Leiden und Freuden des unbekannten GUI-Benutzers

Karol Frühauf und Ruedi Schild
INFOGEM AG, Postfach, CH-5401 Baden
Tel.: 056 222 65 32, Fax: 056 222 00 38
E-Mail: karol_fruehauf@infogem.ch, ruedi_schild@infogem.ch

Zusammenfassung

Graphische Schnittstellen erleichtern den Computerbenutzern das Leben wesentlich, indem sie die Eigenheiten des zugrundeliegenden Betriebssystems weitgehend verstecken und dem Benutzer eine mehr oder weniger einheitliche Arbeitsoberfläche zur Verfügung stellen, ob es sich nun um ein Text-verarbeitungssystem, eine Programmierumgebung oder eine Datenbank-anwendung handle. Allerdings sind diese Schnittstellen, wie so vieles in der Informatik, "organisch" entstanden und gewachsen, und nicht immer haben deren Entwickler die Prinzipien des guten Schnittstellenentwurfs beachtet oder überhaupt gekannt.

Dieser Artikel beschreibt einige der weit herum bekannten Leiden und Frustrationen im Umgang mit graphischen Schnittstellen, erläutert D. Normans vier Prinzipien des Schnittstellenentwurfs und zeigt, wie die Beachtung dieser Prinzipien die gängigen Risiken vermeiden oder doch vermindern kann.

Einleitung

In grauer Vorzeit hatten nur Spezialisten Zugang zu elektronischen Rechnern, und sie kommunizierten mit ihnen auf gar wunderliche Weise: Zunächst stanzten sie mit Hilfe eines Geräts, welches etwa die Ausmasse eines ganzen Arbeitstisches und ungefähr dessen doppeltes Gewicht aufwies, rechteckige Löcher in längliche Kärtchen aus Halbkarton, welche dann an einem Schalter abgegeben wurden. Nach einer bestimmten Zeit — Grössenordnung eine bis drei Stunden — konnte das Kartenpaket an einem Regal wieder abgeholt werden, zusammen mit einem Stapel Endlospapier, auf welchem die Resultate ausgedruckt waren.

Der heutigen Mensch-Computer-Kommunikation ähnlich war die Tastatur des Lochstanzers und der Ausdruck auf Papier, wobei man allerdings auf Kleinbuchstaben verzichten musste. Es existierten wohl auch andere Ein- und Ausgabemöglichkeiten (wie Bildeingabe, Zeichnungsausgabe etc.), aber sie waren bei weitem nicht an jedem Rechenzentrum verfügbar.

Heute stehen auch dem Nichtspezialisten vollgraphische Schnittstellen zur Verfügung, und die Zeiten zwischen Abgabe eines Programms und Anzeige der Resultate bewegen sich im Sekundenbereich.

Ist die Arbeit mit dem Computer damit komfortabler geworden? Ohne Zweifel ja. Wir heutigen Computer-Benutzer, ob als Entwickler oder als Anwender, können unvergleichlich viel mehr in viel kürzerer Zeit und mit wesentlich kleineren Kosten erreichen als die Spezialisten der grauen Vorzeit. Ist aber diese Schnittstelle wirklich so gut, wie sie sein könnte, wie sie auch sein sollte? Da können nun Zweifel aufkommen.

Mit den Möglichkeiten, welche graphische Benutzeroberflächen den Benutzern von Betriebssystemen bieten und welche die Entwickler von Anwendungen unter Benutzung der Programmierschnittstelle zur Verfügung stellen, gewinnen altbekannte aber dennoch immer noch vernachlässigte Erkenntnisse an Bedeutung.

