[RS88] Ralf Steinmetz, Gundolf Gremler: *Designkriterien einer graphisch orientierten Telefon-Benutzerschnittstelle*. In: *ITG Fachtagung "Nutzung und Technik von Kommunikationsendgeräten", Bad Nauheim*, September 1988.

Design-Kriterien einer graphisch orientierten Telefon-Benutzerschnittstelle

Ralf Steinmetz

Philips Kommunikations Industrie AG.
Business Systems International
Advanced Development
Eiserfelder Str. 316
5900 Siegen 31

Gundolf Gremler

Universität Siegen
Fachbereich Elektrotechnik
Hölderlinstr.
5900 Siegen

Kurzfassung:

Graphische Benutzeroberflächen basieren auf Window Systemen wie zum Beispiel GEM, NeWS, MS-Windows oder X-Windows. Das Design einer benutzerfreundlichen graphischen Oberfläche erfordert die Berücksichtigung vieler Bedingungen. Wesentliche Kriterien zum Design einer solchen graphischen Oberfläche (einer Window-Applikation) werden in diesem Artikel aufgeführt. Die Erfahrungen werden anhand einer bei der PKI (Philips Kommunikations Industrie AG) implementierten ISDN-Telefonapplikation dargestellt.

Schlüsselwörter:

Graphische Benutzerschnittstelle, Window Systeme, Softwareentwurfskriterien, Software Engineering

1. Einleitung

Ein Telefon wird in Zukunft integraler Bestandteil von Bürocomputern, insbesondere in ISDN-Umgebungen, sein.

Heutige Komfort-Telefone sind mit einer großen Menge von teilweise dreifachbelegten Tasten ausgestattet. Es bedarf eines nicht unerheblichen Aufwands, ein solches Telefon mit seinen Raffinessen zu beherrschen. Außerdem wird bei nur sporadischer Anwendung vieler Funktionen deren Handhabung sehr schnell wieder vergessen. Mit einem Telefon als Teil eines Bürocomputers können einige dieser wesentlichen Nachteile heutiger Benutzerschnittstellen durch die Darstellung auf einem Bildschirm unter Verwendung graphisch orientierter Resourcen vermieden werden. Das angestrebte Ziel besteht in der Realisierung einer extrem benutzerfreundlichen Bedieneroberfläche.

Obwohl das Thema der Gestaltung von Benutzeroberflächen als sehr relevant eingestuft werden muß, ist die Literatur nicht entsprechend ausführlich, sondern meist sehr abstrakt /Cormick 82/ oder zu problemspezifisch /Clemons, Greenfield 85/. Als primäre Informationsquelle gilt die SIGCHI (ACM Special Interest Group for Computer-Human Interaction) und die jährlich

stattfindende Human Factors Society Conference. Viele Ergebnisse werden aus Experimentreihen abgeleitet, siehe u.a. /Carrol, Mazur 86/. Eine gute Übersicht zur Gestaltung graphischer Oberflächen wird in /Foley, Wallace, Chan 84/ mit einer Klassifikation der Interaktionen (Selektion, Positionierung, Folgen von Selektionen und Positionierungen, Quantifizierung, Text) und der ausführbaren Aktionen gegeben. Dabei werden auch einige Designkriterien genannt. In dem vorliegenden Artikel wird ausgehend von einem konkreten Beispiel eine Reihe spezieller und allgemein gültiger Kriterien zur Konzeption der Benutzeroberfläche zusammengetragen. Einige Autoren beschäftigen sich mit Teilaspekten wie die Farbgestaltung /Murch 84/, die Lage der Fenster zueinander und deren Abmessungen /Gait 85, Gait 86/ oder der Verwendung von Icons in alternativen Window Systemen /Myers 84/.

Die Designkriterien der Telefonapplikation setzen sich zusammen aus bzw. werden beeinflußt von:

- für Window Systeme allgemein gültige Regeln zur Benutzerinterfacegestaltung,
- implementierungsbedingte Randbedingungen und Anforderungen,
- problemspezifische Randbedingungen und Anforderungen.

Sie spiegeln sich im Designprozeß wieder und lassen Folgerungen auf die Programmierumgebung zu.

Eine graphisch orientierte Oberfläche bedeutet heute das Vorhandensein und die Programmierung eines *Window Systems*. In diesem Artikel wird deshalb zuerst auf das Prinzip von Window Systemen aus Sicht des Applikationprogrammierers eingegangen. Die zur Verfügung stehenden und benötigten graphischen Objekte bilden einen Baukasten zum Design und zur Implementierung der erstrebten Graphikapplikation.

Anschließend wird ein gemeinsames Verständnis des Begriffs "Benutzerfreundlichkeit" als wesentliches Ziel der *effektiven Realisierung benutzerfreundlicher Applikationen* in Window Systemen entwickelt. Zusätzlich muß die Implementierung als solche und Wartbarkeit eines solchen Softwarepakets beachtet werden.

Im vierten Abschnitt werden einige *problemspezifische und implementierungsbezogene Anforderungen* (z.B. von Seiten der DBP) formuliert.

Während des Designprozesses fallen viele *Entscheidungen bezogen auf die graphische Benutzeroberfläche* an; diese werden erläutert und veranschaulicht.

Auf Erfahrungen bezüglich der Designkriterien der Telefonapplikation und anzuwendende Methoden wird im Ausblick eingegangen.

2. Window-Umgebung

In diesem Abschnitt wird das Window System (WS) hauptsächlich aus Sicht des Applikations-Programmierers (d.h. das Toolkit) beschrieben. Es wird nicht auf alle Einzelheiten einer Implementierung der Window-Schnittstelle (z.B. für MS-Windows) eingegangen.

Die weite Verbreitung von WSen wurde durch die kostengünstige Verfügbarkeit von Graphikschirmen inkl. Controller mit schnellen Speicherbausteinen (als Ausgabemedium) und die Maus (neben der Tastatur als Eingabegerät) ermöglicht. Mit einem WS können dem Benutzer die Multitasking-Möglichkeiten eines entsprechenden Systems zur Verfügung gestellt werden, ohne daß sich hier gleichzeitig oder quasi-gleichzeitig laufende Applikationen in der Verwendung der Ein- und Ausgaberessourcen behindern. Dabei bietet ein WS konkret die Möglichkeit, die Ressource Bildschirm auf die Ausgabe mehrerer Anwendungen aufzuteilen und dabei auch auf einer Applikation mehrere Sichten (d.h. mehrere Fenster) zu unterstützen.

