Benutzungsschnittstellen in der Software–Technik

Was macht eine gute Benutzungsschnittstelle aus?

Dipl.-Informatiker Adriano Gesué

Entwurf von Benutzungsschnittstellen

Bei der Einführung eines neuen Software-Systems in die Praxis kommt es oft zu Problemen, da die Benutzungsschnittstellen nur unzureichend an die Probleme angepasst sind.
Die VDI-Richtlinie 5005 Software-Ergonomie in der Büro-Kommunikation beschreibt drei wichtige Anforderungen an Software-Systeme.

Kompetenzförderlichkeit Das Software-System soll

- *konsistent* und
- *Handlungsunterstützend*

gestaltet sein und so das Wissen des Benutzers über das Software-System steigern.

Handlungsflexibilität Das System muss alternative Lösungswege und Aufgabenstellungen unterstützen.

Aufgabenangemessenheit Das System muss erlauben, die Aufgabe gut und effizient zu erledigen.

Jede dieser Anforderungen hat konkrete Auswirkungen auf die Gestaltung der Benutzungsschnittstelle.

Konsistenz und Kompetenz

Die Interaktion zwischen dem Benutzer und der Software soll so gestaltet sein, daß der Benutzer kompetent mit den Software-Systemen umgehen kann und dadurch seine Handlungskompetenz gefördert wird.

Handlungskompetenz bedeutet, dass der Benutzer Wissen über die Software-Systeme und ihre organisatorische Einbettung erworben hat und dass er dieses Wissen auf die von ihm zu erfüllenden Aufgaben beziehen kann.

Grundsätze:

1. Benutzeroperationen sollen *konsistent gestaltet* sein
2. Benutzeroperationen sollen *die Aufgabenstellung unterstützen*

Konkrete Maßnahmen zum Steigern von Konsistenz und Kompetenz:

- Objekt-Aktivierung und -Bearbeitung *einheitlich, übersichtlich und durchschaubar* darstellen. Wenig syntaktische Fehler zulassen.
- Nur *im Kontext anwendbare Funktionen* darstellen; sinnlose Funktionen blockieren (z.B. grau darstellen)
- *Letzte Parametereinstellung* beim Aufruf einer Menüoption anzeigen (z.B. Formatieren einer Diskette)

Konkrete Maßnahmen zum Steigern von Konsistenz und Kompetenz:

•*Sicherheitsabfragen* bei Operationen mit schwerwiegenden Folgen. Folgen verdeutlichen.

•*Undo/Redo*-Funktion anbieten, d.h. alle durchgeführten Operationen sollten storniert werden können.
Auf Wunsch muß die Stornierung wieder aufgehoben werden (*Redo*).
Mindestens einstufig. Wunsch mehrstufig.

•*Inkrementelle Aufgabenbearbeitung* ermöglichen, d.h. kleine, unabhängige, nichtsequentielle Teilschritte mit jeweiliger Ergebnisrückmeldung.
Keine Operation darf in einer „Sackgasse“ enden.

•*Rückmeldungen* auf alle Benutzeroperationen. Anzeigen, ob Eingabe erwartet wird oder gerade eine Verarbeitung stattfindet. ¨
Uberdurchschnittliche Verarbeitungszeiten anzeigen (Art, Objekt, Umfang oder Dauer). Systembedingte Verzögerungen, Unterbrechungen oder Störungen explizit anzeigen.

Regelwerke für die Dialoggestaltung

•Das wichtigste Hilfsmittel, einheitliche Programmbedienung zu erhalten, sind *Regelwerke* (*style guides*) für die Oberflächengestaltung.

•Style Guides können bis zu hundert Seiten umfassen!

•Verbreitete Stile und Regelwerke der Dialoggestaltung sind
- •*Windows* (Windows Interface Application Design Guide)
- •*Motif* (OSF/Motif Style Guide)
- •*MacOS* (Macintosh Interface Guidelines)

