# IV

# Modulare anwendungsneutrale Benutzerschnittstellen

Michael Herczeg

Benutzerschnittstellen stellen im Gegensatz zu Schnittstellen zwischen Programmen den Kontakt des Computers zur Außenwelt, zum Menschen her. Sie haben den Eigenschaften zweier grundlegend verschiedener Systeme Rechnung zu tragen: denen des Menschen und denen des Computers. Dies macht sie zum am schlechtesten spezifizierbaren Teil eines Computersystems. Die Benutzerschnittstelle muß sowohl den technischen Gegebenheiten des Computers als auch den psychologischen und physiologischen Anforderungen des Menschen gerecht werden.

Die Benutzerschnittstelle ist der Zugang zum Werkzeug Computer. Diese wichtige Stellung, die Komplexität dieser Schnittstelle und die zunehmende Bedeutung des Werkzeugs Computer für unser Arbeits- und Privatleben rechtfertigt nicht nur, sondern fordert geradezu eine neue Wissenschaft, die sich mit dieser Schnittstelle unter ergonomischer Sichtweise befaßt, die *Software-Ergonomie*. Da die Software-Ergonomie informationstechnische, psychologische sowie Aspekte des Arbeitsumfelds zu betrachten hat, kann sie nur eine interdisziplinäre Wissenschaft sein, die zumindest Physiologie, Psychologie, Informatik und Arbeitswissenschaften zu bemühen hat.

Neben den ergonomischen Betrachtungen ist die Erstellung von Benutzerschnittstellen auch eine ingenieurwissenschaftliche Aufgabe [Herczeg 85]. Die Anforderungen an Softwaresysteme, die Benutzerschnittstellen realisieren, werden in Zukunft weiter ansteigen. Man benötigt allgemeine Methoden und Werkzeuge zum Design und zur Konstruktion von Benutzerschnittstellen, um zu möglichst guten und wirtschaftlich vertretbaren Lösungen zu kommen.

Ein bekanntes und bewährtes Ingenieurprinzip, die *Modularisierung* von Systemen, ist ein möglicher Weg, der bei der Entwicklung von Benutzerschnittstellen auf seine Eignung untersucht werden muß. Das Bereitstellen von Bausteinen setzt zum einen voraus, daß es isolierbare Methoden und Eigenschaften von Benutzerschnittstellen gibt, zum anderen ist es zu wünschen, daß diese Methoden und Eigenschaften in dem Sinn *anwendungsneutral*[1] sind, daß sie als Bestandteil der Benutzerschnittstelle für verschiedene Anwendungssysteme geeignet sind. Mit der Methode der Standardisierung verbindet sich häufig das Problem der geringen Individualisierbarkeit. Wir werden sehen, wie objektorientierte Programmierung und damit zusammenhängende Vererbungsprinzipien diese Restriktion nicht nur beseitigen, sondern Individualisierbarkeit von Systemen sogar fördern (siehe auch [Smalltalk 81]).

In diesem Kapitel werden verschiedene Gründe diskutiert, die zu dem Interesse an modularen anwendungsneutralen Benutzerschnittstellen führen. Daraufhin werden vier Modelle von Benutzerschnittstellen beschrieben, ausgehend von einem einfachen Modell, bei dem die Benutzerschnittstelle fest im Anwendungssystem integriert ist, bis hin zu einem Modell, bei dem sie durch ein weitgehend losgelöstes Netz von Objekten repräsentiert wird. Am Beispiel von Icons[2] wird beschrieben, wie, ausgehend von Benutzeranforderungen, eine derartige Komponente realisiert wurde. Im Projekt Inform haben wir einen Lösungsansatz gewählt, der sich auf die objektorientierte Wissensrepräsentationssprache ObjTalk[3] stützt. Die entwickelten Bausteine für Benutzerschnittstellen sind Objekte, die ihre vielfältigen Eigenschaften außer durch deskriptive und prozedurale Definitionen auch durch multiple Vererbungsmechanismen und Constraints erhalten.

## 1. Gründe für anwendungsneutrale Benutzerschnittstellen

Das Ziel, das mit anwendungsneutralen Benutzerschnittstellen angestrebt wird, ist, einen "Baukasten" zu erstellen, um damit Benutzerschnittstellen für möglichst viele Anwendungen aufbauen zu können. Es gibt inzwischen einige Softwaremoduln zur Konstruktion von Benutzerschnittstellen, die gewissermaßen Bausteincharakter haben. Heutzutage reali-

---

[1] Statt des Begriffs "Anwendungsneutralität" wird im folgenden gelegentlich "Anwendungsunabhängigkeit" synonym verwendet werden.

[2] Diese speziellen, ikonischen Darstellungen auf Computerbildschirmen werden auch Piktogramme genannt.

[3] Siehe Kapitel III.

sierte und häufig verwendete Bausteine sind beispielsweise Menüs zur Auswahl von Systemfunktionen und sogenannte Bildschirmmasken oder Formulare zur Dateneingabe. Ein solcher "Baukasten" hat die folgenden Vorteile:

- Konsistenz in der Benutzung
- einfache Änderbarkeit
- hohe Leistung
- unabhängige Konstruktion
- Rapid Prototyping
- hohe Wirtschaftlichkeit

## 1.1 Konsistenz in der Benutzung

Gegenwärtig sieht sich der Benutzer von Anwendungssystemen einer Vielfalt von unterschiedlichen Benutzerschnittstellen gegenüber. Nahezu jede Anwendung hat ihr eigenes Konzept der Kommunikation mit dem Benutzer. Da er in Zukunft sicher weitere Anwendungssysteme zu bedienen haben wird, ist eine schrittweise Konvergenz der Methoden für die Mensch-Computer-Kommunikation unumgänglich. Durch die Verwendung von standardisierten Komponenten zum Bau von Benutzerschnittstellen kann die Konsistenz bei der Benutzung von Computern gefördert und damit die Interaktion erleichtert werden.

Die Vereinheitlichung muß nicht einhergehen mit plumpen, für die Anwendung unangemessenen Standardtechniken, sondern sollte eher mit der Auswahlmöglichkeit bei heutigen Automobilen zu vergleichen sein, wo man sich eines der Grundmodelle heraussucht und darauf basierend noch vielfältige Austattungen erhalten kann, ohne daß dafür ein spezielles und grundlegend neues Auto gebaut werden müßte. Trotz dieser Individualisierung bleibt die Reihenfolge von Kupplungs-, Brems- und Gaspedal dieselbe. Das Bremspedal eines Computersystems ist heutzutage - um bei dieser Metapher zu bleiben - im Vergleich zu einem anderen Computersystem nicht nur mit einem anderen Pedal vertauscht, sondern findet sich unter Umständen im Handschuhfach, wo es wahrscheinlich nicht vermutet wird. Derartige Inkonsistenzen zwischen verschiedenen Systemen sind auf Dauer nicht tragbar. Sie belasten die Benutzer unnötig und provozieren dadurch Fehler und Unzufriedenheit.

## 1.2 Einfache Änderbarkeit

Die Komplexität von Anwendungssystemen hat allein schon durch ihre Größe zugenommen. Es wird für Softwareentwickler und erst recht für Anwender immer schwieriger, Änderungen an der Benutzerschnittstelle durchzuführen. Die Isolation von einzelnen Mechanismen und Eigenschaften der Benutzerschnittstelle in Komponenten ist ein gangbarer Weg, um zu besserer Veränderbarkeit zu kommen. Existieren für diese Komponenten darüberhinaus geeignete Schnittstellen, um ihr Verhalten zu ändern (Metaschnittstellen, wissensbasierte Komponenten), so hat man eine neue Ebene der Programmierung erreicht: die Modifikation von Systemen durch den Benutzer selbst. Man spricht in diesem Zusammenhang von adaptierbaren Systemen. Dadurch hat der Benutzer die Möglichkeit, die Benutzerschnittstelle innerhalb bestimmter Grenzen nach seinen Bedürfnissen zu modellieren. So werden viele Reibungsverluste vermieden, die zwangsläufig immer dann entstehen, wenn der Benutzer seine Änderungswünsche auf einem langen und meist unerfreulichen Weg über mehrere Zwischeninstanzen einem Programmierer mitteilt. Schon das Nahziel, dem Systementwickler bessere Eingriffsmöglichkeiten bei der Gestaltung von Benutzerschnittstellen zu bieten, ist kaum zu überschätzen.