Jede Applikation erhält einen virtuellen Schirm, das sogenannte "Top Level Window". Es kann somit zwischen verschiedenen Applikationen "hin- und hergeschaltet" werden, ohne die Lage des virtuellen Bildschirms zu ändern. Eine Applikation kann ihren virtuellen Bildschirm in kleinere Bereiche, sogenannte Fenster, unterteilen. Aus Sicht der einzelnen Applikationen ist wieder jedes Fenster ein virtueller Rasterbildschirm. Ein Top Level Window kann seinerseits Fenster enthalten, die wiederum Fenster enthalten, u.s.w.. Fenster gleicher Hierarchiestufe können sich überlappen, während die Kinder an den Rändern ihres Vaters abgeschnitten werden.

Eine Window-Applikation kann Fenster öffnen und schließen und kann die verschiedensten Objekte auf dem Bildschirm präsentieren. Diese Übergänge können von dem Programm selber veranlaßt werden (auch durch Timer gesteuert) oder durch Eingabeereignisse von dem Anwender. Dabei gelten bestimmte Regeln, wie z.B. daß durch das Schließen eines Fensters auch alle beinhaltene Fenster geschlossen werden (eine Ausnahme bilden machmal die Pull-Up und Drop-Down Menüs).

Als **Eingabe**gerät wird in diesem Artikel die Maus (neben einer Tastatur) betrachtet. Es können folgende Eingaben unterschieden werden:

- Taste Drücken
- Taste Loslassen
- Einzelnes Drücken und Loslassen einer Taste (single click)
- Gleichzeitiges Drücken und Loslassen mehrerer Tasten (gleichzeitiger single click)
- Zwei single clicks schnell hintereinander (double click)
- Taste gedrückt halten (push).

Diese Eingaben können dann ausgewertet werden. Hierbei ist zu beachten, daß ähnliche Operation durch denselben Eingabemechanismus ausgelöst werden sollten.

Wenn sich der Mauszeiger innerhalb eines Fensters befindet oder eine Eingabe über die Maus geschieht, dann wird eine entsprechende Meldung an dieses Fenster abgesetzt. Bei der Zuordnung der Eingaben zu den Fenstern werden diese in Abhängigkeit ihrer Lage übereinander abgefragt. Weiterhin können Fenster Nachrichten entgegennehmen und an andere Fenster schicken. Diese Meldungen repräsentieren z.B. Benutzereingaben von der Tastatur und Maus oder systeminterne Ereignisse (z.B. Timer und Interrupt). Die Verwaltung der Ereignisse sowie die entsprechende Zuteilung von Nachrichten obliegt dem WS. Das WS verwaltet die Gesamtheit aller Fenster.

Innerhalb eines Fensters können einzelne Pixel gelesen oder gesetzt werden. Zur komfortableren Gestaltung der Fensterinhalte werden **Graphik-Primitiven** als Komponenten der "Graphics Language" angeboten. Typische Graphik-Primitiven können sein:

- Bereichskopie oder Bit-Blt-Operationen
- Proportionale Zeichensätze
- Linien mit unterschiedlichen Attributen wie Linienstärke und Muster
- Kreise und Kreisbögen
- Bereiche mit Muster zum Ausfüllen
- Bitmaps, dh. Icons
- Verknüpfungen (z.B. OR, EXOR) zwischen neuen und im Fenster vorhandenen Pixelmustern
- Direktes Setzen und Lesen von Pixeln, um auch eigene Primitiven implementieren zu können

WSe arbeiten objektorientiert; das einfachste Objekt ist ein Fenster. Zwischen verschiedenen Objekten werden Meldungen ausgetauscht, auf die Objekte bzw. Fenster mit entsprechenden Methoden reagieren. Auch die Maus- und Tastatureingaben sind Meldungen. Bei einigen WSen

wird diese objektorientierte Umgebung auf einer prozeduralen Sprachumgebung (wie in MS-Windows auf Pascal) zum Teil nachgebildet.

Einige oft verwendete Kombinationen zwischen Ein- und Ausgabe stehen auf der Window-Programmier-Schnittstelle als Objekte zur Verfügung. Auf dieser höheren Schnittstelle (dem *Toolkit*) stellt das WS eine Menge graphischer Objekte (wie editierbaren Text, Buttons) zur Verfügung, die sich zu einem Objektbaum zusammensetzen (nach den Regeln der Window-Hierarchie). D.h. ein Objekt kann weitere Objekte beinhalten und kann aus diesen bestehen. Wenn z.B. ein Fenster mit einem Pull-up Menü und zwei Icons (Papierkorb und Zettelkasten) auf dem Bildschirm dargestellt wird, dann sind hier 4 Objekte sichtbar. Durch Eingaben (meistens mit einer Maus) können Aktionen ausgelöst werden, es entsteht ein Dialog.

Primitive Objekte sind u.a. Icons (Ikonen), editierbare Textstücke, Knöpfe und Schieberegler. *Icons* sind Bilder, Symbole, graphische Darstellungen. Das Icon kann

- der beschreibenden realen Aktion sehr ähnlich sein (z.B. ein Telefonbild zum Telefonieren),
- ein abstraktes Symbol dafür sein (z.B. ein Papierkorb zum Löschen von Daten) oder
- ein allgemein bekanntes beliebiges Zeichen sein (z.B. das Zeichen für Radioaktivität) /Chang 87/.