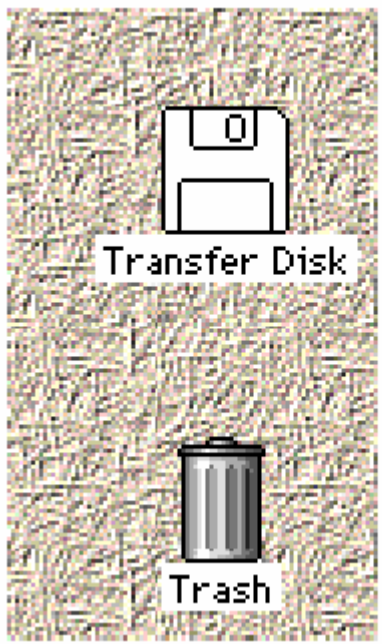
•Manche Betriebssysteme (z.B. X Window System/UNIX), Programme (z.B. StarOffice), und Bibliotheken (z.B. Swing, Qt) unterstützen mehrere Stile – entweder wahlweise oder gleichzeitig.
Generell nähern sich Aussehen und Verhalten der Oberflächen (*look and feel*) zunehmend einander an.

Beispiele für inkonsistente Benutzeroperationen

Den Sinn einheitlicher Regeln für Benutzungsschnittstellen erkennt man am besten, wenn man *Verstöße* betrachtet.

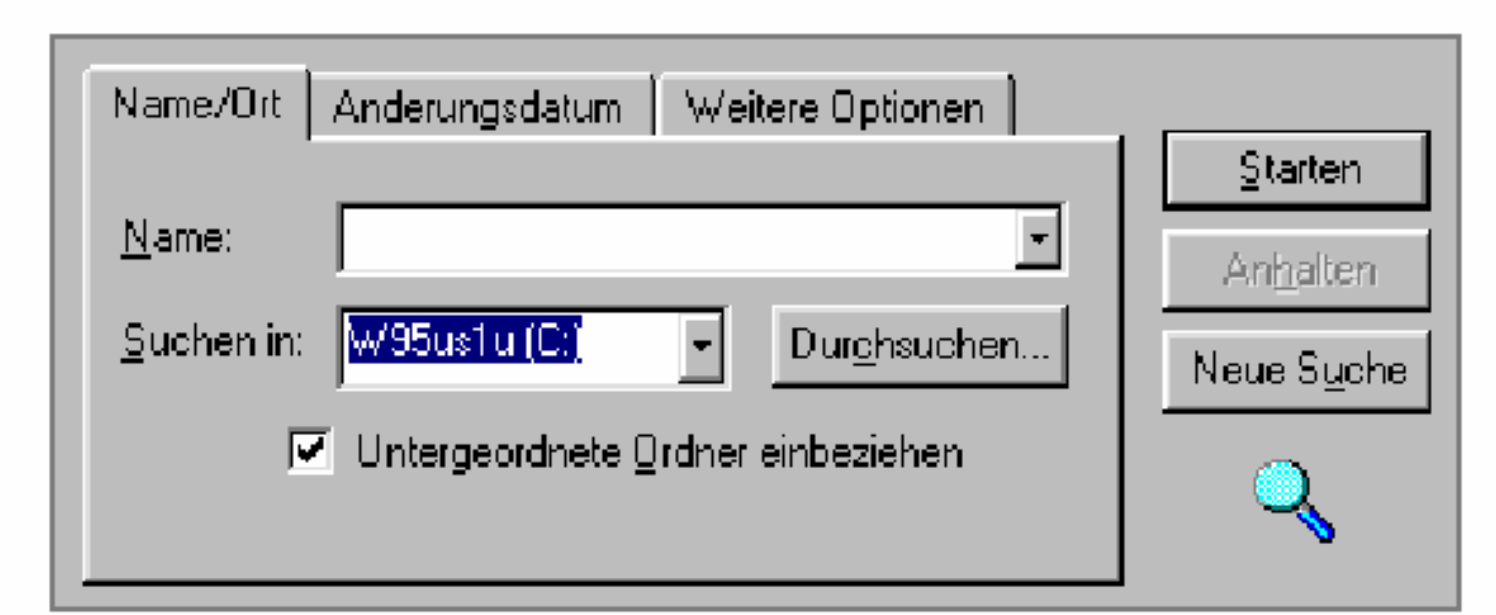
Auf einem *Macintosh*-Mülleimer kann man beliebige Objekte löschen, indem man sie auf den Mülleimer zieht. Zieht man jedoch eine Diskette auf den Mülleimer, wird sie nicht gelöscht, sondern ausgeworfen.

Ein ”magischer“ Mülleimer, der neue Benutzer verwirrt.



Frage für Macintosh-Experten: Wie lösche ich eine Diskette?