Eine benutzerfreundliche Schnittstelle muss dem üblichen Verhalten des Bedieners Rechnung tragen. Wie wir alle schmerzlich wissen, tun wir Menschen nicht immer das Richtige. Hierfür gibt es mehrere Gründe:

- wir schnitzern, machen Flüchtigkeitsfehler
  Bei Routinetätigkeiten entwickeln wir ein "automatisches" Verhalten. Unsere Handlungen werden aus dem Unterbewusstsein eingeleitet und gehen unterwegs aus unerfindlichen Gründen in die Irre. Ein Beispiel hierfür hat jeder schon erlebt:

Bediener:	*Lösche "Wichtigste_Datei".*
System:	*Wollen Sie Ihre "Wichtigste_Datei" löschen?*
Bediener:	*Ja.*
System:	*Sind sie sicher, dass sie Ihre "Wichtigste_Datei" löschen wollen?*
Bediener:	*Jaaaaa!*
System:	*Ihre "Wichtigste_Datei" ist gelöscht.*
Bediener:	*Auuuuu!*

- wir erliegen Missverständnissen, machen aus Versehen das Falsche
  Bei komplizierteren Handlungssträngen ziehen wir an sich korrekte Schlüsse auf der Basis partieller oder gar fehlerhafter Evidenz, was uns in der Folge zu falschen Handlungen führt.
  Ein Beispiel hierzu kommt vielleicht auch einigen bekannt vor:

  Es ist Freitag Abend, Sie haben gerade eine wichtige Arbeit beendet und wollen jetzt Ihre Datei speichern. Die folgende Meldung erscheint:

Abbildung 1: Solche Systemmeldungen führen of zu unüberlegten, aber folgenschweren Handlungen.

Sie haben schon lange nicht mehr aufgeräumt auf Ihrer Disk. Jetzt ist allerdings nicht der richtige Zeitpunkt dazu, aber immerhin, einige Dateien gibt es sicher, die man löschen könnte, z.B. M_PROTO.DOC. Das ist über zwei Jahre alt und wäre gross genug, wahrscheinlich das Protokoll irgendeiner Monstersitzung. Sie löschen es, speichern Ihre Arbeit und gehen wohlgemut ins Wochenende. Am Montag fragt Ihr Chef nach der Dokumentation für den Prototypen, den Ihre Abteilung vor etwa zwei Jahren produziert hat.

- wir erinnern uns, wir assoziieren
  Wir vertrauen zu voreilig darauf, dass der gleiche Bezeichner oder das gleiche Symbol mit dem gleichen Bedeutungsinhalt (Handlung) hinterlegt ist, wie wir es gewöhnt sind (aus anderen Lebensbereichen, von anderen Programmen her).

  Ein Beispiel, das noch aus der Zeit der nichtgraphischen Texteditoren stammt: Ein Kursteilnehmer hatte während der letzten zwei Stunden ein Übungsprogramm eingetippt. Dessen Struktur hatte er selbstverständlich vorher mit Papier und Bleistift entworfen, die Details ebenso natürlich direkt am Bildschirm eingegeben. Zum Speichern und Verlassen des Editors, den er normalerweise benutzte, verwendete er den Befehl "quit". Der Editor, der im Kurs benutzt wurde, erwartete für diese Aktion "exit"; "quit" bedeutete für ihn "nichts abspeichern und Editor verlassen". Zwei Sekunden nach der Eingabe des Befehls wussten alle im Raum, was passiert war.

- wir sind vergesslich und faul
  Bei der Vielzahl der gebotenen Möglichkeiten ist es kaum verwunderlich, dass unser Erinnerungsvermögen überfordert wird. Statt nachzuschauen oder Hilfe zu suchen, tendieren viele Menschen zum Ausprobieren. Dies ist unbedenklich, solange man keinen Schaden anrichten kann. Meldet jedoch der Befehl "Rückgängigmachen" zurück, dass die Handlung zu komplex war und leider nicht rückgängig gemacht werden kann, dann wird das Ausprobieren zum Leichtsinn.

Die folgenden Prinzipien sollen helfen, mit diesen unseren Unzulänglichkeiten leben zu können.