Zur korrekten Interpretation muß der Erfahrungshintergrund von dem Designer der Icons und dem Anwender sehr ähnlich sein, sonst kann die Verwendung der Icons durch den Anwender zeitintensiv, d.h. nicht intuitiv, sein. Außerden besteht die Gefahr der Doppeldeutigkeit. Ein Fenster kann in ein Icon verkleinert werden ("to iconize"). Durch Icons kann auch eine Hierar-

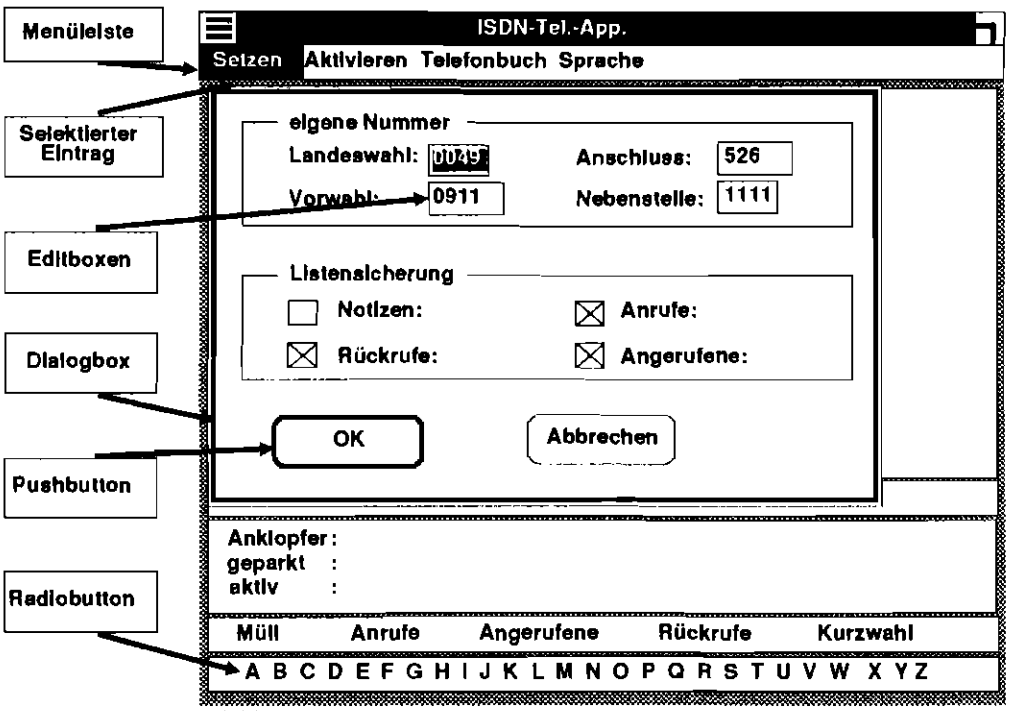


Abb. 2.1: Einige Objekte der graphischen Oberfläche

chie wie die folgende Analogie dargestellt werden:
Filesystem als Schrank -- Directory als Ordner -- Papier als Datei.

Ein *Pushbutton* ist ein Taster der zwei Zustände annehmen kann, die durch das Anwählen umgeschaltet werden. Dabei hat dieser Taster ein in gewissen Grenzen vom Programmierer veränderbares Aussehen. Auch die beiden Zustände können verschieden angezeigt werden (z.B. Kreuz in einem Pushbutton oder Invertieren des Tasters). Unter Umständen soll für den Anwender dieses Umschalten gesperrt werden, dies ist meist durch einen grauen anstatt eines schwarzen Knopfes verdeutlicht.

Der *Cyclebutton* verändert durch mehrmaliges Anwählen seinen Zustand und nur die gerade angewählte Alternative ist sichtbar.

Ein *Radiobutton* abstrahiert (wie auch der Cyclebutton) die Verknüpfung 1 aus n (n größer als 1) auf die Benutzerschnittstelle: Es wird von den n Möglichkeiten immer eine und nur eine gesetzt sein. Dabei sind alle Alternativen sichtbar. Eine "gesetzte" oder "aktivierte" Alternative wird meist invertiert dargestellt. Hiermit ist für den Programmierer durch Verwendung dieses Objekts immer gewährleistet, daß nur eine Alternative gesetzt ist. Diese kann dann abgefragt werden, bzw. bei einer Zustandsänderung wird dies signalisiert.

Höhere Objekte sind aus den graphischen Primitiven und primitiven Objekten zusammengesetzt; sie können die Semantik mehrerer Eingabemechanismen beinhalten.

Menüs werden i.allg. zur Auswahl von Kommandos eingesetzt, sie können als Einträge Text oder Bilder enthalten. Eine spezielle Variante ist der *Menübalken*. Er stellt eine Kombination zwischen Text (bzw. Bildern) und aktivierbaren Bereichen dar und befindet sich meistens am oberen Rand des Fensters. Er stellt sozusagen das Hauptmenü dar. Durch das Aktivieren eines Bereich kann entweder direkt eine Aktion ausgelöst werden, meistens erscheint ein *Drop-Down Menü* aus dem dann ein Element selektiert werden kann. Der Text oder die Graphik dieser Menüs ist meistens zur Übersetzungszeit bekannt. Eine Alternative sind die Pop-Up Menüs, die an beliebiger Stelle eines Fenster erscheinen können.

Durch sogenannte "accelerator Keys" kann ein Menüeintrag über die Tastatur direkt angewählt werden.

Texte, bzw. Daten, die selektiert werden sollen, werden über eine *Listbox* dargestellt. Die Größe dieser Box begrenzt die Anzahl der gleichzeitig auf dem Schirm erscheinenden anzuwählenden Elemente. Es wird immer ein Element auf einer Zeile dargestellt und kann so als Ganzes angesprochen werden. Falls der Bereich der Listbox zu klein für die darzustellende Informationsmenge ist, kann ein Scrollmechanismus aktiviert werden. Dies geschieht entweder durch Scrollbars (z.B. in GEM oder MS-Windows) oder durch das Bewegen der Maus während des Drückens einer Taste (in einer Smalltalk-Umgebung).

Der in einer *Editbox* dargestellte Text kann durch den Anwender verändert werden. Die Mechanismen zum Editieren und Scrollen werden durch dieses Objekt bereitgestellt.

In *Dialogboxen* sind verschiedenen Objekte zusammengefaßt. Meistens kann der Anwender zwischen mehreren Alternativen wählen und auch Texteingaben tätigen. Oft werden auch einfache Dialogboxen mit Warnungen dargestellt ("Confirmer" oder "Modal Dialogues"). Diese Confirmer können den Anwender zur Antwort auf die angezeigte Meldung zwingen (dann werden alle Eingaben abgeblockt, außer die auf diese Dialogbox bezogenen).