Will man in *Windows95* eine Datei suchen, erhält man diesen Dialog:



Originellerweise wird die Suche mit *Starten* aktiviert – der *Suchen*-Knopf erlaubt nur die Auswahl eines Startverzeichnisses.

Das *IBM RealPhone*, ein Telefonprogramm, kommt ganz ohne Standard-Bedienungselemente aus:

Hier ein paar einfache Aufgaben:

- Wie wähle ich eine Nummer über die Tastatur?
- Wie programmiere ich eine der zehn Kurzwahltasten?
- Was muß ich tun, um mehr als zehn Kurzwahltasten zu erhalten?
- Wie hebe ich den Hörer ab?
- Was macht eine Schublade in einem Telefon?
- Was passiert, wenn ich aus Versehen auf das *RealPhone* klicke?

In der wirklichen Welt mag es wichtig sein, anders auszusehen. Was Bedienung angeht, ist das Befolgen von Standards der richtige Weg.

Kompetenzfördernde Dialoge

Die DIN-Norm 66324, Teil 8 führt zum Thema *Kompetenzförderlichkeit* besonders auf:

- **Selbstbeschreibungsfähigkeit** Jeder Dialogschritt muß verständlich sein.

- **Erwartungskonformität** Dialoge sollen den Erwartungen der Benutzer entsprechen

- **Fehlerrobustheit** Dialoge sollen robust mit Fehleingaben umgehen.

Selbsterklärende Dialoge

Selbsterklärende Dialoge steigern die Handlungskompetenz, indem sie das Wissen über das Software-System fördern.

Ein Dialog ist selbsterklärend, wenn

- dem Benutzer auf Verlangen Einsatzzweck sowie Leistungsumfang des Dialogsystems erläutert werden können und wenn
- jeder einzelne Dialogschritt
- unmittelbar verständlich ist oder
- der Benutzer auf Verlangen dem jeweiligen Dialogschritt entsprechende Erläuterungen erhalten kann.

Konkrete Maßnahmen für selbsterklärende Dialoge:

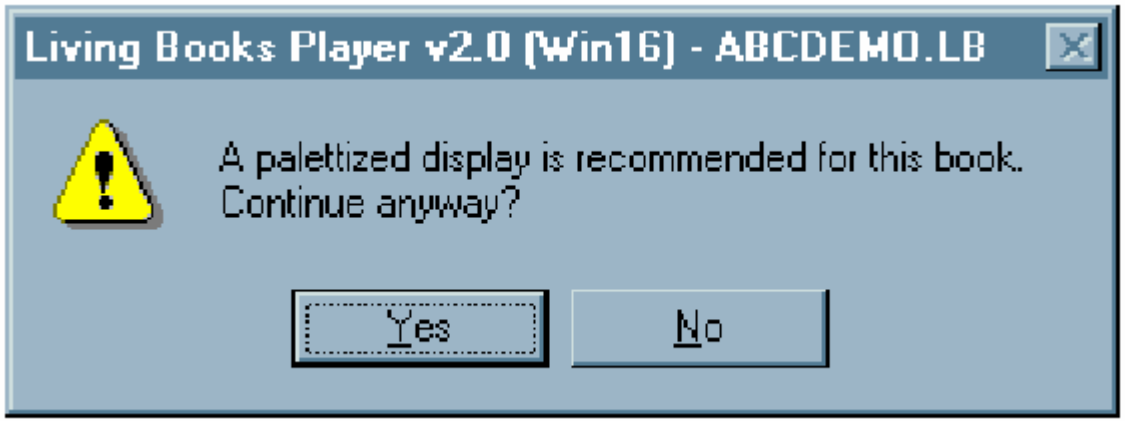
- Der Benutzer muß sich zweckmäßige Vorstellungen von den Systemzusammenhängen machen können

- Erläuterungen sind an allgemein übliche Kenntnisse der zu erwartenden Benutzer angepasst (deutsche Sprache, berufliche Fachausdrücke)

- Wahl zwischen kurzen und ausführlichen Erläuterungen (Art, Umfang)