## 1.3 Hohe Leistung

Je mehr Zeit für die Erstellung von Systemteilen aufgewendet wird, desto höher kann ihr Leistungsumfang, ihre Effizienz und ihre Zuverlässigkeit werden. Beispielsweise sind Datenbanksysteme ihrer Konzeption nach anwendungsneutrale Komponenten zum Bau von Anwendungssystemen. Das große Interesse an diesen Komponenten führte zu einem hohen theoretischen und praktischen Standard. In ähnlicher Weise interessiert man sich heute für anwendungsneutrale Benutzerschnittstellen, allerdings ohne dafür eine tragfähige Theorie zu haben. Bislang wurde aus verschiedenen Gründen nur die Entwicklung einiger Grundsysteme vorangetrieben, die ansatzweise den Anspruch der Anwendungsunabhängigkeit tragen können. Masken-, Menü- und in neuerer Zeit Fenstersysteme[4] könnten als Beispiele dafür genannt werden. Hat man den Nutzen derartiger Komponenten erkannt, so ist man i.a. bereit, beträchtliche Zeit für ihre Entwicklung und Verbesserung aufzuwenden. Dieser Zeitaufwand läßt sich durch die vielfältige Verwendbarkeit wieder mehrfach einsparen.

---

[4] Siehe Kapitel V.

## 1.4 Unabhängige Konstruktion

Es gibt inzwischen viele Rahmenbedingungen und Designkriterien für den Entwurf von Benutzerschnittstellen, beispielsweise ergonomische Forderungen, Möglichkeiten, die die Hardware bietet, und methodische Erfahrungen. Diese Anforderungen müssen jedesmal erneut auf ihre Berücksichtigung überprüft werden, wenn ein neues Anwendungssystem mit spezieller Benutzerschnittstelle entwickelt wird. Geht man allerdings den Weg der Bausteinbildung, so müssen diese Kriterien nur beim Entwurf eines Bausteins berücksichtigt werden; mit jedem Einsatz eines solchen Bausteins ist deren Berücksichtigung dann weitgehend garantiert. Die Konstruktion von Benutzerschnittstellen wird damit zu einer teilweise isolierten Aufgabe, die von Spezialisten auf diesem Gebiet gelöst werden kann.

## 1.5 Rapid Prototyping

Mit Hilfe existierender Schnittstellenkomponenten können leichter und schneller Prototypen für bestimmte Anwendungssysteme erstellt werden. Der Bau solcher Prototypen hilft einerseits dem Systementwickler, frühzeitig Designentscheidungen auf ihre Eignung und Funktionalität hin zu kontrollieren, und andererseits dem Endbenutzer zu entscheiden, ob das skizzierte (prototypisierte) System seinen Anforderungen gerecht werden kann. Der Prototyp dient ihm dann als Gegenstand der Kritik. An Schnittstellenkomponenten für prototypische Systeme müssen weitaus weniger hohe Anforderungen gestellt werden als an solche, die für Endprodukte verwendet werden sollen. Es bietet sich deshalb an, zuerst einmal solche Komponenten für Prototypen zu erstellen.

## 1.6 Hohe Wirtschaftlichkeit

Die bisher genannten Verbesserungen durch die Bereitstellung von Komponenten führen zu besserer und wirtschaftlicherer Erstellung und Wartung von Software. Häufig entfällt etwa die Hälfte der Software auf die Benutzerschnittstelle. Wirtschaftlichkeitsbetrachtungen waren die Haupttriebfeder für den Entwurf von Maskengeneratoren und Menügeneratoren, die schon seit längerer Zeit mit Erfolg angewendet werden. Softwareerstellende Betriebe sind heute stärker denn je bestrebt, derartige Generatorprinzipien für Benutzerschnittstellen weiterzuentwickeln. Diesen Bestrebungen steht allerdings ein Mangel an geeigneten Modellen und Implementierungstechniken gegenüber, der es erschwert, weitere, komplexere Bausteine in ihrem Aufbau und ihren Wechselwirkungen mit anderen Systemteilen auf eine operationale Art und Weise zu entwerfen, zu implementieren, mit Erfolg einzusetzen und problemlos zu warten.

# 2. Modelle für den Bau von Benutzerschnittstellen

Anhand von vier Modellen soll im folgenden der Weg skizziert werden, der von der herkömmlichen Methodik, Benutzerschnittstellen zu Anwendungssystemen zu bauen, zu einer neuen Vorgehensweise führt, nämlich der objektorientierten Realisierung weitgehend anwendungsneutraler Bausteine für Benutzerschnittstellen. Es werden drei Schritte beschrieben, ausgehend von der Benutzerschnittstelle als integralem Bestandteil des Anwendungssystems (Modell 1):

- *Abkopplung (Modularisierung)* eines möglichst großen Teils der Benutzerschnittstelle vom Anwendungssystem zu einer weitgehend eigenständigen Systemkomponente (Modell 2);
- *Anwendungsneutralität,* d.h. die Verwendung einer Benutzerschnittstelle (oder zumindest Teile davon) für mehrere Anwendungssysteme (Modell 3);
- *Aufgliederung in Objekte,* d.h. die Benutzerschnittstelle selbst wird zerteilt in einzelne Objekte, die mit dem Benutzer, dem Anwendungssystem sowie untereinander kommunizieren (Modell 4).

## 2.1 Die Benutzerschnittstelle als Bestandteil des Anwendungssystems

In vielen Anwendungssystemen ist die Benutzerschnittstelle integraler Bestandteil des Anwendungssystems. Sie ist praktisch untrennbar davon, es sei denn, man trennt beide auf der Ebene einzelner Statements der Programmiersprache.

Lange Zeit hat man der Benutzerschnittstelle in Computersystemen nur geringe Beachtung geschenkt. Die Funktionalität der Systeme stand im Vordergrund. Um diese Funktionalität wurden die Elemente der Benutzerschnittstelle "herumgestrickt". Dies entspricht gewissermaßen einem Systementwurf von innen (ausgehend von der Funktionalität) nach außen (zur Benutzerschnittstelle). Auf diese Weise erhält jedes Anwendungssystem seine eigene und speziell dafür erstellte Benutzerschnittstelle (siehe Abbildung IV-1). Für den Benutzer bedeutet dies eine spezielle Form der Kommunikation, die abhängig vom Anwendungssystem ständiges Umdenken und Umlernen erfordert. Der Systementwickler muß für jedes Anwendungssystem eine neue Benutzerschnittstelle bauen, deren Eigenschaften sich meist nicht lokal ändern lassen, sondern sich durch das ganze System ziehen. Dies führt zu beträchtlichen Zeit- und Kostenfaktoren bei der Erstellung von Systemen.

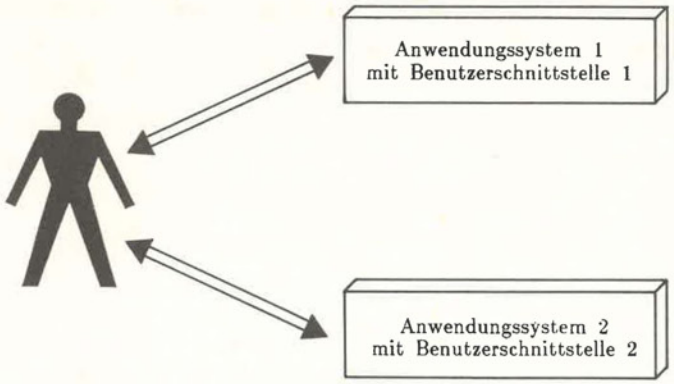


**Abbildung IV-1:** Die Benutzerschnittstelle als integraler Bestandteil der Anwendungssysteme (Modell 1)

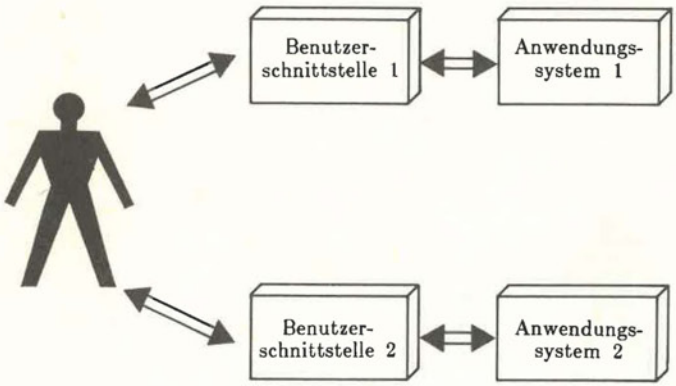


**Abbildung IV-2:** Die Benutzerschnittstelle als Modul zum jeweiligen Anwendungssystem (Modell 2)

## 2.2 Die Benutzerschnittstelle als Modul zum Anwendungssystem

Zieht man die Benutzerschnittstelle oder wesentliche Bestandteile als möglichst eigenständige Moduln aus dem jeweiligen Anwendungssystem heraus, so isoliert man damit Eigenschaften der Benutzerschnittstelle (siehe Abbildung IV-2).