Die Objekte besitzen eine Menge von Attributen die durch den Anwenderprogrammierer verändert werden können (z.B. Lage, Form, Größe, Anzahl zusammenhängender Radiobuttons). Alle Attribute müssen einen Wert erhalten.

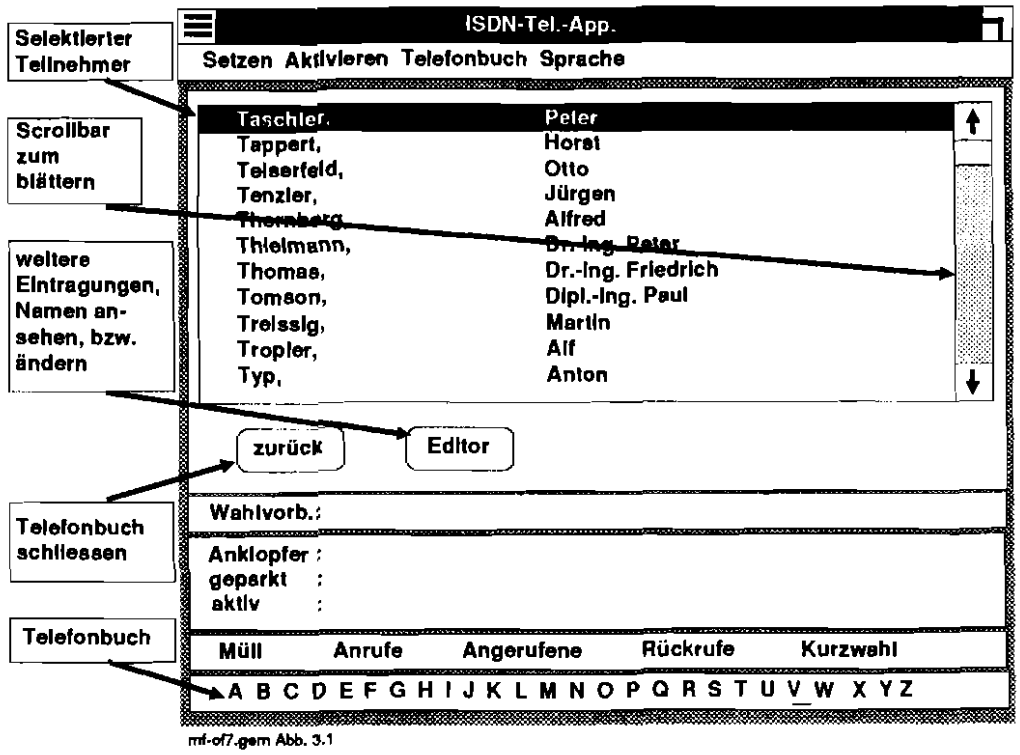
3. Einige Kriterien zur effektiven Realisierung benutzerfreundlicher Applikationen in Window Systemen

Zur Gestaltung und Realisierung einer benutzerfreundlichen graphischen Oberfläche (BGO) wird im Folgenden in erster Linie auf den Begriff "Benutzerfreundlichkeit" eingegangen. Anhand der Telefonapplikation werden die aufgeführten Kriterien verdeutlicht.

Die Bedienung der Applikation muß von verschiedenen Benutzerklassen leicht erlernbar sein. Das herkömmliche Telefon mit der Wählscheibe erfordert keine Lernzeit. Ein einfaches Tastentelefon mit Kurzwahl und Wahlwiederholung erfordert maximal 10 min. Ein ISDN-Komforttelefon erfordert leider mehr als 20 min. D.h. eine Telefonapplikation sollte in ihrer wesentlichen Funktionalität in max. 10 min. zu erlernen sein. Dies wird u.a. durch das innerhalb eines WSs homogene Interface unterschiedlicher Applikationen erreicht.

Auf eine Bedienungsanleitung sollte auch in der Lernphase nur in den seltesten Fällen zurückgegriffen werden müssen. Eine kontextsensitive Helpfunktion ist sehr hilfreich: Je nach Zustand der Applikation werden verschiedene Hilfstexte angezeigt. So könnte nach dem Selektieren der Funktion "Anrufumlekung" eine Hilfe angefordert werden. Diese würde die Funktion und mögliche Folgeeingaben erläutern.

Eine BGO zeichnet sich auch dadurch aus, daß sie vom Anwender leicht behalten werden kann. Dies wird durch die intuitive Assoziation mit Bekanntem unterstützt; das Telefonbuch (Register) wird durch Selektion des Anfangsbuchstaben geöffnet und eine Teilnehmerliste erscheint auf dem Bildschirm (siehe Abb. 3.1). Der Anwender kann nun z.B. durch einen Doppelklick auf



mf-of7.gem Abb. 3.1

einen Teilnehmer diesen selektieren und gleichzeitig anrufen. Mit einem einfachen Klick wird in MS-Windows allgemein ein Element selektiert, dieses wird dann invers angezeigt. Die Daten des selektierten Teilnehmers können angezeigt und editiert werden.

Der Gestalter einer BGO muß sich in die Situation der Anwender versetzen, wobei die teilweise sehr unterschiedlichen Benutzerklassen zu berücksichtigen sind.

Die Telefonapplikation soll durch die BGO effektiv anwendbar sein. Dies bedeutet, daß

- einzelne Funktionen logisch zusammenhängend gruppiert und ähnlich gestaltet sind (z.B. Anrufumlenkung und Anrufweiterlenkung. Beide Funktionen erfordern die Eingabe einer Rufnummer bzw. eines Teilnehmers und sie schließen sich gegenseitig aus, siehe Abb. 3.2),.
- einige graphische Symbole, die textuelle Ein- und Ausgabe ersetzen, fördern das schnellere Erkennen. Anstatt der Begriffe "Müll" und "Kurzwahl" können entsprechende Symbole angegeben werden
- zwischen verschiedenen Applikationen auch Daten ausgetauscht werden können. So würde z.B. dasselbe Adreßbuch auch für FAX-, Telex-, Teletex- und Mailing-Applikationen verwendet werden. Eine einfache Clipboardfunktion ist i.allg. nicht ausreichend. Die Applikationen sollten gekoppelt werden können. Hypertext und Hypermedia sind hier Ansätze, die auch auf andere Applikationen übertragbar sind (siehe z.B. /Gibbs Tsichritzis Fitas Konstantas Yeorgaroudakis 87/).

