der Interaktion damit.

1.3 Alltäglicher Gebrauch von Designsprachen

Designsprachen sind allgegenwärtig. Die meisten haben sich unbewusst entwickelt.
Beispielsweise passt sich die Architektur einerseits den natürlichen Gegebenheiten und andererseits der schon vorhandenen künstlich erschaffenen Umgebung an. Von bewusster Entwicklung sprechen wir bei fast allen Arten von fassbaren Gütern, sei es Kleidung, Nahrung oder Luxusartikel. Jedes Unternehmen hat Richtlinien, die sich in der Erscheinung der Produktlinien wiederfinden.
Designsprachen werden gerne bewusst entwickelt, um neue Technologien in den Alltag zu integrieren. Xerox hat dazu beisielsweise ein Programm entwickelt, das dem Benutzer immer wieder gleich gekennzeichnete Bedienelemente bietet. Oft unbewusst können dann Benutzer anderer Xerox Produkte das neue Gerät sofort verstehen und bedienen.
Die verwendeten Mittel können Formen, Farben aber auch spezielle Mechanismen sein, die immer wieder in der selben Form auftreten, zum Beispiel immer gleich gestaltete und positionierte Griffe für Klappen oder Schubfächer.
So wie wir auch unsere Muttersprache nicht strukturell, sondern einfach durch ihre Verwendung erlernen, funktioniert auch der Umgang mit Artefakten, die unter den Richtlinien einer Designsprache entwickelt wurden.

1.4 Designsprache und ihre Bedeutung für Objekte und ihre Benutzer

Die Designsprache ist das Mittel, mit dem Designer ihren Produkten eine Bedeutung geben, so dass diese schon in der äußeren Erscheinungsform vom Benutzer erkennbar ist. Sowie die natürliche Sprache aus Wörtern und Grammatik besteht, besteht auch eine Designsprache aus Grundelementen und Regeln, wie diese zu kombinieren sind.
Designsprache wird gesprochen und verstanden, in dem Sinne, dass damit erschaffen, sowie interpretiert wird.
Designsprachen sind damit die Basis dafür, wie Dinge sich darstellen und wie Menschen Dinge verstehen.
Traditionell wird die Bedeutung und der Nutzen eines Objektes an Form und Aussehen festgemacht. Designsprachen werden jedoch um so effizienter, wird der Kontext mit einbezogen. Die Fragestellung erweitert sich in dem Falle darauf, wieviel Sinn ein Objekt macht, in Bezug auf seine Umgebung. Damit ist nicht unbedingt die örtliche Umgebung gemeint, dies kann sich auch auf Erfahrungen mit dem Objekt oder Erwartungen an das Objekt beziehen.

1.5 Die Struktur einer Designsprache

Typischerweise bestehen Designsprachen aus:

1. Einer Sammlung von Grundelementen; die Grundmittel der Kommunikation, zum Beispiel Farben, Formen, Vorgänge oder Bilder.
2. Kompositionsregeln, welche vorgeben, wie die Grundelemente miteinander kombiniert werden können.
3. Sammlung von Anwendungsgebieten, dies sind Beispiele, wie sich die Elemente und deren Komposition verändern können, abhängig von ihrer Umgebung.

Designsprachen sind keine festen Strukturen. Sie unterliegen natürlichen Mechanismen, durch die sie sich durch fortwährende Interaktion verändern und weiterentwickeln. Die Benutzer der Objekte reagieren auf die Elemente, die ihnen von den Designern zur Verfügung gestellt werden. Durch Erfolg oder Misserfolg tragen sie zur Überarbeitung der Sprache bei. Durch die bewusste Entwicklung einer Designsprache kann dieser Revisionsprozess erheblich beschleunigt werden.
Designsprachen entwickeln sich kontinuierlich und festigen sich mit der Zeit. Sie werden zu Tradition und schwer angreifbar. In der Regel werden geltende, verinnerlichte Regeln nicht in Frage gestellt, was in gewisser Weise eine Gefahr darstellt. Ein Unternehmen, welches eine Sprache benutzt, in der es die Kunden nichtmehr voll und ganz ansprechen kann, läuft Gefahr, dass sich Konkurrenten mit neuen, angepassteren Produktlinien in den Markt drängen.

1.6 Nutzen der Designsprache

Vorteile lassen sich aus drei verschiedenen Richtungen betrachten: Interpretation, Produktion, Assimilation.