- Kontextabhängige Erläuterungen

Beispiele für schlechte Erläuterungen

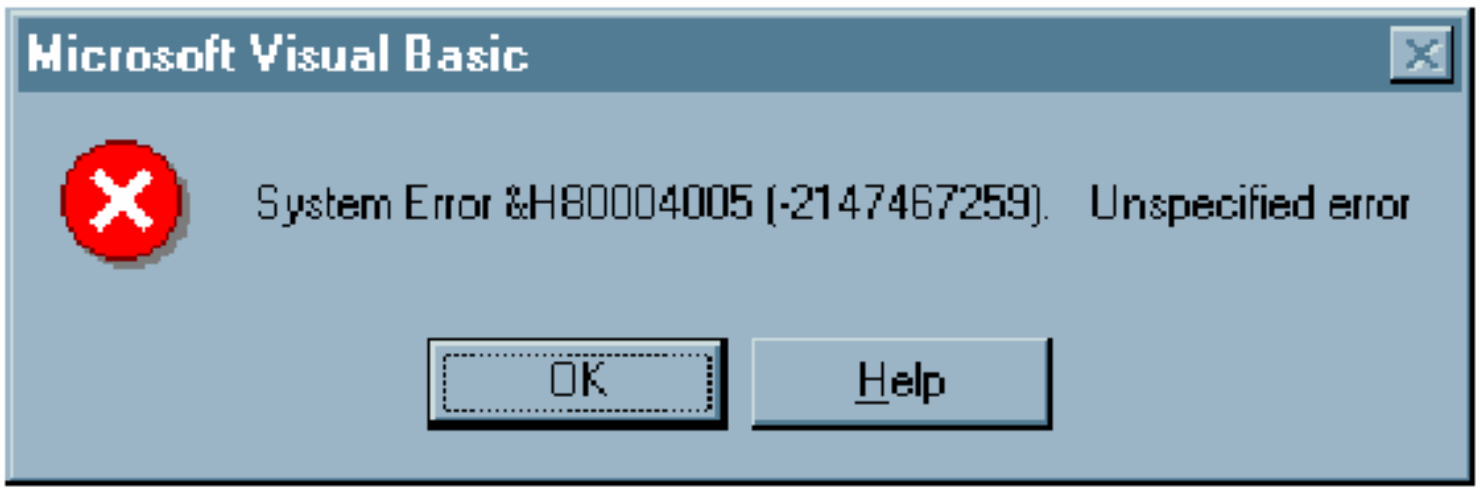


Was ist ein „palettized display“?

Ein PC-Experte mag diese Meldung vielleicht verstehen. Diese Warnung entstammt aber *Dr. Zeuss’s ABC*, ein Alphabet-Lern-Programm für *3- bis 5jährige Kinder*.

Noch bemerkenswerter: Die Warnung ist völlig überflüssig, denn auch ohne ”palettized display“ funktioniert das Programm einwandfrei.

Diese höchst aussagekräftige Fehlermeldung ist Microsoft *Visual Basic 5.0* zu entnehmen:



Nach dem Klicken auf *Help* erhalten wir:

Visual Basic encountered an error that was generated by the system or an external
component and no other useful information was returned.
The specified error number is returned by the system or external component (usually from an Application Interface call) and is displayed in hexadecimal and decimal format.

Das bedeutet:

Irgendetwas ist passiert. Wir wissen nicht, was hier passiert ist oder warum es passiert ist. Wir wissen nur, dass die dargestellte hexadezimale Zahl eine hexadezimale Zahl ist, aber die Zahl selbst hat keine Bedeutung.

Erwartungskonforme Dialoge

Die Handlungskompetenz wird durch *erwartungskonforme Dialoge* unterstützt.

Ein Dialog ist erwartungskonform, wenn er den Erwartungen der Benutzer entspricht,

- die sie aus Erfahrungen mit bisherigen Arbeitsabläufen oder aus der Benutzerschulung mitbringen sowie

- den Erwartungen, die sie sich während der Benutzung des Dialogsystems und im Umgang mit dem Benutzerhandbuch bilden.