Die Konzeption der Benutzerschnittstelle kann damit zumindest teilweise getrennt vom speziellen Anwendungssystem erfolgen. Dies ist ein Schritt in Richtung eines benutzerorientierten Entwurfs und damit eine Loslösung vom Prinzip "Systeme von innen nach außen zu entwickeln" (siehe dazu Abschnitt 2.1). Änderungen an der Benutzerschnittstelle können dann oft lokal erfolgen, ohne das Anwendungssystem selbst zu betreffen.

## 2.3 Die Benutzerschnittstelle zu mehreren Anwendungssystemen

Im vorhergehenden Modell besitzt jedes Anwendungssystem seine individuelle Benutzerschnittstelle. Mit dem Schritt der Modularisierung wird ein naheliegendes, weiteres Ziel verfolgt. Man möchte die gesamte oder wenigstens Teile der Benutzerschnittstelle für mehrere Anwendungssysteme verwenden (siehe Abbildung IV-3). Man kann Systeme dann schneller erstellen und dem Benutzer eine konsistentere Interaktion mit verschiedenen Systemen ermöglichen (siehe dazu Abschnitt 1).

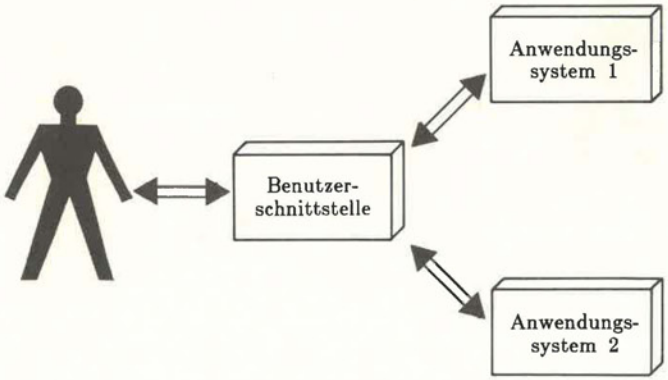


**Abbildung IV-3:** Die Benutzerschnittstelle zu mehreren Anwendungssystemen (Modell 3)

Anwendungsneutralität in Verbindung mit Modularisierung wird in vielen existierenden Systemen in Ansätzen praktiziert. Die Entwicklung von Maskengeneratoren, Menüsystemen oder Datenbankschnittstellen sind Beispiele dafür. Existierende Komponenten für Benutzerschnittstellen sind allerdings meist noch wenig individualisierbar und von keinem gemeinsamen Konzept getragen. Sie lösen damit nur einen Teil der Anforderungen.

## 2.4 Die Benutzerschnittstelle als Netzwerk von Objekten

Durch Verfeinerung des vorhergehenden Modells kommen wir zu einem Konzept, mit dem sich die gesamte Benutzerschnittstelle einheitlich beschreiben und realisieren läßt (siehe Abbildung IV-4).

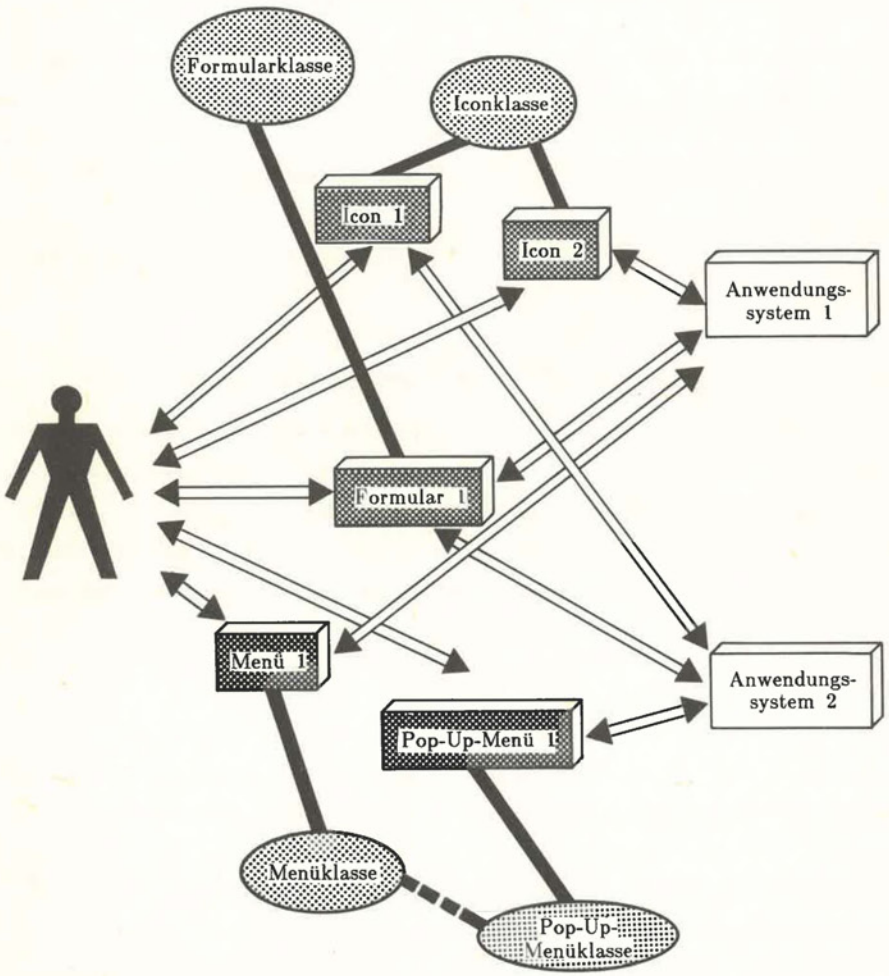


**Abbildung IV-4:** Die Benutzerschnittstelle als Netzwerk von Objekten (Modell 4)

Mit dieser Systemarchitektur gewinnt man höchste Modularität, kombiniert mit einfacher Individualisierbarkeit (siehe dazu die Ausführungen in Abschnitt 2.4.2), sowie eine Reihe weiterer Eigenschaften, die im folgenden beschrieben werden. Gemeinsame Grundlage der Elemente der Benutzerschnittstelle ist deren Konzeption als Objekte[5]. Sie können prinzipiell drei Arten Kommunikationspartnern haben: den Benutzer, das Anwendungssystem und andere Objekte der Benutzerschnittstelle.

### 2.4.1 Kommunikation mit dem Benutzer

Die Bausteine (Instanzen) der Benutzerschnittstelle kommunizieren mit dem Benutzer, indem sie seine Eingaben entgegennehmen, evtl. vorverarbeiten (z.B. korrigieren, formatieren) und weiterleiten sowie Information auf dem Bildschirm oder anderen Medien ausgeben. Obwohl der Benutzer mit Objekten der Benutzerschittstelle kommuniziert, kann er den Eindruck haben, er kommuniziere direkt mit dem Anwendungssystem. In Wirklichkeit filtert die Benutzerschnittstelle die "Benutzerfeindlichkeit" des Anwendungssystems heraus.

### 2.4.2 Kommunikation mit dem Anwendungssystem

Die Stellung von Objekten der Benutzerschnittstelle als Transferglieder bei der Kommunikation des Benutzers mit dem Anwendungssystem erfordert vielfältige Informationsübertragung zwischen ihnen und dem Anwendungssystem. Die Instanzen erfassen die Anfragen des Benutzers und formen sie geeignet in Anfragen an das Anwendungssystem um. Darüber hinaus können sie die Ausgaben des Anwendungssystems in für den Benutzer geeignete bzw. gewünschte Darstellungen verwandeln [Shaw et al. 83].