1.6.1 Interpretation: Designsprachen und Lernen

Designer geben dem Benutzer die Möglichkeit zum „learning-by-doing". D abei profitieren sie von Dingen, die der Nutzer bereits kennt und mit denen er umgehen kann. Als Beispiel kann der Designer eines Kopierers, der die Sprache der Türgriffe kennt diese in Form von „D inge öffnen" auf diverse Klappen oder Fächer des Gerätes übertragen. So werden Benutzer mit physischen Hinweisen bedient, die „ö ffne mich!" vermitteln, ohne mit diesen Worten beschriftet zu sein.
Eine wichtige Angelegenheit ist die Unterscheidung zwischen dem sogenannten „B lack-Box-Design" und dem „Tran sparent-Box-Design". Ein mit dem Prinzip Black-Box-Design gestaltetes Artefakt verbirgt seine Funktionalität, in dem Sinne, dass der Benutzer keine Einflussmöglichkeiten auf die inneren Prozesse hat, sondern nur die offensichtlichen Eingaben tätigen kann und das Ergebnis präsentiert bekommt. Eine solche Konstruktionsweise gilt als eher Narrensicher.
Im Gegensatz dazu kann ein Designer dem Benutzer eine gute Lern-Hilfestellung geben, wenn er sich für das Transparent-Box-Design entscheidet. Die Funktionalität ist dabei zu einem gewissen Grad nachvollziehbar und beeinflussbar. Nichtsdestotrotz bleibt die Möglichkeit, dem Benutzer gewisse Voreinstellungen zu bieten, damit dieser nicht überfordert ist. Die Bedeutung eines Artefakts kann sich so gemeinsam mit dem Benutzer weiterentwickeln, während dieser lernt mehr und mehr Möglichkeiten selbstständig zu nutzen.
Wünschenswert ist, dass zwischen ähnlichen Produkten auch Konsistenz in der Bedienung zu finden ist, so können die Nutzer von Erfahrungen profitieren und beginnen mit dem Lernprozess nicht von vorne.

1.6.2 Produktion: Designsprachen in Unternehmen

Unternehmen können Designsprachen auf viele Weisen nutzen, um über ihre Produkte einen Eindruck über das Gesamtunternehmen zu vermitteln. Eine konsistente Qualität bei konsistentem Design und Funktionalität sorgt für ein entsprechendes Bild des Unternehmens beim Kunden. Die Meinung eines Kunden über das Unternehmen prägt im Gegenzug wiederum das zukünftige Kaufverhalten bei neuen oder anderen Produkten.
Gute Designsprachen vermitteln nicht nur Genanntes, sondern darüberhinaus vor allem starke Relevanz des Produktes für den Nutzer. Trifft eine Sprache eine breite Masse an Zielpersonen und wird im Generellen akzeptiert, bedeutet dies eine starke Marktstellung für ein Unternehmen. Konkurrenten bleibt dann nur die Möglichkeit diese Richtlinien absolut zu übernehmen, oder aber gleichstarke Werte zu erschaffen, was allerdings um so schwerer wird, je länger sich eine Sprache bzw. Produktlinie behauptet.
Nur eine neue oder überarbeitete Produktlinie, die plötzlich mehr als zuvor die Bedürfnisse der Verbraucher erfüllt, kann erfolgreich sein, insbesondere dann, wenn ein Unternehmen das gesamte Spektrum seiner Möglichkeiten ausnutzt und die eigene Designsprache über Werbung, Service, Verkauf, Erfahrungen, etc. den Kunden vermittelt.

1.6.3 Assimilation: Designsprachen und bedeutsame Erfindungen

Eine Sache macht Designsprachen mächtig, aber auch gefährlich in einer sich rasend schnell entwickelnden Umgebung. Die Präsenz über einen langen Zeitraum sorgt für eine Assimilation dieser Sprache ins alltägliche leben. Dort üben sie den größten Einfluss auf Menschen aus, oft unterbewusst. Unternehmen können sich dies zu Nutzen machen, indem sie neue Erfindungen durch alte Werte charakterisieren und so den Einfluss der längst Assimilierten Sprachen der Vergangenheit ausnutzen. So gingen zum Beispiel die ersten Automobile als pferdelose Kutschen an die Verbraucher. Es ist dann der Evolutionsprozess, dem jede Sprache unterliegt, der bei genügender Bekanntheit und Vertrautheit dann den entscheidenden Schritt weg von den alten Werten ermöglicht.

1.7 Entwicklung einer Designsprache

Die Erstellung einer Designsprache lässt sich in 5 Schritte unterteilen:

1.7.1 - Charakterisierung

Bei der Charakterisierung geht es um die Feststellung der vorhandenen Gegebenheiten. Wichtig dafür sind insbesondere die im angepeilten Bereich vorherrschenden bereits assimilierten Designsprachen, die grundlegend erstmal zusammengefasst werden, ohne sie in Frage zu stellen. Erst mit einer kompletten Übersicht der Verhältnisse, kann damit begonnen werden, darauf neue Aspekte aufzubauen oder aber auch Gegebenes zu ändern, wobei aber die Grundlagen nie ausser acht gelassen werden sollten.