Konkrete Maßnahmen für erwartungskonforme Dialoge:

1. Das Dialogverhalten ist einheitlich (z.B. konsistente Anordnung der Bedienungselemente).

2. Bei ähnlichen Arbeitsaufgaben ist der Dialog einheitlich gestaltet (z.B. Standard-Dialoge zum Öffnen oder Drucken von Dateien)

3. Zustandsänderungen des Systems, die für die Dialogführung relevant sind, werden dem Benutzer mitgeteilt.

4. Eingaben in Kurzform werden im Klartext bestätigt.

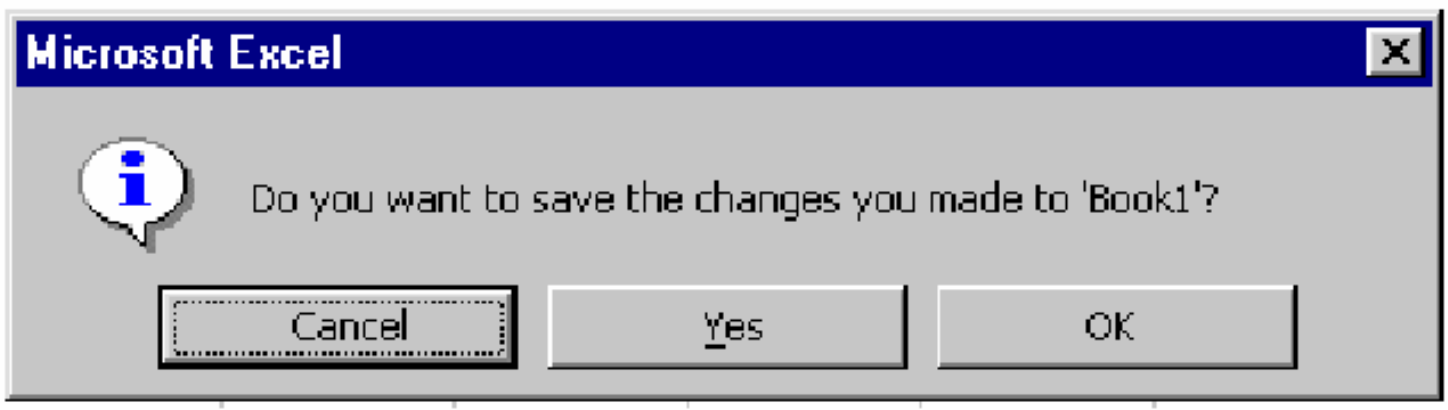
5. Systemantwortzeiten sind den Erwartungen des Benutzers angepaßt, sonst erfolgt eine Meldung.

6. Der Benutzer wird über den Stand der Bearbeitung informiert.

Beispiele für unerwartetes Dialogverhalten

Gewöhnlich schreiben Standards vor, wie Bedienungselemente anzuordnen sind – etwa *Bestätigen* vor *Abbrechen*, oder *Ja* vor *Nein*, oder *Hilfe* ganz rechts.

Microsoft Excel 95 macht alles anders:
Wo ist der Unterschied zwischen *Yes* und *OK*?



Wenn man *Freeloader*, einen WWW-Browser deinstalliert, er scheint dieses Fenster:

Der Benutzer mag denken, er könne auswählen, was denn nun tatsächlich deinstalliert werden soll. Weit gefehlt! Dies ist eine *Statusmeldung*, die anzeigt, was bereits deinstalliert wurde.

Originellerweise ändern die Knöpfe ihren Zustand, wenn man auf sie klickt – aber diese Änderung hat keine Auswirkungen.

- Ein Dialog ist *fehlerrobust*, wenn trotz erkennbar fehlerhafter Eingaben das beabsichtigte Arbeitsergebnis mit minimalem oder ohne Korrekturaufwand erreicht wird.

- Dem Benutzer müssen Fehler verständlich gemacht werden, damit er sie beheben kann.

Konkrete Maßnahmen für fehlerrobuste Dialoge:

- Benutzereingaben dürfen nicht zu Systemabstürzen oder undefinierten Systemzuständen führen.

•Automatisch korrigierbare Fehler können korrigiert werden. Der Benutzer muß hierüber informiert werden.

•Die automatische Korrektur ist abschaltbar.

•Korrekturalternativen für Fehler werden dem Benutzer angezeigt.

•Fehlermeldungen weisen auf den Ort des Fehlers hin. z.B. durch Markierung der Fehlerstelle.

•Fehlermeldungen sind
-verständlich,
-sachlich und
-Konstruktiv

•zu formulieren und sind einheitlich zu strukturieren (z.B. Fehlerart, Fehlerursache, Fehlerbehebung).