#### Kommunikationstechnik mit dem Anwendungssystem

Eines der Probleme mit aus dem Anwendungssystem separierten Elementen der Benutzerschnittstelle ist deren Kommunikation mit dem Anwendungssystem. Es gibt mehrere Techniken, wie Anwendungssystem und Objekte der Benutzerschnittstelle miteinander kommunizieren können:

- *Funktionsaufrufe:*
  Die Instanzen rufen Funktionen (oder Prozeduren) des Anwendungssystems auf.

---

[5] Im technischen Sinne sind es Instanzen verschiedener Klassen, siehe Kapitel III.

- *Botschaften:*
  An Objekte des Anwendungssystems[6] bzw. der Benutzerschnittstelle werden Botschaften gesendet.
- *Constraints:*
  Bestimmte Deskriptoren der Benutzerschnittstelle sind über Constraints[7] mit Deskriptoren des Anwendungssystems verknüpft.

Diese mögliche Bandbreite der Kommunikation ist notwendig, da unter Umständen sehr verschiedenartig konzipierte Anwendungssysteme mit der Benutzerschnittstelle kommunizieren wollen. Bei prozeduralen Systemen, die gegenwärtig die Mehrzahl aller Systeme ausmachen, sind Funktionsaufrufe (Prozeduraufrufe) die übliche Methode, während sich bei objektorientierten Anwendungssystemen eine Kommunikation auf der Ebene von Botschaften anbietet. Die Kopplung über Constraints wird nur in den Fällen in Frage kommen, wo Benutzerschnittstelle und Anwendungssystem auf gleicher Basis konzipiert und implementiert wurden.

**Zuordnung zu Anwendungssystemen**

Es soll nun dargestellt werden, wie die Zuordnung der Elemente (Instanzen) der Benutzerschnittstelle zu den einzelnen Anwendungssystemen aussieht. Es sind zwei Fälle zu unterscheiden (siehe dazu auch Abbildung IV-4):

- Eine Instanz der Benutzerschnittstelle kommuniziert mit mehreren Anwendungssystemen. Damit wird der Charakter der *Anwendungsneutralität* demonstriert (siehe Abbildung IV-5).
- Ein Anwendungssystem kommuniziert mit mehreren Instanzen der Benutzerschnittstelle. Der Teil der Benutzerschnittstelle wurde also aus dem Anwendungssystem herausgezogen und in einzelne *Moduln* zerteilt (siehe Abbildung IV-6).

**Individualisierung der Benutzerschnittstelle**

Ein häufig genannter Nachteil der Benutzung von Bausteinen zum Aufbau von Systemen ist ihre relative Starrheit. Manche Bausteine lassen sich nur mit unverhältnismäßig hohem Aufwand anpassen. Hat man sich entschieden, einen Baustein zu verwenden, so muß man ihn so verwenden, wie er ist. Beim Bau von Benutzerschnittstellen braucht

---

[6] Falls es solche gibt - das Anwendungssystem muß nicht objektorientiert realisiert sein.

[7] Siehe Kapitel III.

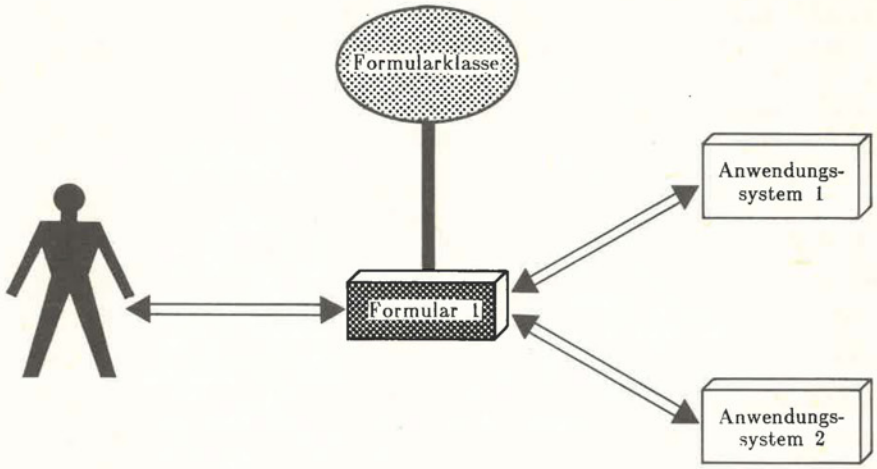


**Abbildung IV-5:** Eine anwendungsneutrale Instanz

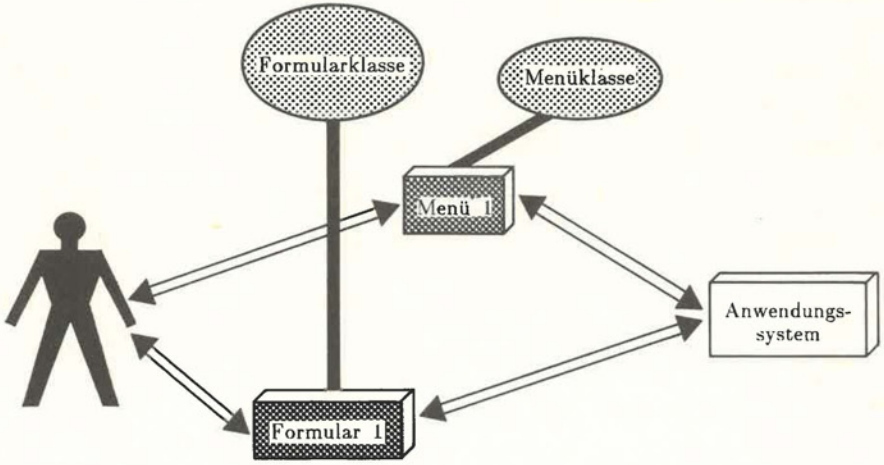


**Abbildung IV-6:** Instanzen als Moduln für Benutzerschnittstellen

man aber beträchtliche gestalterische Eingriffsmöglichkeiten, d.h. eine gewisse Individualisierbarkeit muß gegeben sein. Andererseits gibt es eine Menge von Eigenschaften der Benutzerschnittstelle, die sich von Anwendung zu Anwendung nicht ändern. Dies sollte man sich zunutze machen können.

Die objektorientierte Realisierung mit Vererbungsmechanismen liefert dazu elegante und praktikable Lösungen:

- Zwei Instanzen der gleichen Klasse kommunizieren mit je einem Anwendungssystem (siehe Abbildung IV-7). Die Instanzen verkörpern jeweils unterschiedliche Zustände bei gleicher Funktionalität *(Individualisierung durch Instantiierung)*. Dies könnten zum Beispiel zwei Icons sein, die jeweils speziellen Text enthalten.

- Zwei Instanzen haben eine gemeinsame Klasse in ihrem Vererbungsnetz (siehe Abbildung IV-8). Dies hat zur Folge, daß sie nur teilweise gleiche Slots und Methoden[8] haben *(Individualisierung durch Spezialisierung)*. In der Abbildung soll als Beispiel die Verfeinerung von normalen Menüs zu sogenannten Pop-Up-Menüs dargestellt werden. Pop-Up-Menüs unterscheiden sich von anderen Menüs ausschließlich durch die Eigenschaft, daß sie nur sichtbar sind, solange sie gebraucht werden, d.h. solange eine Auswahl im Menü stattfindet. Die Realisierung von Pop-Up-Menüs geschieht durch die Definition einer neuen Klasse, die die bereits vorhandene Menüklasse zur Superklasse hat und damit deren Eigenschaften ererbt. Die Pop-Up-Menüklasse enthält ansonsten nur die speziellen Pop-Up-Methoden.