1.7.2 - Neupositionierung

Durch verschiedenste Methoden wird ein neuer Rahmen an Bedingungen gesucht, der die aktuellen Bedürfnisse der Zielgruppe gut widerspiegelt. Diese Sammlung entsteht neben dem vorher festgestellten Rahmen und funktioniert am besten, wenn alle beteiligten Personen: Entwickler, evtl. Händler und Vertreter der Endverbraucher am Entwicklungsprozess beteiligt werden. Eine breite Palette an Werkzeugen bietet je nach Anwendung die Marktforschung oder die Verhaltensforschung. Geht es beispielsweise um die Einführung einer neuen Produktlinie, sind Meinungen und Verbesserungsvorschläge zu bereits existenten ähnlichen Produkten erforderlich, um das Produkt für die Zielgruppe interessant zu gestalten. Handelt es sich um die Gestaltung von Bedienelementen irgendeiner Form ist es unerlässlich zu wissen, wie der Umgang mit ähnlichen Systemen in der Vergangenheit funktioniert hat. Dies führt uns auch fließend zum nächsten Punkt, bei dem eine Verhaltensforschung praktisch live während der Entwicklung stattfindet.

1.7.3 Entwicklung und Demonstration

Entwicklung beginnt mit dem ersten Erstellen sichtbarer Ergebnisse. Das Designteam fügt die vorher gesammelten Informationen zusammen, kombiniert den alten Rahmen mit den neuen Annahmen und Erkenntnissen und erstellt Prototypen. Gleichzeitig mit der Erstellung werden die Prototypen schon testweise in fiktiven Umgebungen präsentiert. Dies stellt sicher, das die Wechselwirkung der Grundelemente miteinander und die Wirkung auf die Benutzer besser eingeschätzt und genutzt werden kann. Dies bedeutet nicht, dass die vorigen Phasen generell abgeschlossen sind, gerade die Kommunikation mit den Benutzern über die Prototypen wird intensiv beobachtet um etwaige Unzulänglicheiten auszubessern.

1.7.4 Evaluation

Im Evaluationsprozess wird die entwickelte Designsprache nun in realem Kontext platziert, um zu sehen, wie sie auf die Benutzer wirkt. Im Idealfall findet die Evaluation bereits nach oder während der ersten Demonstration statt. Das Designteam erstellt mehrere Objekte, die unter den Aspekten der Sprache designt sind und prüft das Verhalten in den zukünftig erwarteten Situationen. Während der Evaluation gibt es weiterhin intensiven Kontakt mit Entwicklern und Nutzern, um in einem iterierenden Prozess die Sprache immer weiter zu präzisieren. Der Evaluation ist ein eigenes Kapitel gewidmet, darum bedarf es hier keines tieferen Einblicks.

1.7.5 Evolution

Während die Anforderungen an Produkte sich im Laufe der Zeit ändern, sowie sich auch die Wahrnehmung von Produkten und deren Eigenschaften ändern kann, wird sich auch die Designsprache mit der Zeit anpassen. Die Designer müssen sich dessen bewusst sein, dass eine Designsprache niemals ein abgeschlossenes Werk darstellt. Auch niedergeschriebene und bewusst erstellte Regeln können sich im Allgemeinen nicht auf Dauer halten, da die Autoren und Designer zukünftige Ereignisse und Trends nicht mit Bestimmtheit einplanen können. So unterliegt jede Designsprache, wie auch unsere natürliche Sprache, einem Evolutionsprozess. Sie entwickelt sich mit ihrer Umwelt, bewusst oder unbewusst. Auch die von Winograd angeführten "Macintosh Human Interface Guidelines" sind heute an vielen Stellen überholungsbedürftig und liegen längst nicht mehr in vollem Umfang dem aktuellen Mac OS zugrunde.