Die vier Prinzipien benutzerfreundlicher Schnittstellen

In seinem Buch *The Psychology of Everyday Things* definiert Donald Norman vier Grundprinzipien für eine benutzerfreundliche Schnittstelle von alltäglichen Dingen. Diese Prinzipien sind abgeleitet von Beobachtungen an so einfachen Dingen wie Wasserhahn, Tür, Telefonapparat etc., gelten aber in noch grösserem Masse für kompliziertere Geräte oder Systeme wie Videorecorder, Automobile, oder eben auch Benutzerschnittstellen von Betriebssystemen und Anwendungsprogrammen.

1. Gutes mentales Modell (conceptual model)

Die Präsentation gegenüber dem Bediener ist konsistent und kohärent, der Bediener kann sich "ein Bild davon machen, was geschieht".

Ein gutes mentales Modell erlaubt uns, die Wirkung unserer Handlungen vorherzusagen. Ohne ein gutes Modell arbeiten wir blind; wir tun, was man uns gesagt hat, wir müssten es tun, ohne zu verstehen warum, ohne zu wissen, welche Wirkung wir zu erwarten haben oder was zu tun ist, wenn etwas nicht richtig funktioniert. Solange alles gut geht, schaffen wir es gerade noch. Sobald aber etwas schief läuft, finden wir uns in einer Situation wieder, in der ein tieferes Verständnis notwendig ist, eben ein gutes Modell.

Menschen bilden sich mentale Modelle durch Erfahrung, Anwendung oder Schulung. Das mentale Modell eines Geräts wird hauptsächlich durch Interpretation des beobachteten Verhaltens des Geräts und seiner sichtbaren Struktur gebildet, durch "das Erscheinungsbild" des Geräts. Wenn dieses "Bild"

inkohärent oder unangemessen ist, kann der Mensch das Gerät nur mit Mühe benutzen. Wenn "das Bild" unvollständig oder gar widersprüchlich ist, dann wird er Probleme haben.

2. Sichtbarkeit (visibility)

Der aktuelle Zustand und die möglichen Handlungen sind für den Bediener immer ersichtlich.

Je grösser die Anzahl der möglichen Handlungen ist, verglichen mit der Anzahl der zur Verfügung stehender Bedienelemente, um so wahrscheinlicher ist es, dass der Benutzer Schwierigkeiten mit der Bedienung hat. Wenn die Anzahl Bedienelemente gleich der Anzahl der Gerätefunktionen ist, kann jedes Bedienelement spezialisiert sein und entsprechend gestaltet oder angeschrieben werden. Die möglichen Funktionen sind sichtbar, jede korrespondiert mit einem Bedienelement. Vergisst der Benutzer die Funktion, so wird er durch das Bedienelement an sie erinnert.

Wenn es weniger Bedienelemente als Funktionen gibt, wird es schwierig bis unmöglich die Bedienelemente sinnvoll anzuschreiben. Es gibt nichts, was uns an die Funktion erinnern würde. Die Funktionen sind unsichtbar, versteckt. Die Bedienung wird zu einem Mysterium.

Sichtbarkeit ist eine Gedächtnisstütze: "Das kann gemacht werden, und so geht man vor." Eine gute Beziehung zwischen den Bedienelementen und ihren Wirkungen vereinfacht das Finden des richtigen Bedienelements für die Aufgabe. Es gibt wenig, was man sich merken muss.

3. Zuordnungen (mappings)

Der Bediener kann Handlung und Resultat, Bedienelemente und ihre Wirkung, Zustand und das Sichtbare immer zuordnen.

Zuordnung bedeutet hier die Beziehung zwischen dem Bedienelement mit seiner Betätigung und dem Resultat dieser Handlung. Natürliche Zuordnungen, d.h. Ausnutzen von physikalischen Analogien und kulturellen Normen, führen zum unmittelbaren Verständnis.