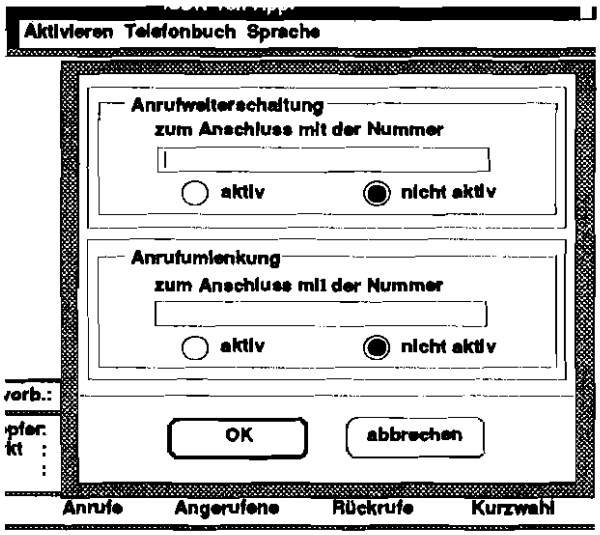


Abb 3.2: Anrufumleitung und -weiterschaltung

- Aktionen schnell ausgelöst werden können (z.B. Wählen durch Anklicken des Teilnehmers). Das Finden eines Teilnehmers muß sehr schnell ausgeführt werden. Durch Angabe eines teilqualifizierten Nachnamens kann auch ein Teilnehmer angewählt werden, siehe Abb. 3.3. Dies ist die Alternative zum Öffnen der Listen, evtl. Scrollen und einem Doppelklick)
- die Bedürfnisse sowohl für sporadische als auch für professionelle Anwender befriedigt werden müssen. Eine einfache Lösung bietet (soweit erforderlich) die anwendergesteuerte Konfiguration.

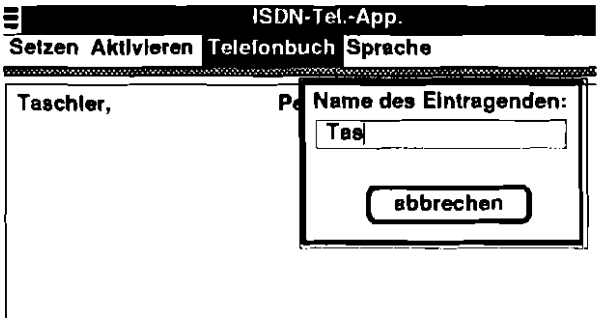


Abb. 3.3: Schnelle Auswahl eines Teilnehmers

Bezüglich der Ästhetik sind die Fonts, Farbkombinationen, Auflösung und Form der Fenster zu beachten, diese prägen den wesentlichen ersten Eindruck des Anwenders (siehe z.B. /Gait 85, Gait 86/).

Erstrebenswert ist die Erstellung nur einer Applikation für unterschiedliche Anwenderkreise und Sprachen. Dies kann durch entsprechende Trennung (im WS oder der Applikation realisiert) der Texte, Graphik und des eigentlichen Programms erreicht werden.

Um das Ziel einer effektiven Realisierung der BGO zu erreichen, müssen auch die Anforderungen von Seiten des Applikationprogrammierers berücksichtigt werden, die sich vor allem in einer kostengünstigen Lösung niederschlagen. Um dem evtl. fehlenden Wissen über den späteren Kundenkreis frühzeitig entgegenzuwirken, bietet sich eine "Rapid Prototyping" Methode an. D.h. ohne die Inhalte des eigentlichen Programms ausgefüllt zu haben, kann die BGO getestet und verändert werden. Die Applikation wird durch die Verwendung von WS in gewissen Grenzen hardwareunabhängig. Durch Programmgeneratoren bzw. Toolkit-Environments kann der Entwicklungsaufwand verkürzt und die Wartungsfreundlichkeit erhöht werden. Diese sind heute leider kaum erhältlich.

4. Problem- und implementierungsspezifische Anforderungen als Designkriterien

Neben den im vorherigen Abschnitt erläuterten allgemeinen Kriterien bei der Gestaltung einer BGO können aus der eigentlichen Aufgabe problemspezifische Merkmale abgeleitet werden.

Das Fernsprechnetz und die telefonspezifischen Endgeräte unterliegen in der BRD den Richtlinien der DBP. Auch bei einer Liberalisierung der Politik der DBP soll das Endgerät Telefon weiterhin von der DBP vertrieben werden, jedes anzuschließende Gerät muß durch das FTZ eine Zulassung erhalten. Die von der DBP gestellten Anforderungen sind in der "Technische Forderung an digitale Endgeräte mit S_0-Schnittstelle" (1 TR 3) von der FTZ Richtliniensammlung zusammengestellt. Für den Entwurf der BGO sind der Teil 3 und Teil 10 dieser 1 TR 3 von besonderer Bedeutung.

Im Teil 3 werden die "Bedienerprozeduren" spezifiziert. Die dort getroffenen Aussagen gelten für alle am ISDN der DBP angeschlossenen Endgeräte. "Sie dienen dazu, dem Benutzer des ISDN an möglichst vielen Endgeräten dieser Dienste eine möglichst gleichartige Benutzungsweise zu ermöglichen. Es soll dadurch verhindert werden, daß vor der Benutzung der Grundfunktionen eines Endgeräts zunächst das Studium einer gerätespezifischen Bedienungsanleitung zwingend erforderlich ist. ... Sofern bei einem Gerät Verbesserungen oder Alternativen zu den in dieser Richtlinie enthaltenen Benutzerprozeduren implementiert sind, muß daneben auch die hier beschriebene Benutzerprozedur möglich sein" (/FTZ 1TR3T3 87/ S. 4). Unter der Benutzerprozedur ist ein Zustandsübergangsdiagramm mit den möglichen Eingaben der Teilnehmer zu verstehen. Eingaben werden über Tasten getätigt. Diese Forderung ist auf einen herkömmlichen Telefonapparat mit Tasten und einem alphanumerischen, einzeiligen Display ausgerichtet. Bei der Verwendung eines WSs kann selbstverständlich ein solcher Dialog programmiert werden, doch es ist nicht sehr sinnvoll: Wesentliche Vorteile eines WSs werden nicht ausgenutzt. Um die Anforderung an Kompatibilität zu erfüllen, kann ein Tastenapparat mit den entsprechenden Bedienerprozeduren nachgebildet werden (siehe Abb. 4.1).