1.8 Bedeutung der Designsprache für aktuelle Interfaces

Designsprachen sind mächtige Werkzeuge; sie drücken aus, was wir über unsere Umwelt wissen und teilen. Henry Ford erfand eine Sprache, die es ihm ermöglichte Autos am Fließband zu produzieren und damit Verbraucher zu erreichen. Erfindungen und Weiterentwicklungen fanden zu hauf statt in dieser Zeit, wohingegen die Sprache sich nur langsam entwickelt hat. Bei Softwaredesign scheint es einen ähnlichen Lauf zu nehmen. Schon für Jahre und vermutlich noch weit in die Zukunft werden Fenster, Icons, Menüs und Zeiger (das WIMP Interface) unantastbarer Standard für Computerinterfaces sein. Spiele allerdings, oder stärker Spiel-basierte Anwendungen scheinen andere Arten von Designsprachen zu besitzen. Wir beobachten, dass gerade die Entwicklung von Sprachen, die auf ganz spezielle Kontexte bezogen sind, viele neue Ideen hervorbringt und vielleicht zur Evolution oder sogar Revolution der WIMP-Sprache führen wird, in Hinblick auf eine intensivere Interaktion mit Computern.

Kapitel 2

Die Macintosh Human Interface Guidelines

2.1 Einführung und Übersicht

Die Macintosh Guidelines, erstmals veröffentlicht 1987 von Apple, basieren auf dem klassischen WIMP (Windows, Icons, Menus, Pointer) Interface. Um genau zu sein, haben sie dieses sogar geprägt.
Das Dokument ist zweigeteilt. Im ersten Teil werden die grundlegenden Prinzipien dargestellt und erläutert, dazu gleich mehr. Der zweite Teil behandelt die einzelnen Elemente des Interfaces im Detail. Letzterer ist Pflichtlektüre für jeden Apple Programmierer und passt sich mit der Weiterentwicklung der Mac OS Betriebssysteme weiter an, während die Leitprinzipien seit jeher unangetastet sind. Für jedes denkbare Interfaceelement existieren genaue Vorschriften wo und wie es verwendet werden darf. Position, Form, Farbe, evtl. Textart und -Stil sowie die Funktionalität sind haarklein beschrieben.

Durch die heutige Integration von Computern in den Alltag ist das Aussehen dieser Interfaces - sei es auf einem Mac, Windows oder Linux System - bekannt und Intuitiv erkenn- und bedienbar. Die einzelnen Elemente zu behandeln wäre eine sehr ausgedehnte und langweilige Lektüre.
Aus diesem Grund wollen wir uns nicht lediglich auf eine Darstellung dieses Dokumentes beschränken, sondern stellen eine alternative - wenn auch theoretische - Lösung als Ersatz des klassischen Macintosh Interfaces vor: den Anti-Mac.

Wir halten uns dabei an die Veröffentlichung "The Anti-Mac Interface" von Don Gentner und Jakob Nielsen. Die beiden Autoren legen ihrer Untersuchung Das zugrunde, womit Rheinfrank und Evanson in ihrem Kapitel über Designsprachen aufhören, nämlich die Aussage, dass die Softwareindustrie feststeckt und die Neuentwicklung bzw. Weiterentwicklung des klassischen WIPM Interfaces verschlafen hat.
Ausgehend davon erlauben sich Gentner und Nielsen, sich die Grundsätze der Interfacegestaltung aus den "Macintosh Human Interface Guidelines" herauszunehmen und allesamt konsequent zu ignorieren, um ein neuartiges Interface zu schaffen.

2.2 Die Prinzipien des Mac und Gründe, sie zu brechen

Wir stellen nun dar, welche Prinzipien dies im Einzelnen sind und werden Gründe der Autoren für die Abschaffung dieser Prinzipien anführen, um zum Schluss das durch die pervertierten Grundsätze erstellte hypothetische Interface zu präsentieren.

2.2.1 Metaphern (Metaphors)

Das Verwenden von Bildern für Objekte und Programme soll dem Benutzer das Erkennen und Benutzen spezifischer Dienste erleichtern. So ist die Grundlage jedes üblichen Interfaces der Desktop, der stark angelehnt ist an sein reales Vorbild, den Schreibtisch. Dateien und Ordner werden mit entsprechenden Symbolen versehen, und lassen sich ähnlich der Realität ablegen, öffnen, verschieben und ordnen. Ein weiteres deutliches Beispiel für eine Metapher ist der Papierkorb, in dem wir, wie in Wirklichkeit, Dateien entsorgen, indem wir sie schlicht "hineinwerfen". Der Inhalt bleibt solange erhalten, bis der Papierkorb geleert wird.
Für die Autoren steht fest, dass diese Analogie in den Anfängen der Personal Computer dafür gesorgt hat, dass unerfahrene Benutzer leicht mit der neuen Technik umgehen konnten. Heute, in einer Generation, die komplett mit Computern aufgewachsen ist, ist diese festgefahrene Analogie allerdings ein Rückschritt und birgt folgende Gefahren bzw. Unzulänglichkeiten:

- das repräsentierte Computerprogramm oder Objekt hat Funktionen und Eigenschaften, die das zugrunde liegende reale Objekt nicht besitzt. So wird ein Benutzer der ein als Schreibmaschine dargestelltes Textverarbeitungsprogramm startet wohl von selber nicht Funktionen wie: "Suchen und Ersetzen" finden und benutzen, da er diese von der Schreibmaschine nicht kennt.
- andersherum kann auch das reale Objekt Bedienungsmöglichkeiten haben, die das Programm nicht hat. So kann man in eine Schreibmaschine nahezu jedes Format an Papier einführen und beschreiben, bei einem Programm bedarf es dabei für gewöhnlich umständlicher Einstellungen und wird versucht, gar einen Briefumschlag an einer bestimmten Stelle zu beschriften, scheitern Einsteiger komplett.
- andere Funktionen oder Eigenschaften können sehr unterschiedliche Bedeutung haben, zum Beispiel die Behandlung von Leerzeichen, Tabs und Zeilenumbrüchen.

2.2.2 Direktzugriff (Direct Manipulation)

Das Prinzip „Direct Manipulation“ erlaubt es dem Benutzer, auf kleinster Ebene direkt mit den Objekten (Dateien) zu arbeiten, zum Beispiel sie zu verschieben, zu kopieren oder zu löschen. Das funktioniert, solange es sich um einzelne Objekte und einzelne Aktionen handelt; erhöhen wir jedoch die Anzahl der Objekte auf eine ganze Gruppe von Objekten und wollen komplexe Befehlsfolgen darauf ausführen versagt das WIMP Interface.
Dazu kommt, dass die erreichte Präzision sich darauf beschränkt, wie präzise der jeweilige Nutzer tatsächlich mit dem Zeigegerät umzugehen vermag. Wünschenswert wäre daher eine komplexere Steuerung die auch, gestützt von mathematischen Funktionen, präzisere Ergebnisse erzielt während komplexe Aufgaben, die zum Beispiel an Bedingungen geknüpft sind, erledigt werden.

2.2.3 Zeiger (See-and-Point)

Das „See-and-Point“ Prinzip erlaubt uns, das was wir auf dem Bildschirm sehen, mit der Maus anzuklicken um Informationen zu erhalten. Nach Gentner und Nielsen handelt es sich dabei um die primitivste Form der Kommunikation. Das was unsere komplexe Kultur ausmacht, die Sprache, wird wertlos und wird durch stures Zeigen ersetzt.
Erstrebenswert hier wäre ein Interface, dass sich der Sprache bedient um genauer und effizienter zu arbeiten. Damit ist kein sprachgesteuerter Computer gemeint; die Möglichkeit des Zeigens soll nicht verbannt werden, stattdessen soll eine sinnvolle Kombination beider Prinzipien erreicht werden.

2.2.4 Konsistenz (Consistency)

Konsistenz spielt nicht nur in den Mac Interface Guidelines eine große Rolle, sondern ist das wichtigste Prinzip aller Designsprachen. Das Ziel eines konsistenten Designs über mehrere Produkte bzw. Programme hinweg ist, den Benutzer beim erlernen neuer Techniken zu unterstützen. Funktioniert das Prinzip, kann ein Benutzer, der ein Programm schon bedienen kann, ein anderes intuitiv schneller erlernen und benutzen.
Die Problematik besteht hier nicht im Prinzip selber sondern in einer zu konsequenten Anwendung dieses Prinzips. In der Realität gibt es große Unterschiede zwischen Objekten, dessen Funktion doch als ähnlich oder gleich vom Menschen erkannt wird. Bücher zum Beispiel sehen von Grund auf verschieden aus, werden aber doch als solche erkannt. Die Unterscheidung zwischen Büchern wird erst durch ihr individuelles Aussehen erreicht, schließlich wollen wir Bücher nicht erst lesen, um sie zu erkennen. So kann man das auch für den Computer umsetzen und die Konsistenz nur zu einem gewissen niedrigen Grad einhalten. Dabei ist allerdings zu beachten, dass Objekte oder Programme mit gleichen Eigenschaften oder Funktionen zwar nicht gleich aussehen müssen, andersherum aber von der Erscheinung her ähnliche Dinge auch ähnliche Funktionen haben sollten.