- **Beispiele für schlechte Fehlermeldungen**

Diese Meldung erscheint in IBM's *Aptiva Communication Center*, wenn der Benutzer einen leeren Datensatz ausgewählt hat und auf den *Change*-Knopf klickt:

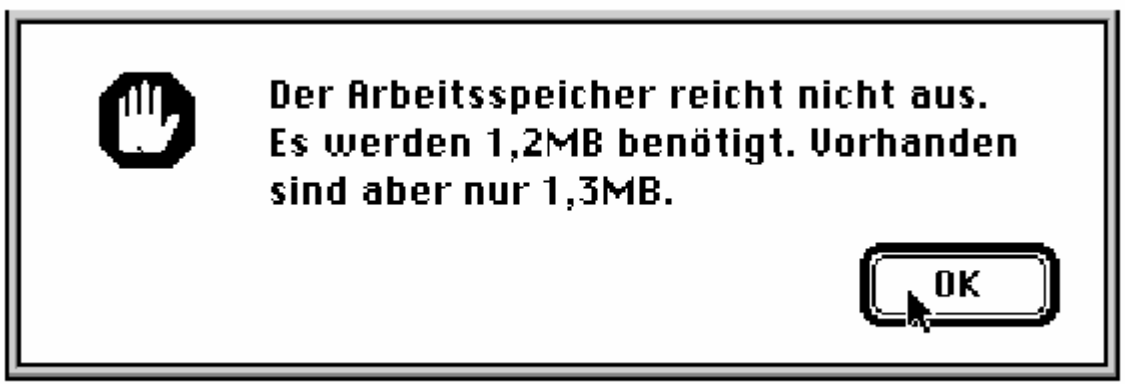
Nun gut, dies ist ein Fehler. Vielleicht auch ein dummer Fehler, aber wenn Rechner dumme Fehler ermöglichen, machen Benutzer sie auch. Was Benutzer jedoch nicht erwarten, sind unhöfliche Rechner. Diese Fehlermeldung könnte genauso gut lauten:

Ändern?! Wie dumm sind Sie eigentlich?
Was zum Teufel soll ich hier ändern?!

Eine sehr viel einfachere Lösung:

Niemals eine Funktion anbieten, die in einer Fehlermeldung endet

Stattdessen Funktionen *ausblenden*, die nicht angewandt werden können.
Zum Abschluß eine freundliche und aussagekräftige *Macintosh*-Meldung:



Flexibilität

Eine Anwendung ist dann *flexibel*, wenn

- der Benutzer mit einer *geänderten Aufgabenstellung* seine Arbeit noch effizient mit demselben System erledigen kann,
- eine Aufgabe auf alternativen Wegen ausgeführt werden kann, die dem Benutzer entsprechend seinem *wechselnden Kenntnisstand* und seiner aktuellen Leistungsfähigkeit wählen kann,
- *unterschiedliche Benutzer* mit unterschiedlichem Erfahrungshintergrund ihre Aufgaben auf alternativen Wegen erledigen können.

Konkrete Maßnahmen zum Steigern der Flexibilität:

- *Makrobildung* ermöglichen, d.h. Operationen bei wiederkehrenden Abläufen können zu einer einzigen Operation zusammengefaßt werden.
- *Mengenbildung* ermöglichen, d.h. Objekte, auf die die gleichen Operationen angewendet werden sollen, können zu größeren Einheiten zusammengefaßt werden (Beispiel: im Zeichenprogramm mehrere Objekte gruppieren und dann verschieben)
- Soweit wie möglich *nicht-modale Dialoge* verwenden.

Effizienz und Angemessenheit

Der Benutzer soll seine Arbeitsaufgabe mit Hilfe der Anwendungen in einer Weise bearbeiten,
die der Aufgabe angemessen ist:

- Kann der Benutzer die Zielsetzung seiner Aufgabe überhaupt mit dem System erreichen, oder muß er zusätzlich andere Systeme oder Medien einsetzen (z.B. Speicherung von Zwischenergebnissen auf Papier)

- Mit welchem Planungs- und Zeitaufwand (einschließlich des Aufwandes zur Korrektur von Fehlern) sowie mit welcher Qualität des Arbeitsergebnisses kann dieses Ziel erreicht werden?

Ein System ist immer dann *nicht* aufgaben angemessen, wenn

- bestimmte erforderliche Funktionalitäten nicht vorhanden sind (*fehlende Funktionalität*) oder

- der Benutzer zur Ausführung eines in seinem Verständnis zusammenhängenden und einheitlichen Arbeitsschrittes eine größere Anzahl von Interaktionsschritten benötigt“ (*geringe Effizienz der Mensch-Rechner-Interaktion*).