Im Teil 10 der 1 TR 3 werden die Leistungsmerkmale, die in einer Telefonapplikation zu unterstützenden sind, einzeln erläutert. Auch hier wird von dem Telefonapparat mit Tasten und einem alphanumerischen einzeiligem Display ausgegangen. So sind die am Endgerät darzustellenden Zustände angegeben. D.h. z.B. bei aktivierter Anrufumleitung "der Aktivierungszustand wird am Fernsprechapparat (FeAp) optisch signalisiert" (/FTZ 1TR3T3 87/ zu Zeile 39). Diese

Beschreibung liefert die Grundlage zu unserer Implementierung, sie wurde jedoch an die WSspezifischen Merkmale angepaßt.

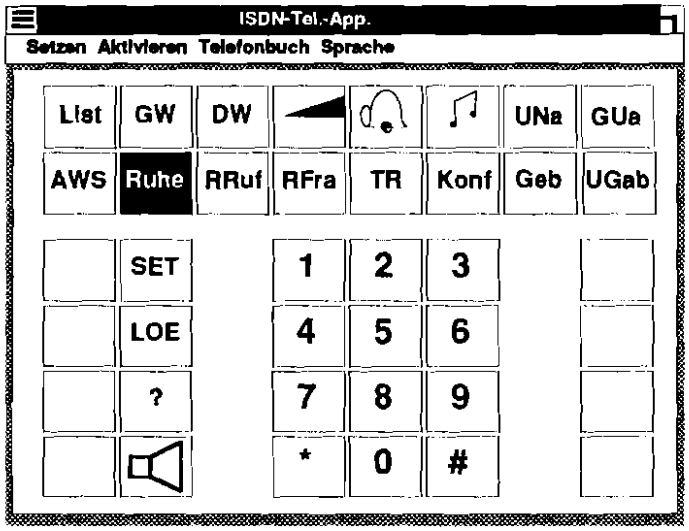


Abb 4.1: Tastenapparat als graphische Oberfläche

Ein FeAp muß immer einsatzbereit sein. Diese Forderung schlägt sich in der erforderlichen Hardware und Software nieder. Ein PC ist nicht für den Dauerbetrieb, d.h. 24 Sunden täglich, ausgelegt; er kann auch nicht ohne weitere Eingriffe bei einem ankommenden Ruf in der geforderten Zeit hochfahren. Die einfachste Lösung ist eine fremdgespeiste Komponente der Telefonhardware, die ohne eingeschalteten PC funktionstüchtig ist. Bei der Konzeption von kommunikationsorientierten Workstations wird auch eine Lösung zum Hochfahren des Rechners bei externen Ereignissen (z.B. ankommender Ruf) angestrebt werden. Dann kann auch die Festplatte als Sprachspeichermöglichkeit dienen. Während der PC eingeschaltet ist muß die Telefonapplikation geladen, d.h. betriebsbereit sein. Auch muß immer der Zustand des FeAp (d.h. der Telefonapplikation) sichtbar sein.

An die Reaktionszeit der Telefonapplikation werden gewisse Anforderungen gestellt. Dies bedeutet, daß das WS einen "preemtive" Schedulingmechanismus bereitstellen muß. Jeder Task des WS muß nach maximal einer gewissen Zeitspanne unterbrochen werden können, unabhängig von der augenblicklich ausgeführten Aufgabe. Damit können z.B. ankommende Rufe fast sofort signalisiert werden. Eine vorübergehende Lösung kann in einer auf der Telefonhardware ausgelagerten akustischen Signalisierung bestehen. Anschließend kann der Teilnehmer ggf. die aktuelle Applikation unterbrechen und die Telefonapplikation in den Vordergrund holen.

Um den Zustand der Telefonapplikation immer ohne Verwendung des gesamten Bildschirms anzuzeigen, wird der Zustand als Teil des Telefon-Icons (wenn das Fenster geschlossen, die Applikation aber noch aktiv ist) angezeigt. D.h. je nach anzuzeigenden Zuständen ändert sich das Icon.

Die anzuwählende Nummer hängt von dem Ort des Anschlusses ab. Um die Daten der BGO in verschiedenen Rechnern einsetzbar zu machen wird jeweils die komplette Rufnummer gespeichert und der Teilnehmer gibt seine lokale Rufnummer an (siehe Abb. 4.2)

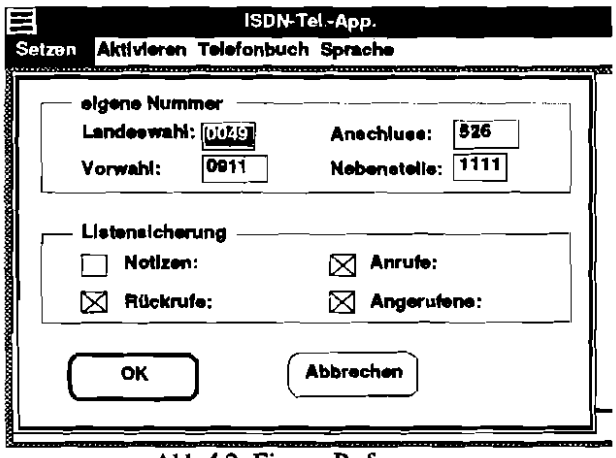


Abb 4.2: Eigene Rufnummer

Leider fließen beim Design der BGO auch einige Randbedingungen bezüglich einer speziellen Implementierungsumgebung, wie teilweise

schon erwähnt, ein. Weiterhin ist neben den Primitiven des WSs der Graphikschirm mit seiner Auflösung, Größe, Farbpalette oder S/W maßgeblich. Der Prototyp wurde auf einem Philips P3200 (AT kompatibel) mit EGA Graphik (d.h. u.a. Farbe, 640 x 350 Pixel) unter Verwendung von MS-Windows realisiert.