2.2.5 WYSIWYG ("What you see is what you get")

"What you see is what you get" sagt genau das, was es meint. Bezogen zum Beispiel auf eine Textverarbeitung, sagt es aus, dass der Inhalt, der auf dem Monitor sichtbar ist und den man dort live bearbeitet, später auch genau so in der gedruckten Version aussehen wird.
Das Problem besteht darin, dass hier eine große Einschränkung vorgenommen wird. Oft kann es von Vorteil sein, Informationen zu haben, die über die reine gedruckte Textinformation hinausgehen. Es kann wichtig sein im Nachhinein zu wissen, warum man eine ganz bestimmte Stelle so gestaltet hat, wie sie tatsächlich vorliegt.
Im Hinblick auf Programme, wie Screenreader, die nach dem WYSIWYG Standard schlicht allen Text von Bildschirm vorlesen können, wären Informationen wünschenswert, welche intelligenteren Programmen ermöglichen, mit etwas Hintergrundwissen sinnvoller zu arbeiten.
Die digitale Darstellung von Informationen aller Art bietet so viel mehr, als nur die schlichte Aneinanderreihungen von Informationen. So können zum Beispiel Nachrichten so gestaltet werden, dass wichtige Informationen erst erscheinen, wenn der Nutzer diese sehen will, oder Möglichkeiten bieten gleich verschiedene Sichtweisen darzustellen, ohne das diese Informationen in einer Gesamtansicht in einem Meer von Text untergehen.

2.2.6 Benutzerkontrolle (User Control)

Der Benutzer soll prinzipiell die Kontrolle über alles haben, was in einem Rechner vorgeht. Mit der Zeit und dem technischen Fortschritt hat sich allerdings gezeigt, dass einerseits viele Vorgänge zu langweilig und zeitaufwendig sind,

um sie regelmäßig selbst zu bearbeiten und es zum anderen auch viele Prozesse gibt, die so komplex sind, dass ein normaler Benutzer sie gar nicht verstehen kann.
Das Prinzip hat heute nurnoch untergeordnete Bedeutung und wird nicht bis ins Kleinste praktiziert. Das zeigt sich allein schon darin, dass Programmierer heute nichtmehr in Maschinencode schreiben, sondern sich von Compilern und Interpretern einen Großteil der Arbeit abnehmen lassen.
Auch wenn „User Control“ ein wünschenswertes Prinzip wäre, so hätte es doch spätestens mit der Etablierung von vernetzten Systemen und insbesondere dem Internet ausgedient. Dort gibt es so vielfältige Einflüsse sowohl von der Technik als auch von anderen Individuen, dass dort nicht ein Benutzer die volle Kontrolle bewältigen kann.

2.2.7 Informationskontrolle (Feedback and Dialog)

Dieser Punkt ist stark verbunden mit dem Vorherigen Prinzip der Benutzerkontrolle. Mit sinkender Kontrolle sinkt allerdings auch der Bedarf an detaillierter Information. Dort, wo Programme Aufgaben übernehmen, braucht der Benutzer keine Informationen, solange die Programme ihren Dienst ohne ein Eingreifen des Nutzers verrichten können.

2.2.8 Absicherung und Sicherheitsbestätigung (Forgiveness)

Das Prinzip des Vergebens schreibt vor, dass generell alle Benutzeraktionen rückgängig gemacht werden können, und dass das endgültige Verändern oder Entfernen von Inhalten immer eine gesonderte Bestätigung vom Benutzer erfordert. Dieses Prinzip dient der Benutzerfreundlichkeit und hilft Einsteigern möglicherweise dabei, sich vor sich selbst zu schützen. An dieser Stelle fordern die Autoren nicht die Abschaffung dieser Hilfestellung sondern einen intelligenteren Umgang damit. Sie stellen sich ein System vor, dass auf die letzten Aktionen des Benutzers zurückblickt, um zu entscheiden, ob dem Benutzer eine entsprechende Bestätigungsmeldung gezeigt wird, oder ob seine vorangegangenen Aktionen nicht sowieso unwiderruflich zu diesem Ergebnis führen würden. Wird eine Datei beispielsweise gelöscht und ein Fragedialog erzeugt, der eine Bestätigung verlangt, sollte auf eine erneute Frage verzichtet werden, wenn weitere ähnliche Dateien gelöscht werden.

2.2.9 Stabile Umgebung (Perceived Stability)

Perceived Stability sagt uns, dass während der Abwesenheit des Benutzers seine Umgebung nicht verändert wird. Dies hängt auch insbesondere mit der Benutzerkontrolle zusammen.
Leiten wir die Computerumgebung von der realen Welt ab, wird dieses Prinzip jedoch auch hinfällig. Unsere Umgebung ist voll von Einflüssen, die wir nicht bestimmen, ist ständig in Bewegung und verändert sich. Viele Programme entkommen diesem Prinzip schon heute, zum Beispiel E-Mail-Programme, die automatisch Mails holen und sie für uns sortieren, ohne dass wir dazu am Computer sitzen müssen. Auch hier ist das Internet wiederum das beste Beispiel für eine verteilte Kontrolle, so würden Newsgroups oder Foren überhaupt keinen Sinn machen, würden sie sich nie ändern in der Abwesenheit eines Benutzers.