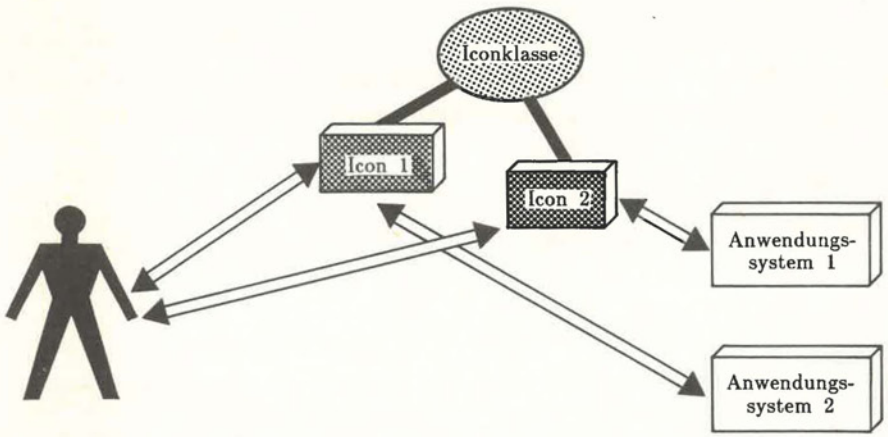


**Abbildung IV-7:** Individualisierung durch Instantiierung

[8] Siehe Kapitel III.

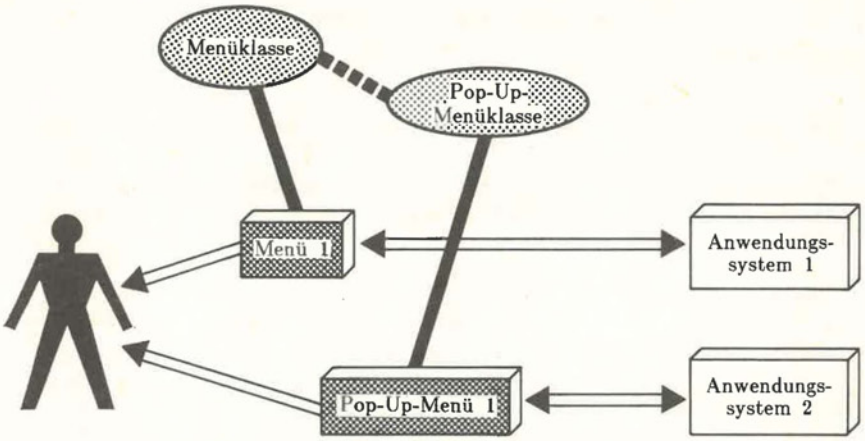


**Abbildung IV-8:** Individualisierung durch Spezialisierung

### 2.4.3 Kommunikation mit andereren Instanzen der Benutzerschnittstelle

Die Benutzerschnittstelle ist ein Netz von Instanzen[9], die in der Hauptsache als Vermittler zwischen Benutzer und Anwendungssystem wirken sollen. Es gibt mehrere Gründe, warum diese Instanzen auch untereinander kommunizieren müssen.

1. **Horizontale Schichtung:**

   Eine Reihe von Instanzen wirken als Bearbeiter von Teilaufgaben innerhalb der Benutzerschnittstelle. Sie reichen sich die veränderte Information weiter, bis sie fertig bearbeitet ist und an den Benutzer bzw. das Anwendungssystem ausgeliefert werden kann (siehe Abbildung IV-9).

   Ein Beispiel horizontaler Schichtung der Instanzen kann man sich bei einem Menüsystem vorstellen, wo die erste Instanz ständig die Position der Maus liest und diese einer zweiten Instanz meldet, die dann um die betreffende Menüzeile einen Rahmen zieht. Wählt der Benutzer dann eine Zeile aus, so stellt die zweite Instanz die Zeile invers dar und meldet einer dritten Instanz die Auswahl, die ihrerseits dann das Anwendungssystem davon unterrichtet, um dort eine entsprechende Funktion auszulösen.

[9] Vernetzung durch Kommunikations- und Vererbungspfade.

Mit dem IFIP-Modell [Dzida 83] wird eine ähnliche Aufteilung der Benutzerschnittstelle in Instanzen[10] vorgeschlagen. Es wird dabei in die Ein-/Ausgabeschnittstelle, die Dialogschnittstelle und die Werkzeugschnittstelle aufgeteilt.

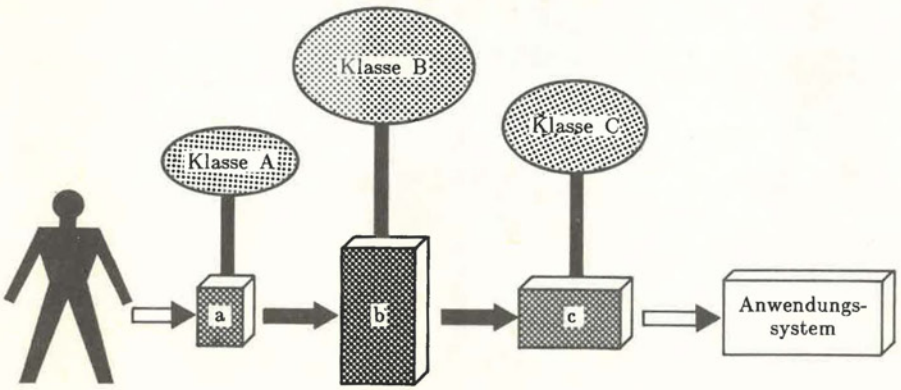


**Abbildung IV-9:** Horizontale Schichtung von Instanzen der Benutzerschnittstelle

**2. Vertikale Schichtung:**

Man kann sich vorstellen, daß es Objekte gibt, die dazu dienen, andere Objekte beeinflussen zu können. Man nennt sie *Metaobjekte*[11] (siehe Abbildung IV-10).

Wenn die betroffene Instanz beispielsweise ein Menü ist, so kann die Metainstanz das Menü dahingehend beeinflussen, daß die Farbe der jeweils ausgewählten Menüzeile geändert wird, um sie deutlicher hervorzuheben. Die Metainstanzen können also als Schnittstelle zur Gestaltung der Benutzerschnittstelle selbst dienen. Diese Gestaltung könnte sowohl vom Benutzer wie auch vom Anwendungssystem her gesteuert werden. Die Metaobjekte sind damit auch die Benutzerschnittstelle zur Benutzerschnittstelle.

**3. Synchronisation:**

Verschiedene Objekte der Benutzerschnittstelle müssen sich synchronisieren, wenn sie mit dem Benutzer oder dem Anwendungssystem kommunizieren wol-

[10] Beim IFIP-Modell spricht man von Prozessen und Schnittstellen.

[11] Siehe Kapitel IX.

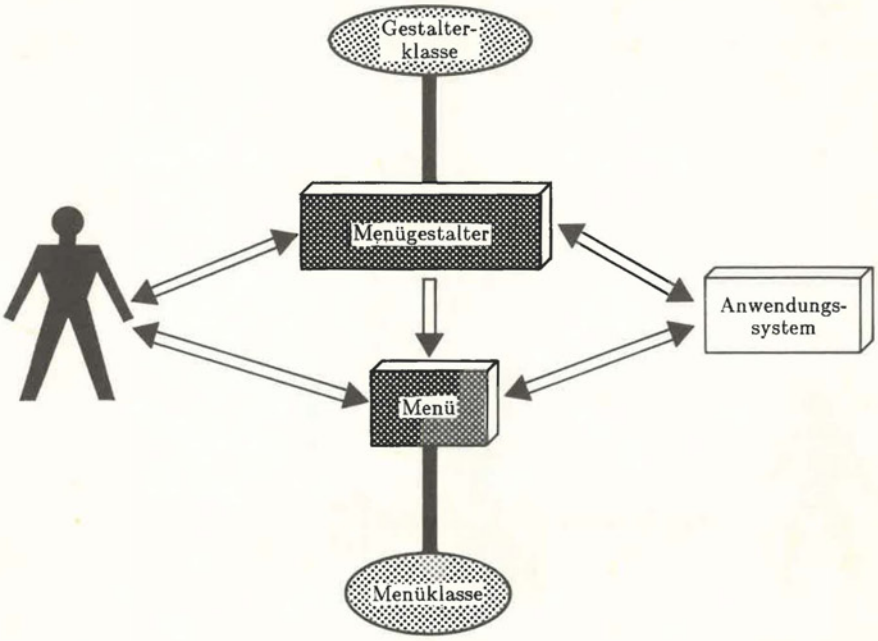


**Abbildung IV-10:** Vertikale Schichtung von Instanzen der Benutzerschnittstelle

len (siehe Abbildung IV-11). Im Fall von alternativen Interaktionsformen (z.B. Menüs oder Kommandozeile) muß zwischen den jeweiligen Objekten, die diese Interaktionsform repräsentieren, entschieden werden, welche zuständig ist. Bei parallelen Aktivitäten ist diese Synchronisation noch komplizierter. Beispielsweise können bei der Auswahl aus einem Menü mehrere Objekte beteiligt sein: eines, das die Darstellung auf dem Bildschirm aktualisiert und ein anderes, das zusätzlich eine kurze Erklärung der Menüzeile gibt, auf die gerade mit der Maus gezeigt wird. Beide Objekte können dazu direkt kommunizieren.