5. Einige beim Designprozeß anfallende Entscheidungen

Es muß die Zuordnung der aktivierbaren Funktionen
- zu Einträgen im Menü (diese sind nicht immer sichtbar) oder
- zu Elementen auf der Oberfläche (diese sind immer sichtbar)

getroffen werden.
Z.B.: Die Funktion "Setzen der Endgeräteauswahlziffer" wird nur selten verwendet und steht deshalb im Bereich der Menüzeile. Das Telefonbuch (die Anfangsbuchstaben) wird immer angezeigt, ebenso z.B. die Kurzwahlliste.

Einzelne Elemente oder Funktionen müssen sinnvoll, zusammenhängend plaziert werden. Dies kann
- eine alphabetische Reihenfolge oder
- eine logische Gruppierung bedeuten.

Die Einträge in das Telefonbuch werden alphabetisch sortiert, Funktionen wie Anrufumleiten und Anrufweiterschalten sind logisch eng verbunden und werden sehr ähnlich behandelt und dargestellt (siehe Abb. 3.2). Kritischer wird z.B. die Entscheidung der Plazierung in den Pull-Down Menüs. Bei der BGO der Telefonapplikation haben wir uns für eine alphabetischen Reihenfolge entschieden. Der Gesamteindruck der BGO muß ausgewogen sein und soll einigen grundsätzlichen Charakteristika von Schönheit /Gait 85, Gait 86/ genügen.

Während des Arbeitens mit der Telefonapplikation nimmt diese unterschiedliche Zustände an, in denen nur gewisse Eingaben getätigt werden sollen. Nicht anwählbare Funktionen können
- entweder nicht vorhanden sein
  (z.B. könnte auf der BGO der Ziffernblock ausgeblendet werden, wenn keine Ziffern eingegeben werden sollen)
- oder sie werden angezeigt, sind aber als deaktiviert gekennzeichnet und Interaktionen mit diesen werden ignoriert
  (z.B. werden in der Menüleiste alle nicht selektierbaren Einträge grau dargestellt, im Gegensatz zu schwarz als anwählbar)
- oder es wird ein Dialogfenster als "Confirmer" konzipiert
  (dieser "Confirmer" muß geschlossen werden, bevor nicht zu diesem Fenster gehörende Eingaben vom System angenommen werden; andere Funktionen werden durch diese Dialogbox ganz oder teilweise verdeckt).

Dabei muß beachtet werden, daß die am häufigsten verwendeten Funktionen als "Control Panel" immer sichtbar sind. Die Funktionen des Control Panels sollten immer sinnvoll einzusetzen sein.

Die Darstellungsart, d.h. das optische Erscheinungsbild an der BGO, kann folgende Varianten annehmen:
- vollständiger Text
- abgekürzter Text
- Icon, d.h. Graphik

Jede Möglichkeit hat ihre Vor- und Nachteile, die leider oft in Abhängigkeit des jeweiligen Anwenders von unterschiedlicher Bedeutung sind:

Der vollständige Text ist einfach zu verstehen, aber unübersichtlich für den geübten Anwender (z.B. "Anrufweiterschaltung"). Abkürzungen wie "AWS" sind dem täglichen Benutzer geläufig, jedoch keinesfalls für sporadische Anwender geeignet. Icons müssen innerhalb des WSs einer homogenen Semantik unterliegen, so ist beispielsweise der Papierkorb eindeutig mit einer Löschfunktion zu assozieren.

Die verschiedenen Dialogboxen sollten einen ähnlichen Aufbau besitzen. So sollten sich z.B. die Buttons "OK" und "Abbruch" immer an einer ähnlichen Position befinden. Auch zusammengehörige Fenster sollten ihre enge Beziehung durch eine gleiche Lage, ähnliches Aussehen, Farbe, u.s.w. ausdrücken. Diese Ähnlichkeit sollte sich auch auf andere Applikationen desselben WSs beziehen. Ähnliche Parameterabfragen und Kommandos können sich in einer statt in mehreren Dialogboxen befinden.

Es ist zu entscheiden, inwieweit mehrere Dialogboxen gleichzeitig geöffnet sein sollten, wie der Zusammenhang verdeutlicht werden kann und wie die neuen Möglichkeiten durch eine zusätzliche Box ausgedrückt werden können (Lage zueinander / ineinander, teilweise Überdeckung alter Dialogelemente durch neue, ...) . Dabei darf auch z.B. der Abbruch-Button eines Fensters nie physikalisch an derselben Position wie ein Save-Button der darunterliegenden Dialogbox liegen.

Einige nützliche Hinweise zum Design einer BGO seien noch am Ende dieses Abschnittes aufgezeigt:

- Das Aussehen des Cursors kann verändert werden, um die momentan anfallende Arbeit oder Zustand zu visualisieren:
  So kann anstatt des Pfeils eine Eieruhr zum Verdeutlichen des "Beschäftigtseins" erscheinen.
- Falls zeitintensivere Arbeiten einer Applikation anfallen, sollte der Verlauf der Verarbeitung von Zeit zu Zeit angezeigt werden; es sollte also die jeweils schon durchgeführte Arbeit in Relation zur gesamten Arbeit(szeit) symbolisiert werden:
  Das Formatieren einer Diskette kann durch einen sich füllenden Balken, eine Rechnerkopplung durch das Fortschreiben der Anzahl der übertragenen Pakete symbolisiert werden. Der eigentliche Sinn dieser Maßnahme besteht darin, daß ein Anwender erkennen kann, daß die Applikation arbeitet und ggf. den Stand dieser Arbeit beurteilen kann. Somit kann er sich auf die noch abzuwartende Zeitspanne einstellen, und es ist klar ersichtlich, daß der Rechner oder die Applikation nicht "abgestürzt" sind.
- Eine vom Benutzer selektierte Operation soll auf der graphischen Oberfläche sofort als "in Arbeit" quittiert werden, bevor sie wirklich begonnen wird. Somit wird gewährleistet, daß nicht unnötige weitere Eingaben dieser Funktion erfolgen.