4. Rückkopplung (feedback)

Der Bediener erhält vollständige und kontinuierliche Information über die Auswirkungen seiner Handlung.

Den Benutzer informieren, welche Handlung er ausgelöst hat und welche Ergebnisse diese Handlung zeitigt, ist eine unabdingbare Voraussetzung für stressfreie Bedienung eines Geräts. Es reicht natürlich nicht aus, irgendwie zu informieren; die Information muss vom Bediener verstanden werden, und er muss anhand dieser Information das Ergebnis seiner Handlung an seinen Absichten messen und gegebenenfalls eine Korrekturhandlung einleiten können.

Das Betriebssystem als alltägliches Ding

Mittlerweile sind Rechner und damit Betriebssysteme zu alltäglichen Dingen geworden. Die Einfachheit ihrer Schnittstelle stellt einen Wirtschaftsfaktor dar. Bereits ein Unterschied von einer Stunde für das Erlernen der Bedienung eines Betriebssystems hat bei 100 Millionen Benutzern eine immense Auswirkung auf die Einführungskosten. Wenn man für eine Handlung, die man täglich hunderte von Malen verrichtet, wegen der ungünstig gewählten Bedienung nur eine Sekunde länger braucht, läppert sich auch einiges an unnötigen Kosten zusammen. Benutzerfreundlichkeit hat offensichtlich nicht nur mit dem Gefühl zu tun, angenehm, ohne dauernde Adrenalinschübe mit einem Ding arbeiten zu können, sondern auch mit harten wirtschaftlichen Auswirkungen. Es lohnt sich also zu untersuchen, was diese vier einfachen Prinzipen für ein Betriebssystem bedeuten.

1. Gutes mentales Modell

Der Schlüssel heisst Konsistenz. In allem: Bei Bezeichnern, bei der Wahl von Farben und Symbolen, bei der Wahl der Art der Bedienung. Der Vorteil von GUIs — die unzähligen Möglichkeiten etwas zu tun — kann sehr schnell zu einem Nachteil werden, wenn die Bedienung keinen für den Benutzer erkennbaren Grundregeln gehorcht. Bei einem überlegten Design reicht es, wenn der Benutzer die 20 bis 30 Grundregeln erkennt oder erlernt; er ist damit in der Lage, "normale" Arbeiten ohne Zögern zu verrichten.

Es ist aber schwierig, dem Benutzer plausibel zu machen, dass es notwendig ist,

- zum Beenden einer Anwendung "Schliessen", bei einer anderen Anwendung aber "Verlassen" zu verwenden
- eine Auswahl das eine Mal mit Hilfe von Pull-down-Menüs, ein anderes Mal mit Hilfe einer Leiste und Seitenbalken, und ein drittes Mal mit der rechten Maustaste als Kontextmenü zu realisieren
- drei bis vier verschiedene Möglichkeiten zu haben, die identische Handlung auszulösen
- für verschiedene Teile des Betriebssystems verschiedene Konventionen bezüglich Bedienung zu befolgen
- ..... (setzen Sie hier Ihr eigenes Lieblingsthema ein)

Eine Möglichkeit, dem Bediener das Bilden eines mentalen Models zu erleichtern, ist das Verwenden einer Metapher. Die *Tischfläche* ist eine Metapher (stand den "Fenstern" Pate), der *Abfallkorb* eine andere. Die Metapher ist aber nur dann wirklich hilfreich, wenn sie vollständig anwendbar ist bzw. dort, wo eine Analogie herangezogen wird, diese vollständig zutrifft. Fred Brooks würde die "Tischfläche" lieber als *Flugzeugsitz* (Economy Class) bezeichnen, diese Metapher wird den Platzverhältnissen eher gerecht. Ein "Abfallkorb", dessen Inhalt überschrieben werden kann, ist keiner; in einem Abfallkorb sammelt man das, was man meint nicht mehr zu gebrauchen, und wenn man später unweigerlich entdeckt, dass man es doch noch braucht, ist es noch da, auch das älteste Objekt. Irgendwann wird man ihn zwar dennoch leeren, aber das ist dann eine bewusste, unabhängige Handlung — durch Einwerfen eines weiteren Zettels wird ein normaler Abfallkorb nicht seines bisherigen Inhalts beraubt!