Konkrete Maßnahmen zum Steigern der Effizienz:

•*Minimierung der Interaktionsschritte*, die zur Ausführung einer Aufgabe oder einer einzelnen Operation benötigt werden.

Hilfreiche Techniken:
•Mnemonische Auswahl von Menüoptionen über die Tastatur

•Auswahl über Tastaturkürzel (z.B. *Strg+X* statt *Bearbeiten***0** *Ausschneiden*)

•Symbolbalken (Toolbar) mit häufig benutzten Funktionen

•Aufführung der zuletzt benutzten Objekte / Einstellungen

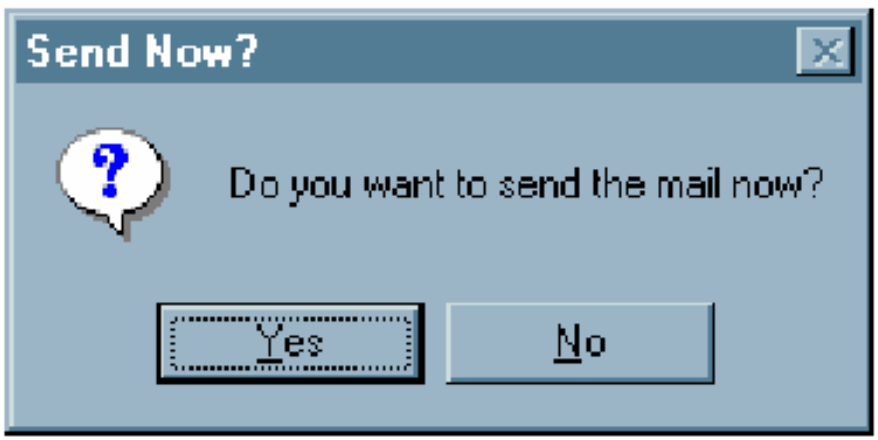
•Kommandosprache (ggf. mit Kontrollstrukturen) _ *Planungsaufwand reduzieren*, z.B. durch syntaktisch einfache Aufgaben oder Reduktion vieler Einzelschritte

•*Makro- und Mengenbildung* (s. Abschnitt 7.3) _ *Syntaktische Fehler* verhindern oder abfangen. ¨ Uberflüssige Systemmeldungen, die der Benutzer quittieren muß, vermeiden.

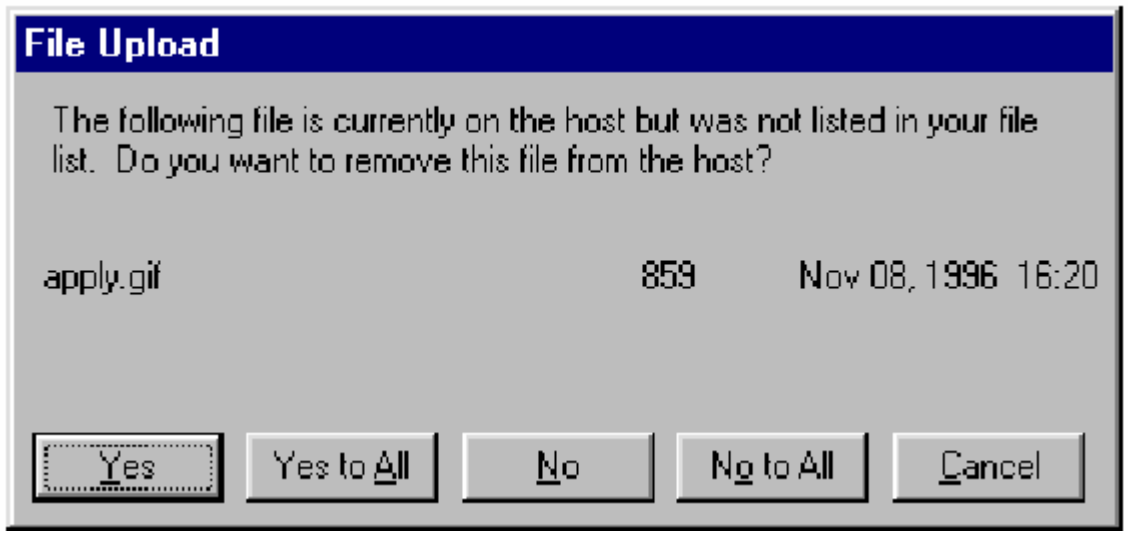
Am besten ist die volle Funktionalität eines Menüsystems und einer Kommandosprache!