Bei den Window-Bound-Icons (siehe Abschnitt 3.4) gibt es auch eine sehr enge Synchronisation mit ihren jeweils assoziierten Fenstern, um zu garantieren, daß immer nur entweder Icon oder Fenster sichtbar ist, da diese Klasse von Icons gerade die Aufgabe hat, Fenster platzsparend zu repräsentieren.

Wir haben auf Grund der Modelle gesehen, wie sich der Weg von vollständig in das Anwendungssystem einbezogenen Benutzerschnittstellen über mehrere Stufen hin zu einer Benutzerschnittstelle aus einem Netz von Objekten zieht, die vielfältige Arten von Kommunikation und Individualisierbarkeit ermöglichen.

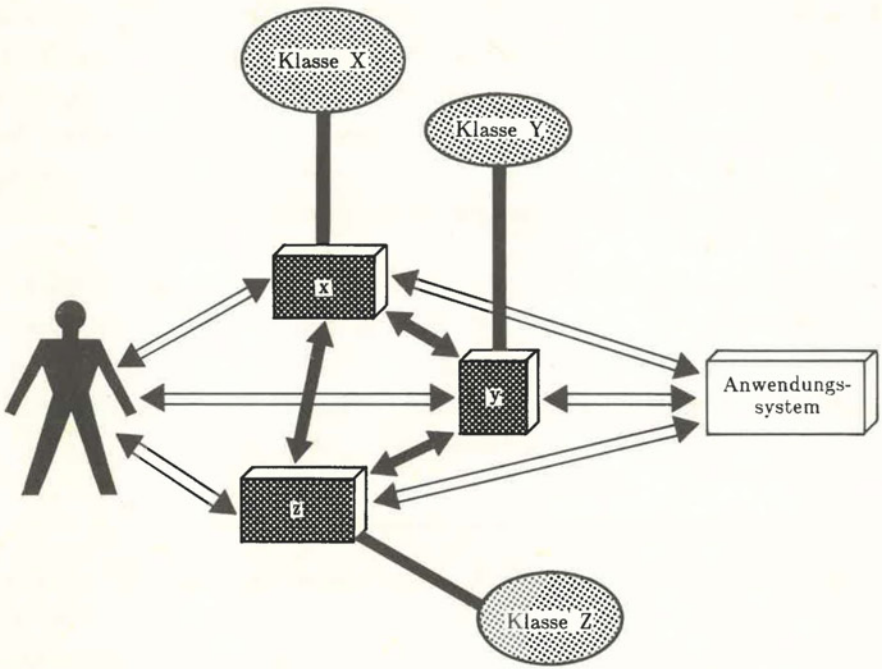


**Abbildung IV-11:** Synchronisation von Instanzen der Benutzerschnittstelle

Im folgenden Abschnitt wird anhand eines Beispiels erläutert, wie sich das Prinzip der modularen und anwendungsneutralen Benutzerschnittstellen bei einer Realisierung niedergeschlagen hat. Das Beispiel sind Icons. Wir werden zunächst begründen, wozu Icons benötigt werden und warum sie in vielen Anwendungen benutzbar sind.

## 3. Fallstudie: Icons als anwendungsneutrale Komponenten

Gründe für das Unbehagen bei der Umstellung von manuellen Systemen auf Computersysteme sind unter anderen das Nichtsichtbarsein oder Nichtgreifbarsein der vorhandenen Arbeitsobjekte sowie das neue Abstraktionsniveau und die eingeengten Möglichkeiten bei der Bearbeitung (Manipulation) dieser Objekte.

Der Sachbearbeiter hat keinen Karteikasten mehr, den er, in der gewohnten Art und Weise blätternd, durchsuchen kann. Er kann nicht einfach zwei Formulare (ausgefüllte

Bildschirmmasken), die noch aufeinander abzustimmen sind, mit einer Büroklammer zusammenheften und auf den Stapel der noch offenen Arbeiten legen. Er kann keinen Markierstift mehr nehmen, um eine fragliche Eintragung in einem Auftragsformular (das ihm als ausgefüllte Bildschirmmaske "vorliegt") zu markieren, die erst am nächsten Tag geklärt werden kann. Er kann keine Rechnungsaufstellung in eine Mappe packen und diese seinem Kollegen zur Weiterbearbeitung über den Tisch reichen.

All diesen gewohnten Praktiken liegt ein Prinzip zugrunde: die *direkte Manipulation* [Herczeg et al. 85; Shneiderman 83]. Die bisherigen Objekte am Arbeitsplatz hatten einen Aufenthaltsort, der meist bekannt, benennbar und damit natürlich war. Sie waren leicht sichtbar zu machen, und man konnte auf sie zeigen. Sie waren darüber hinaus greifbar, mit verschiedenen Methoden manipulierbar und damit gestaltbar.

## 3.1 Direkte Manipulation in Computerspielen

Bei den Computerspielen kann man das Prinzip der direkten Manipulation sehr gut beobachten. Dort möchte man dem Benutzer einen möglichst direkten Kontakt zu den Objekten des Spiels ermöglichen; man muß ihn aus Gründen der Akzeptanz sogar garantieren. Bei der Entwicklung von Schachcomputern wird dies auch sehr deutlich. Bei Geräten früher Generationen wurden die Züge in Form von Koordinaten zwischen Mensch und Computer ausgetauscht. Heutige Schachcomputer visualisieren das Schachbrett in der vom manuellen Schach gewohnten Art. In einer noch ausgereifteren Form benötigt man zur Eingabe der Züge auch keine Tastatur mehr, sondern deutet mit einem Zeigeinstrument auf die zu bewegende Figur und bewegt sie zum gewünschten Ziel. Dies ist eine Entwicklung von der indirekten zur direkten Manipulation in einer für den Menschen natürlichen Art und Weise.

## 3.2 Direkte Manipulation in Bürosystemen

Das Prinzip der direkten Manipulation wurde bereits mehrfach auf Bürosysteme angewandt. Dabei werden die Objekte der Bürowelt, wie z.B. Dokumente, Ordner, Ordnerschränke, Papierkorb und Postablagefächer, durch kleine Icons [Smith 77] visualisiert. Diese können dann bei Bedarf geöffnet werden, um den zugehörigen Inhalt anzusehen und zu bearbeiten. Durch Zeigen mit der Maus können Objekte ausgewählt und bewegt werden. Ein Fenstersystem sorgt dafür, daß sich solche Objekte auf dem Bildschirm überlappen können und somit keinen Restriktionen hinsichtlich der Platzwahl und Größe unterliegen. Man hat sich bemüht, mit diesen Systemen Teile der manuellen Bürowelt so

direkt wie möglich auf dem Computer nachzubilden. Systeme dieser Art, die bereits kommerziell verfügbar sind, sind das Xerox STAR 8010 Information System [Smith et al. 82] sowie die Systeme LISA [Williams 83] und MACINTOSH [Williams 84] von Apple. Andere Computerhersteller arbeiten ebenfalls an entsprechenden Systemen. In anderen Anwendungsbereichen (z.B. Computer-Aided Design) wird die direkte Manipulation in ähnlicher Art und Weise mit großer Selbstverständlichkeit und überzeugendem Erfolg seit längerer Zeit eingesetzt.

Da man sich das Prinzip der direkten Manipulation bei sehr vielen Anwendungssystemen vorstellen kann, ist es besonders interessant, inwieweit diese Methode in Form weitgehend isolierter Systemkomponenten (Komponenten der Benutzerschnittstelle) realisierbar ist. Im folgenden soll ein Ansatz zu einer solchen Lösung kurz erläutert werden, der sich auf das Modell 4 des Abschnitts 2 stützt.