6. Ausblick

Trotz der Erarbeitung und Anwendung aller in diesem Artikel beschriebenen Designkriterien war es schwer, eine BGO zu entwerfen. Viele Ideen und Varianten sind erst bei der konkreten Realisierung und Anwendung entstanden. Das schwierigste Problem ist und wird die Vorstellung des Designers und Implementierers von der inhomogenen Benutzergruppe mit ihren Auswirkungen auf den Entwurf bleiben. Aus dieser Erkenntnis leitet sich eindeutig der folgende Weg einer Realisierung ab: Die Randbedingungen bezüglich des WSs und des Zielsystems werden zusammen mit dem Verständnis des eigenlichen Problems und des zukünftigen Anwenderkreises festgelegt bzw. erarbeitet. Anschließend soll möglichst schnell die eigentliche BGO (ohne der komplett unterlegten Funktionalität) mit Verfahren des "Rapid Prototyping" (z.B. Smalltalk-Um-

gebung oder mit entsprechenden Programmgeneratoren und Tools) realisiert werden. Die endgültige BGO wird durch einen iterativen Prozeß mit Versuchspersonen optimiert. Dabei wird ein klare Trennung zwischen der BGO (dem User-Interface) und der Funktionalität /Coutaz 85/ vorausgesetzt.

Sehr hilfreich für einen solchen Rapid Prototyping-Ansatz sind auch wesentlich komfortablere "Programming Toolkits" bzw. "Resource Construction Sets" als die heute verfügbaren. Hier kann an zwei Komponenten gedacht werden:

- Tool zur Erstellung, d.h. Plazierung und Komposition von Fenstern, z.B. Dialogboxen. Dies soll interaktiv geschehen, um sofort das visuelle Ergebnis kontrollieren zu können.
- Tool zum Zusammensetzen der einzelnen Fenster (dynamischer Objektbaum) zu einem Programm. Z.B: Das Anklicken eines Buttons "Edit" im Fenster "Telefonbuch" öffnet ein Fenster "Eintragungen".

Die Telefonapplikation kann neben der PC-Umgebung auch als Protoyp für ein High-end-Komforttelefon mit Flachbildschirm angesehen werden. Zur Portierung der Software müssen einige graphische Primitiven des WSs auf dem dedizierten Mikroprozessorsystem nachgebildet werden. Hierfür ist in unserem Beispiel allerdings die vollständige Programmierschnittstelle von MS-Windows ungeeignet, es werden einige Funktionen weggelassen andere in veränderter Form konzipiert.

Das gesteigerte Interesse an der Telefonapplikation und die Entwicklung kommunikationsorientierter Arbeitsplatzrechner verschiedener Hersteller deutet auf die Bedeutung graphischer Oberflächen und deren Design in Zukunft hin.
Alternativ oder zusätzlich zu BGOen kann gerade bei dem Telefon die sprachgesteuerte Eingabe wie in zukünftigen Mobilfunksystemen eingesetzt werden.

An dieser Stellen möchten wir uns ebenso bei Herrn A. Wepner für zahlreiche praktische Hinweise und Diskussionen wie auch bei Herrn Dr. K. Meißner für Anregungen und die Unterstützung dieser Arbeit in der PKI und Herrn W. Leinweber für Hinweise bei der Gestaltung der Oberfläche bedanken.

7. Anhang

Anmerkung: Diejenigen Bezeichnungen von in diesem Artikel genannten Erzeugnisse, die zugleich eingetragene Warenzeichen sind, wurden nicht besonders kenntlich gemacht.

7.1 Verwendete Abkürzungen

BGO:Benutzerfreundliche graphischen Oberfläche
DBP:Deutsche Bundespost
FeAp:Fernsprechapparat
FTZ:Fernmelde Technisches Zentralamt, Darmstadt
PKI:Philips Kommunikations Industrie AG
WS:Window System

7.2 Literaturverzeichnis

/CARROL, MAZUR 86/: John M. Carrol, Sandra A. Mazur
LisaLerning,
IEEE Computer, Vol. 19, Nr. 10, Nov. 1986, S. 35-49.

/CHANG 87/: Shi-Kuo Chang,
Visual Languages: A Tutorial and Survey,
IEEE Software, Vol. 4, Nr. 1, Jan. 1987, S. 29-39.

/CLEMONS, GREENFIELD 85/: Eric. K. Clemons, Arnold J. Grennfield,
The SAGE System Architecture: A System for the Rapid Development of Graphics Interfaces for Decision Support,
IEEE Comp. Graphics and Appl., Vol.12, Nr. 11, Nov. 1985, pp. 38 - 50.

/CORMICK 82/: E. J. McCormick,
Human Factors in Engineering and Design,
McGraw hill, New York, 1982.

/COUTAZ 85/: Joelle Coutaz,
Abstractions for User Interface Design,
IEEE Computer, Vol. 18, Nr. 9, Sep. 1986, S. 21-34.

/FOLEY, WALLACE, CHAN 84/: James D. Foley, Victor L. Wallace, Peggy Chan,
The Human Factors of Computer Interaction Techniques,
IEEE Comp. Graphics and Appl., Vol.11, Nr. 11, Nov. 1984, pp. 13 - 48.

/FTZ 1TR3T3 87/
Technische Forderungen an digitale Endgeräte mit S0-Schnittstelle,
Teil 3, Feb. 87.

/FTZ 1TR3T10 87/ FTZ,
Technische Forderungen an digitale Endgeräte mit S0-Schnittstelle,
Teil 10, Mai 87.

/GAIT 85/: Jason Gait,
An Aspect of Aesthetics in Human-Computer Interaction: Pretty Windows,
IEEE Trans. on Soft. Eng., Vol. SE-11, Nr. 8, Aug. 1985, S. 714-717.

/GAIT 86/: Jason Gait,
Pretty Pane Tiling of pretty Windows,
IEEE Software, Vol. 3, Nr. 5, Sep. 1986, S. 9-14.

/GIBBS TSICHRITZIS FITAS KONSTANTAS YEORGAROUDAKIS 87/ S. Gibbs, D. Tsichritzis, A. Fitas, D. Konstantas, Y. Yeorgaroudakis
Muse: A Multimedia Filing System,
IEEE Software, Vol. 4, Nr. 2, März 1987, S. 4-15.

/MURCH 84/: Gerald M. Murch,
Physiological Principles for the Effective Use of Color,
IEEE Comp. Graphics and Appl., Vol.11, Nr. 11, Nov. 1984, pp. 49 - 54.

/MYERS 84/: Brad A. Myers,
The User Interface for Sapphire,
IEEE Comp. Graphics and Appl., Vol.11, Nr. 12, Dec. 1984, pp. 13 - 32.