Beispiele für mangelnde Effizienz

Jedesmal, wenn man in *Lotus Notes* eine e-mail absenden will, verlangt Notes eine Bestätigung.



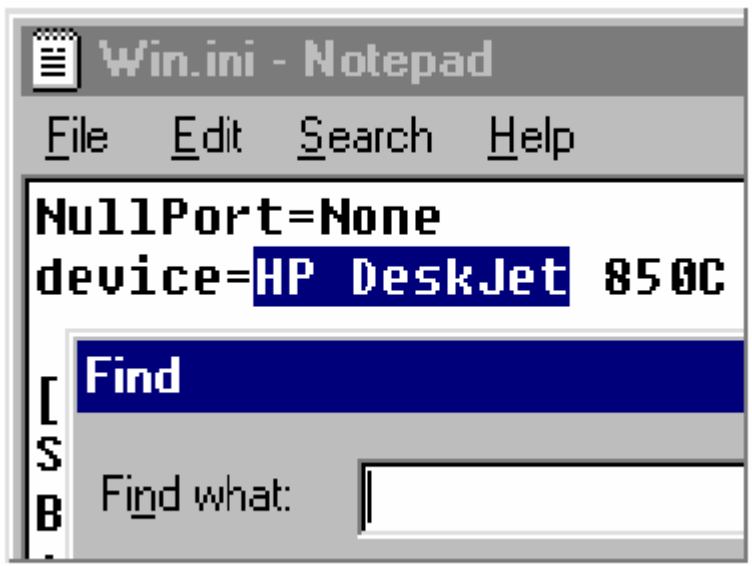
Microsoft's *Web Wizard* ist ein Programm zum Verwalten von Dateien auf WWW-Servern. Während des Ablaufs erstellt Web Wizard eine Liste der Dateien auf dem Server und fragt ab, für jede Datei einzeln, ob sie gelöscht werden soll:



Wenn man etwa die Datei *zeller.gif* löschen wollte, müsste man zunächst 400mal auf *No* klicken

für jede der Dateien, die im Alphabet davor stehen.
Sehr viel effizienter wäre es hier, das Programm würde einfach die komplette Liste der Dateien anzeigen und Löschen von Gruppen anbieten.

Wenn man in *Visual Basic* einen Text selektiert und dann *Search* aufruft, wird der selektierte Text zur Vorgabe. So sollte es sein.
In anderen Programmen wie etwa *Notepad* muß der Benutzer stattdessen aufwendig den Text über die Zwischenablage kopieren und einfügen:



Benutzerfreundliche Web-Seiten

Die Top 10, um die Benutzungsfreundlichkeit zu erhöhen– von Jakob Nielsens Alertbox:

1. Name und Logo auf alle Seiten
2. Suchfunktion (bei > 100 Seiten)
3. Aussagekräftige Titelzeilen
4. Vertikales Scannen ermöglichen
5. Information per Hypertext strukturieren (statt Scrollen)
6. Kleine Photos benutzen
7. Übertragungsgeschwindigkeit maximieren
8. Links mit Titeln versehen
9. Seiten für Behinderte zugänglich machen
10. Konventionen von anderen Seiten übernehmen

. . . und umgekehrt die Top 10, um die Benutzungsfreundlichkeit zu *verringern:*

1. Frames
2. Neueste Plug-Ins
3. Scrollender Text
4. Lange URLs
5. Waisenseiten (= Error 404)
6. Scrollende Navigations-Seiten
7. Keine Unterstützung für Navigation
8. Eigene Farben für Links
9. Obsoleter Inhalt
10. Lange Ladezeiten

http://www.useit.com/alertbox/

Checkliste: Benutzungsoberfläche

Ist die Oberfläche konsistent gestaltet?

Sind die Dialoge selbsterklärend?

Sind die Dialoge erwartungskonform?

Sind die Dialoge fehlerrobust?

Ist die Anwendung flexibel?

Unterstutzt die Oberfläche effizientes Arbeiten?