Die sogenannten "Radio-Buttons" (Abb. 2) sind ein weiteres Beispiel für eine gute Metapher: Wenn man auf einen von ihnen drückt, schnappt der bisher gedrückte automatisch heraus.

Abbildung 2: Radio Buttons für die Auswahl einer aus mehreren Möglichkeiten.

2. Sichtbarkeit

Dieses Prinzip bei Betriebssystemen anzuwenden, ist wohl am schwierigsten. Mit 12 Funktionstasten, 13 Spezialtasten und 3 Maustasten neben der eigentlichen Tastatur kommt man nicht sehr weit. Die Menüleisten, ihre Pop-up-Menüs und die Icons helfen weiter. Wenn man nur erahnen könnte, was sich alles unter einem Menüeintrag verbirgt, und dem Abziehbildchen entnehmen könnte, welche Handlung es wohl symbolisiert (ein inhärentes Problem der Anwendung von Icons ist, dass sie wegen ihrer begrenzten Grösse besser geeignet sind, das Wesen von Dingen oder ihre Beschaffenheit darzustellen als die möglichen Handlungen mit diesen Dingen; gerade für letzteres werden sie aber zur Hauptsache eingesetzt — deshalb ist der verbale Hinweis mit einem Verb am Maus-Pfeil so hilfreich.

Gute Beispiele für Icons sind  und  ; demgegenüber wirkt  nicht besonders intuitiv.

3. Zuordnungen (mappings)

Ist das Problem der Sichtbarkeit gelöst, so bleibt noch immer das Problem der "natürlichen" Zuordnung der Bedienelemente bzw. ihrer Arten zu den Handlungen. Bei einem artifiziellen Ding wie der Computer eines ist, sind natürliche Zuordnungen fast unmöglich zu finden. Wo es keine gibt, hilft die Standardisierung. Mit der Tastatur war es einfach, da konnte man die genormte Anordnung von der Schreibmaschine erben. Bei der Maus setzte wohl der erste "Mauszüchter" den Standard. Die Funktionstasten haben der Standardisierung erfolgreicher widerstehen können; allerdings ist es nicht ratsam, F1 mit einer wichtigen, nicht rückgängigzumachenden Handlung zu belegen.

Zuordnung hat etwas mit dem Erleichtern der Orientierung zu tun. Wie es Nievergelt und Weydert in *Sites, Modes, and Trails* formulieren:

- Wo bin ich?
- Was kann ich hier tun?
- Wie bin ich hierher gekommen?
- Wohin kann ich gehen, und wie gelange ich dorthin?

Im allgemeinen sind die Angaben über den Zustand des Systems recht spärlich. Wenn nichts anderes angezeigt ist, weiss man — oder man meint zu wissen —, dass das Betriebssystem auf eine Eingabe wartet. Sonst erfährt man (ab und zu), dass das System sich im Zustand der Beschäftigung befindet (und deshalb quietscht es, wenn man es veranlassen will, etwas zu tun). Der andere, bei manchen Systemen sehr bekannte Zustand, von dem man als Benutzer nicht gerade erfreut Kenntnis nehmen darf, ist das des "Verlorenseins" im eigenen Irrgarten, zumeist ein für den Benutzer schmerzlicher Abschied des Systems von seiner Funktionsfähigkeit.

Am wenigsten Probleme bereitet die selektive, d.h. zustandsbedingte Anzeige der möglichen Handlungen. Die Konvention, dass die helleren bis unsichtbaren Einträge oder Symbole nicht betätigt werden können, ist ein weitestgehend befolgter Standard. Warum gewisse Handlungen nicht möglich sein sollen, ist jedoch für den Benutzer nicht immer leicht nachvollziehbar (und selten findet er es im On-line Help oder Benutzerhandbuch erklärt).