2.2.10 Einheitsdesign (Aesthetic Integrity)

Dieses Prinzip verlangt eine starke Konsistenz im Design. Wir zählen es auch zu denen, die im laufe der Zeit an Bedeutung verloren haben, bzw dessen Aussage stark abgeschwächt wurde. Heutzutage wirkt ein Interface in dem alle Bedienelemente exakt gleich aussehen schlicht langweilig und uninteressant. Gerade das Internet zeugt davon das ein intensiverer Einsatz von Farben, Formen und Bildern an Bedeutung gewonnen hat.

2.2.11 Zustandslosigkeit (Modelessness)

Zustandslosigkeit soll dem Benutzer ermöglichen, egal wo er sich befindet im Interface, alle Aufgaben erfüllen zu können. Schon die Macintosh Interface Designer brachen etwas mit diesem Prinzip, indem sie eher darauf zielten dem Benutzer nahezulegen, wie man verschiedene Zustände effizient Benutzt.
Das Leben ist stark in Zustände geteilt, so kann man im Schwimmbecken andere Aufgaben erledigen, als in der Küche. Die Autoren wollen dieses Prinzip auch auf das Interface übertragen. Dies ist heute auch in vielen Betriebssystemen möglich oder gar notwendig.

2.3 Im Ergebnis: Das Anti-Mac Interface und seine Prinzipien

Während der Auflistung und Diskussion der einzelnen Macintosh Human Interface Guideline Prinzipien fällt auf, dass die strikte Ignorierung dieser Grundsätze nicht immer möglich ist, bzw zu sehr fragwürdigen Ergebnissen führen würde. Das Essenzielle jedoch herausgenommen, dazu einige Prinzipien abgewandelt oder abgeschwächt kommen wir zu unserem Anti-Mac Interface, welches auf folgenden gegensätzlichen Leitprinzipien basiert:

2.3.1 Eine zentrale Rolle der Sprache

Sprache hat sich über mehrere hunderttausend Jahre entwickelt und bietet uns mächtige Werkzeuge, um zum Beispiel über Objekte zu reden, die nicht greifbar sind, um über mögliche Aktionen zu reden und um Bedingungen oder andere Konzepte intuitiv zu benutzen.
Es stellt sich die Frage, wie wir diese Vorteile in unserem Interface ausnutzen können. Wir reden hier nicht über das „Natural Language Interface“, wie es sich KI-Forscher vorstellen; dies ist sicherlich in weiter Ferne. Denkbar hingegen wäre ein System, angelehnt an das Prinzip alter Text Adventures: Eine Kommandozeile, die die Eingabe bis zu einem gewissen Grad syntaktisch aufschlüsselt, während sie Synonyme, Rechtschreibfehler und Ungenauigkeiten in der Grammatik gewissermaßen toleriert. Eine Auflistung von passenden Aktionen sollte dann die Antwort auf eine solche natürliche Anfrage sein.

2.3.2 Eine detailliertere Repräsentation von Objekten

Aktuelle Interfaces verfügen über sehr eingeschränkte Informationen über Dateien. Mit Angaben zu Autoren, Erstellungsdatum, Angaben zur letzten Benutzung ist zwar einiges in letzter Zeit noch hinzugekommen zu den klassischen Informationen, wie Name und Dateigröße; für das Anti-Mac Interface stellen wir uns allerdings eine viel weiter gehende Beschreibung vor. Erstrebenswert wären dabei zum Beispiel Relationen zu anderen Dokumenten, eine Liste von Schlüsselwörtern, ähnlich, wie man es von modernen Voll-Text Suchmaschinen kennt, aber noch darüber hinaus.
Stellen wir unsere Dokumente um auf eine nicht mehr dem Prinzip WYSIWYG unterworfene Struktur, stehen uns entsprechende Informationen zur Verfügung. Damit kann auch der Computer ein höheres Maß an Kontrolle übernehmen, aufzeichnen wie und wann wir Objekte benutzen und deren Darstellung entsprechend anpassen.