## 3.3 Codierung von Objekten und Funktionen durch Icons

*Icons* sind abstrahierte bildhafte Darstellungen von Objekten[12]. Im folgenden ist ein Icon eine Darstellung (Repräsentant, Codierung) eines oder mehrerer Objekte auf einem Bildschirm. Dargestellt werden können beliebige Objekte des Anwendungssystems (z.B. Dokumente) oder Objekte des Dialogsystems (z.B. die Liste der bereits ausgeführten Aktionen). Der Begriff "Objekt" war bis jetzt noch recht eng gefaßt und bezog sich auf den intuitiven Begriff eines Objekts, das man "anfassen" kann, wie wir es aus unserer Umwelt gewöhnt sind. Diese Metapher soll im wesentlichen beibehalten werden. Icons können aber auch Funktionen repräsentieren.

Die auf dem Bildschirm darzustellenden Objekte sind häufig groß und komplex. Man möchte nicht zu jeder Zeit jedes Objekt vollständig und detailliert auf dem Bildschirm sehen. Es gibt zwei grundlegende Methoden, dies zu lösen:

- *Filtern:*
  Es wird nur der Teil eines Objekts gezeigt, der gerade relevant oder interessant ist. In einem CAD-System sieht man den Teil der Graphik, der gerade bearbeitet wird. Man erhält einen Ausschnitt des Objekts. Innerhalb dieses Ausschnitts möchte man aber nicht jedes Detail der Graphik sehen, so daß man noch einen Filter "darüberlegt", der bestimmte Teile des Objekts unterdrückt. Durch diese Methoden des Filterns wird das Objekt unvollständig dargestellt.

[12] Mit dem Begriff Objekte sind hier Objekte der Anwendung (z.B. Dokumente, Ordner) gemeint; diese Objekte müssen nicht notwendigerweise programmiertechnisch als Objekte in einer objektorientierten Sprache realisiert sein.

- *Codieren:*
  Anstatt das gesamte Objekt oder Teile davon nicht darzustellen, kann man sie auch symbolisieren (codieren). Durch diese Abstraktion soll sich die Darstellung auf ihre in einem bestimmten Kontext wesentlichsten Charakteristiken reduzieren.

Icons wiederum sind nichts anderes als Codierungen von Objekten. Sie erhalten die Präsenz von Objekten mit einem Minimum an dargestellter Komplexität, und dies zeichnet sie aus. Daraus läßt sich schon eine Anwendung von Icons ableiten. Immer dann, wenn ein Objekt sichtbar bleiben soll, in seiner normalen Darstellung aber zu umfangreich oder komplex und darüber hinaus die Abstraktion seiner Details akzeptabel ist, kann man versuchen, es als Icon darzustellen.

Die Realisierung von Icons soll es erlauben, beliebige Objekte als Icons auf dem Bildschirm darzustellen. Zeigt man auf ein solches Icon mit einem Zeigeinstrument und wählt es aus, so hat man auf einfache Art und Weise ein Objekt ausgewählt. Dies ist der erste Schritt, um ein Objekt zu manipulieren. Visualisiert man durch solche Icons nicht irgendwelche Objekte, sondern Funktionen der Anwendungen, so lassen sie sich durch diese Zeigehandlung auswählen und ausführen. Erkennt man Texte auch als solche bildhaften Darstellungen an, so erkennt man in herkömmlichen Menüs nichts anderes als die codierte Darstellung von Funktionen und die Auswahlmöglichkeit über die Zeigehandlung.

Die Repräsentation von Objekten und Funktionen durch Icons (graphisch und textuell) auf dem Bildschirm und die dazugehörige Zeige- und Auswahlhandlung ist eine weitgehend anwendungsneutrale Kommunikationstechnik. In einer speziellen textuellen Form, nämlich Funktionsmenüs, wird sie bereits seit langer Zeit mit überzeugendem Erfolg angewendet. Die Verallgemeinerung durch graphische Elemente verspricht, eine interessante Perspektive im Hinblick auf die direkte Manipulation von Anwendungen sowie auf die Entwicklung weiterer anwendungsneutraler Benutzerschnittstellen zu sein.

Die beschriebenen Typen von Icons sind nur die einfachste Form. Man kann sich sehr viel komplexere Typen vorstellen, die verschiedene spezielle Eigenschaften haben. In unserem Projekt haben wir weitere Realisierungen in prototypischen Anwendungen untersucht. Die Icons wurden in ObjTalk als anwendungsneutrale Bausteine realisiert und sind mühelos in der Benutzerschnittstelle für Anwendungssysteme zu verwenden. In vielen prototypischen Anwendungen wurden sie in unserem Projekt bereits mit Erfolg verwendet.

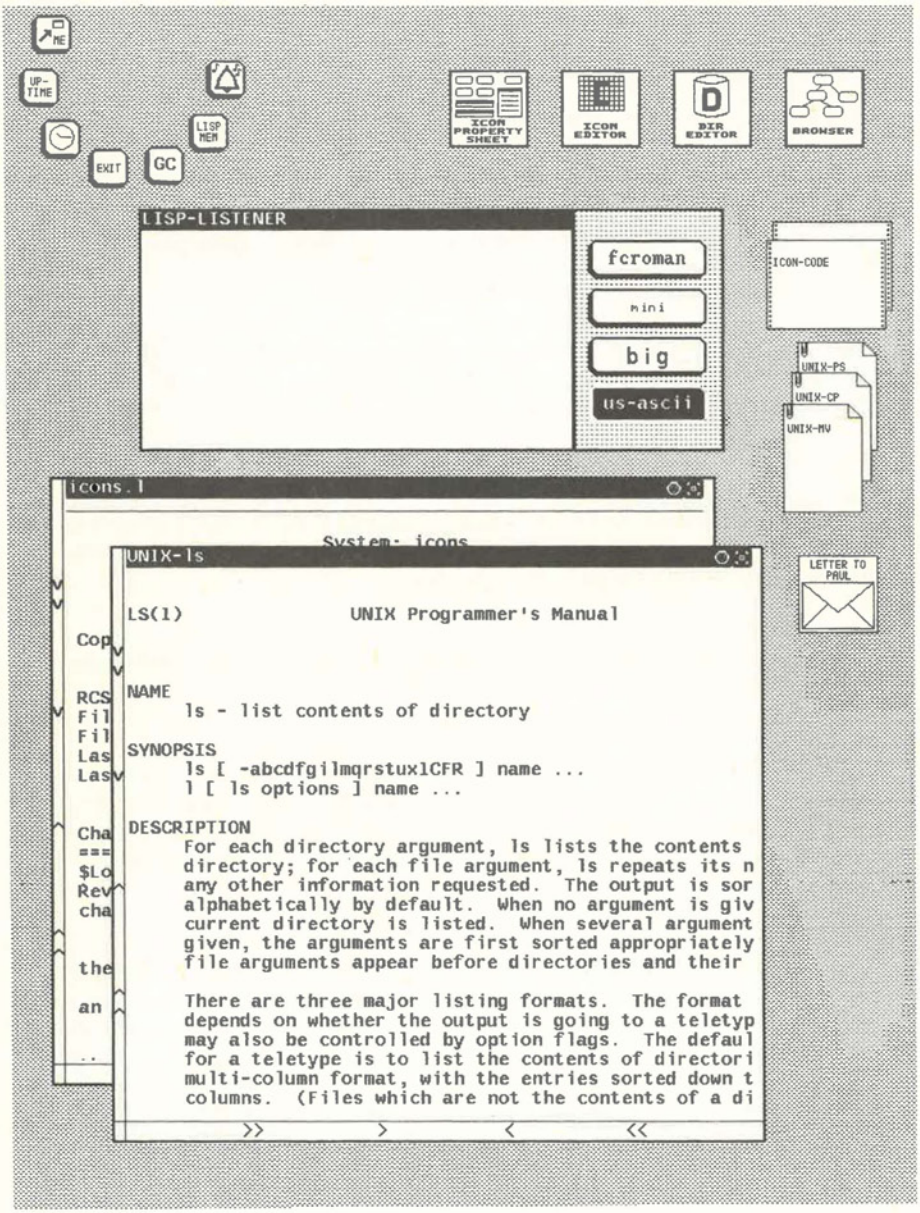


**Abbildung IV-12:** Beispiele von Icons auf dem Bildschirm

## 3.4 ObjTalk-Klassenhierarchie der Icons

Für den interessierten Leser wird im folgenden die vereinfachte ObjTalk-Hierarchie einer Icon-Implementierung graphisch dargestellt (siehe Abbildung IV-13) und erläutert:

- *elementary-icon:*
  Diese Klasse beschreibt die Grundeigenschaften von Icons. Ein Icon besitzt zum Beispiel die Referenz zum Objekt, das es repräsentiert, sowie eine Aktion, die es ausführt, wenn es aktiviert wird.