Kaum beachtet wird das Bedürfnis des Benutzers, zu wissen, wie er in einen gewissen Zustand geraten ist. Seinem Erinnerungsvermögen wird zu viel zugetraut. Aber ohne die Kenntnis darüber, welche Folge von Handlungen er durchlaufen hat, ist es für ihn oft unmöglich, sich selbst zu befreien oder Hilfe zum Befreien zu erhalten, sollte er in eine Sackgasse geraten sein.

Wenn ich zum Beispiel in Winword einen langen Gedankenstrich einfügen möchte, dabei aber die falsche Tastenkombination (zweihändig!) erwische, wird kein Gedankenstrich eingefügt, dafür verwandelt sich der Cursor aus dem Pfeil oder dem Zeichen  in . Rückgängigmachen nützt in diesem Fall nichts; ich kann versuchen, die Kombination nochmals einzugeben, aber selbst wenn ich sie finde, verändert sich der Cursor nicht mehr. Mit einem unguten Gefühl speichere ich die Datei ab und möchte Winword verlassen. Doch siehe, jetzt ist der Cursor wieder normal, ich kann also weiterarbeiten. Spätestens beim nächsten Speicherversuch stelle ich fest, dass jetzt der Menüpunkt "Speichern" verschwunden ist! (Wie sich ausserdem herausstellt, war auch der letzte Speicherversuch erfolglos.)

4. Rückkopplung (feedback)

Dieses eigentlich am einfachsten zu lösende Problem des Entwurfs wird häufig am meisten vernachlässigt. In Seminaren, in denen wir nach den Erfahrungen der Teilnehmer mit dem Verletzen dieser vier Prinzipien fragen, dominiert die fehlende oder mehr oder weniger unsinnige bis irreführende Rückkopplung (dicht gefolgt von Inkonsistenzen). Am Zynischsten wird es, wenn die folgende Fehlermeldung erscheint:

In Ihrer Anwendung ist ein Fehler aufgetreten.
Wenn Sie "Ignorieren" wählen, sollten Sie Ihre Arbeit in einer neuen Datei speichern.
Wenn Sie "Schliessen wählen, wird Ihre Anwendung geschlossen.

Wenn man nun "ignorieren" wählt, passiert überhaupt nichts (ich habe ja gesagt "ignorieren"!). Man kann dann insistieren, und beim zweiten Mal verschwindet die Meldung. Wenn man andererseits "Schliessen" wählt, so wird die Anwendung nicht einfach geschlossen, sondern es erscheint die informative Mitteilung

WINWORD verursachte eine allgemeine Schutzverletzung
in Modul WINWORD.EXE an Adresse 0019:7867

Jetzt darf man noch einmal "Schliessen" wählen, und die Arbeit seit dem letzten Speichern ist verloren.

Eine gut gemeinte aber nicht immer geeignete Lösung ist die unmittelbare "bitte warten" Anzeige, die unverändert stehen bleibt und dem Benutzer suggeriert, das etwas geschieht obwohl das Betriebssystem inzwischen selbst auf das Wiedererwecken wartet.

Die oben gezeigte Fehlermeldung über die Schutzverletzung (memory access violation) gehört, wie "ungültige Eingabe" (invalid input) und ähnliche, bereits zur Folklore, allerdings eine Tradition, die man tunlichst nicht mehr weiterpflegen sollte.

Das grösste Problem bereiten Situationen, in die das Betriebssystem über zu viel verschiedene Wege, aufgrund zu viel unterschiedlicher "Fehlhandlungen" des Benutzers gelangen kann. Wenn man auch nicht alle potentiellen Ursachen des Missstands aufzählen kann, ist es noch immer hilfreicher, dem Benutzer hiervon Müsterchen zu liefern, als ihm einfach mitzuteilen, dass er irgend etwas getan hat, was er nicht hätte tun sollen — das hat er ja höchstwahrscheinlich selbst schon gemerkt.