2.3.3 Ein vielfältigeres Interface

Die vorher behandelten Punkte bieten uns nun die Möglichkeit die Darstellung des Interfaces auch entsprechend zu verbessern. Überlegt man sich, dass das erste Macintosh Interface für einen 9 zoll Monitor mit gerade einmal 200,000 schwarz/weiss Pixeln konzipiert war, so hat sich das Informationsangebot auf heutigen Displays mit 3 Millionen oder mehr Pixeln bei 32 Bit Farbtiefe, weit mehr als verhundertfacht. Die Tatsache dass verschiedene Dokumente lediglich mit dem gleichen Programm erstellt wurden ist heute ausschlaggebend dafür, dass diese Dokumente auch gleich dargestellt werden, was definitiv nicht notwendig ist. Als Beispiel können wir verschiedenst aussehende Bücher doch als solche identifizieren und die Suche nach einem speziellen Buch wird anhand von individuellen Merkmalen erheblich erleichtert. Wünschenswert wäre das Prinzip, das einige Grafikanwendungen schon heute bieten, das Ersetzen des Programmsymbols durch ein Thumbnail des Bildes. Abgerundet durch passende Sound-Unterstützung und neuartige Eingabemethoden ergibt sich die gewünschte Funktionalität des Anti-Mac Interfaces.

2.3.4 Höhere Ansprüche an den Benutzer

Benutzerschnittstellen stellen immer einen Kompromiss dar, zwischen Bedienfreundlichkeit und Funktionalität. Ziehen wir einen Vergleich dieses Kompromisses mit unserer natürlichen Sprache, so müssen wir uns eine Gesellschaft vorstellen, in der das Zeigen auf eine Auswahl von Objekten die höchste Form der Kommunikation darstellt. Zweifelsfrei bleibt uns das erspart, da wir Jahrhunderte damit verbracht haben unsere Sprache zu einem komplexen, mächtigen Werkzeug zu entwickeln. Ausgehend davon, dass die heranwachsende Generation, mehr und mehr zeit mit dem Computer verbringt, ist davon auszugehen, dass sich der oben erwähnte Kompromiss immer weiter in Richtung der Funktionalität oder des "Work-flows" verschiebt. Zukünftige Interfaces sollten dies daher berücksichtigen und die Ansprüche - und somit auch die Leistungsfähigkeit - nach oben zu schrauben.

2.3.5 Verteilte Kontrolle

Die bisherigen Prinzipien steigern die Anforderungen an einen Benutzer natürlich enorm. Da Computer aber auch in Zukunft keine Arbeit schaffen sondern sie bewältigen sollen, greift beim Anti-Mac das Prinzip der verteilten Kontrolle. Computergesteuerte „Agents“ (Dienstprogramme) übernehmen heute schon einfache Aufgaben. Aber auch, wenn Programmierer Interesse an Agents zeigen, sind sie doch lange nicht in der Lage auch komplexe Aufgaben selbstständig zu erfüllen. In unserem Interface sind Agents allgegenwärtig und in der Lage, mit dem Benutzer zu kommunizieren, um komplexe Aufgaben zu übernehmen. Auch andere Individuen sollen gewissen Einfluss auf unsere Umgebung haben, wie in der Realität. Das Internet wäre ohne diese Prinzipien nicht existent. Wichtig dabei bleibt, dass jeder Benutzer das Maß an Kontrolle selbst bestimmen kann.

2.4 Schlusswort

Die ausführliche und detaillierte Diskussion der einzelnen Prinzipien und die Anführung guter Gründe, warum diese nicht mehr ausschlaggebend sind oder seien sollten führt uns zu einer umfangreichen neuen Sicht der Dinge. Wie auch die Autoren des Anti-Mac Interfaces möchten wir deutlich darauf hinweisen, dass diese Ausarbeitung keine Kritik am Macintosh und den angeführten Prinzipien darstellt, schließlich sind wir alle glückliche Nutzer eines zumindest Mac-abgeleiteten Betriebssystems. Wir blicken vielmehr ein Stück weit in die Zukunft, da bei allem technischen Fortschritt die endgültige Entwicklung eines Interfacekonzeptes mit so weitreichenden Änderungen nicht unbedingt uneingeschränkt möglich ist und auf jeden Fall eine Menge an Zeit verschlingen wird. Nichtsdestotrotz ist es vielleicht an der Zeit mit einigen grundlegenden stukturellen Neukonzipierungen zu beginnen, die im Hinblick auf die neu erarbeiteten Prinzipien eine deutliche Weiterentwicklung ermöglichen können.

Literaturverzeichnis

Dieser Vortrag basiert auf dem vierten Kapitel des Buches
„Bringing Design to Sofware“ von Terry Winograd (1996 ACM Press)

Bei der Vorbereitung stützten wir uns weiterhin vor allem auf die Veröffentlichung
„The Anti-Mac Interface“ von Don Gentner und Jakob Nielsen http://www.acm.org/pubs/cacm/AUG96/antimac.htm