Hier könnte die Information über den Weg des Benutzers dorthin (siehe Zuordnung) genutzt werden um die Anzahl möglicher Ursachen zu mindern und so besseren Feedback geben zu können.

Schlussfolgerung

Die graue Vorzeit sehnen wir sicher nicht wieder herbei. Wir möchten auf den Komfort der graphischen Schnittstellen, trotz oder gerade wegen unserer eigenen Unzulänglichkeiten, bestimmt nicht verzichten, denn eine gut entworfene Schnittstelle kann uns helfen, Fehler zu vermeiden. Wir kommen noch einmal auf die in der Einleitung aufgeführten Beispiele für menschliches Fehlverhalten zurück und fragen: Wie können die vier erwähnten Prinzipien dabei risikovermindernd wirken?

- wir schnitzern, machen Flüchtigkeitsfehler
  Automatismen wie der beschriebene lassen sich auch durch ausgeklügelte Vorsichtsmassnahmen nicht vermeiden; der Schnittstelle kann man hier keinen Vorwurf machen. Zur Verminderung des Risikos dient in diesem Fall die Metapher des Abfallkorbes (Prinzip des mentalen Modells), wobei hier vielleicht ein eigens dafür vorgesehenes Programm ("unerase") zum Zuge kommen muss.
- wir erliegen Missverständnissen, machen aus Versehen das Falsche
  Hier liegt das Problem in der Vollständigkeit der Information (Prinzip der Sichtbarkeit). Wenn ich meiner Datei statt M_PROTO den Namen Prototyp_Melkmaschine geben kann, weiss ich eher, ob ich sie wirklich löschen möchte. Ebenso wäre mir geholfen, wenn ich auf schnelle und einfache Weise von ihrem Inhalt Kenntnis bekommen kann.
- wir erinnern uns, wir assoziieren

Wenn derselbe Befehl immer dieselbe oder mindestens analoge Wirkung hat, dann ist das Risiko der Verwendung des falschen Befehls stark vermindert (Prinzip der Zuordnung).

- wir sind vergesslich und faul
  Beim Ausprobieren ist es besonders wichtig, dass wir über die Auswirkungen unserer Tätigkeiten rasch und vollständig informiert werden (Prinzip der Rückkopplung). Aber auch das Prinzip der Sichtbarkeit, des guten mentalen Modells und der Zuordnung unterstützen uns hier, und ihr Fehlen wird oft schmerzlich bemerkt.

Eine konsequente Anwendung der vier Prinzipien des guten Schnittstellenentwurfs kann den unbekannten GUI-Benützer grösstenteils vor Leiden bewahren und ihm statt dessen viele Freuden bescheren.

Literatur

Brooks (1986)
Frederick P. Brooks, Bemerkung, gemacht anlässlich seines Vortrags "No Silver Bullet: Essence and Accidents of Software Engineering" am IFIP World Congress 1986 in Dublin. Der Vortrag ist auch erschienen in *IEEE Computer,* Vol. 20, No. 4 (April 1987), pp. 10-19.

Grams (1990)
Timm Grams, *Denkfallen und Programmierfehler*, Springer, Berlin, 1990, ISBN 3-540-52309-2.

Nievergelt (1979)
J. Nievergelt and J. Weydert, *Sites, Modes, and Trails: Telling the User of an Interactive System Where He Is, What He Can Do, and How to Get to Places*, Berichte des Instituts für Informatik der Eidg. Technischen Hochschule Zürich, Nr. 28, Januar 1979.

Norman (1988)
Donald A. Norman, *The Psychology of Everyday Things*, Basic Books, New York, 1988,
ISBN 0-465-06709-3