- *picture-icon:*
  Das *picture-icon* realisiert ein Icon, das neben den Grundeigenschaften des *elementary-icon* auch eine graphische Darstellung auf dem Bildschirm besitzt. Dies kann beliebige Rastergraphik sein.

- *text-icon:*
  Icons sind häufig nicht nur bildhafte Darstellungen, sondern können zusätzlich Texte als Beschriftung beinhalten. Dazu kennt das Icon u.a. den Text selbst, einen bestimmten Zeichensatz, die Lage des Textes im Icon sowie Methoden, um den Text zu formatieren und auszugeben.

- *basic-window:*
  Diese Klasse stammt aus dem Fenstersystem und definiert die Fenstereigenschaften[13] von Icons.

- *basic-icon:*
  Ein *basic-icon* kann sowohl eine graphische Darstellung wie auch einen Text enthalten. Dies ererbt es jeweils aus den Superklassen *picture-icon* und *text-icon*. Darüber hinaus hat es alle Eigenschaften eines Fensters, d.h. es kann beispielsweise auf dem Bildschirm herumbewegt werden und sich mit anderen Fenstern überlappen.

- *window-bound-icon:*
  Fenster nehmen auf dem Bildschirm oft beträchtlichen Platz ein. Benötigt man ein Fenster vorübergehend nicht mehr, so ist es wünschenswert, es zeitweise zu einem kleinen Icon zusammenschrumpfen zu lassen. Durch Anklicken des Icons möchte man das Fenster wieder in seiner alten Größe erscheinen lassen, während das Icon selbst verschwindet. *window-bound-icons* besitzen die Fähigkeit, mit einem beliebigen Fenster zu diesem Zweck kommunizieren zu können. Dieser Icontyp ist in unseren Systemen ein häufig benutzter Baustein der Benutzerschnittstelle.

- *document-icon:*
  Dieses Icon verfeinert die Eigenschaften von *window-bound-icon* nur geringfügig. Es bietet ein fest definiertes Bild eines symbolisierten Dokuments, um z.B. als Icon für Textfenster zu dienen.

---

[13] Siehe Kapitel V

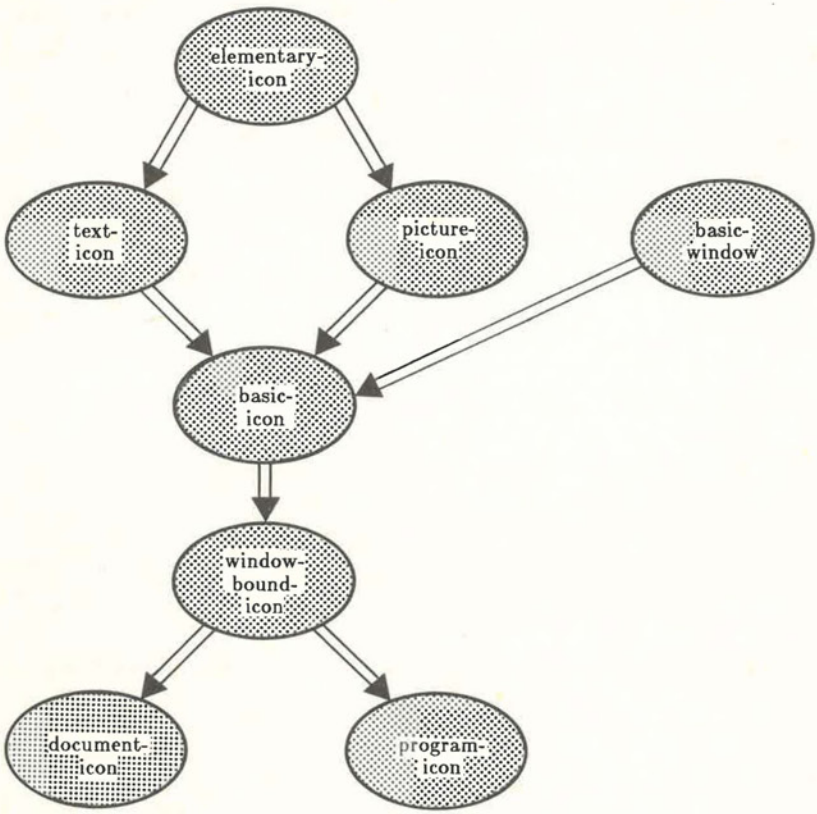


**Abbildung IV-13:** ObjTalk-Klassenhierarchie der Icons

- *program-icon:*
  Das *program-icon* leistet dasselbe wie das vorhergehende Icon, nur symbolisiert es ein Programmlisting, stellvertretend für ein Fenster, in dem Programmtext aufgelistet oder editiert wird.

Aus dieser in der Realisierung viel komplexeren und umfangreicheren Hierarchie kann sich der Anwendungsprogrammierer die für die spezielle Anwendung angemessenen Icontypen heraussuchen. Die Eigenschaften der Icons werden durch diese Hierarchie systematisch von sehr einfachen Icons mit geringer Funktionalität bis hin zu sehr speziellen Icons mit dezidierter Funktionalität verfeinert. Daran ist ersichtlich, daß die Konstruktion der Benutzerschnittstelle durch die Verwendung von Bausteinen nicht zu einer Armut an Kommunikationskonzepten führen muß. Eine verfügbare und leicht anwendbare Menge von

Bausteinen führt eher zur Bereicherung von Benutzerschnittstellen, da mit verhältnismäßig wenig zusätzlichem Aufwand solche Konzepte als Baustein integriert werden können.

## 4. Wissensbasierte Benutzerschnittstellen

Eine vollständige Benutzerschnittstelle, die nach dem Modell 4 aufgebaut ist, wird aus einem komplexen Netz sehr vieler Objekte bestehen. Neben den angedeuteten Objekten (z.B. Fenster, Icons, Menüs etc.) wird man noch eine Vielzahl anderer Objekte benötigen, um zum einen eine funktionelle, vollständige Benutzerschnittstelle zu erhalten und sie zum anderen mit mehr Wissen über verschiedene Dinge auszustatten [Bauer, Herczeg 85].

- Man kann *Metaobjekte* für Interaktionsmethoden (Menüs, Textfelder, Icons etc.) integrieren, um diese Methoden gestaltbar zu machen. Diese Metaobjekte sind nichts anderes als repräsentiertes *Wissen über die Eigenschaften der Benutzerschnittstelle.* Mit ihnen schafft man eine neue Ebene der Gestaltbarkeit *(adaptierbare Systeme)*, die über der Ebene der üblichen Programmierung liegt und damit für den Benutzer handhabbar sein kann [Herczeg et al. 85].

- Man kann Objekte bereitstellen, die *Wissen über die verschiedenen Benutzer* enthalten, um die Kommunikation den speziellen Bedürfnissen einzelner Benutzer anzupassen. Man spricht hierbei häufig vom *Benutzermodell*, dem Modell, das das Systems vom Benutzer hat. Ein solches Benutzermodell könnte beispielsweise enthalten, daß der Benutzer X wenig Systemerfahrung hat und daher Menüs bevorzugt, während Benutzer Y als Experte ausschließlich eine Kommandoschnittstelle erwartet. Derartige Benutzermodelle sind eine Voraussetzung für Systeme, die sich an den Benutzer selbständig anpassen *(adaptive Systeme).*

- Eine andere Art von Wissen, die in einer solchen Benutzerschnittstelle zu repräsentieren wäre, ist *Wissen über allgemeine ergonomische Bedürfnisse* der Benutzer. Es kann festgehalten werden, daß bei einem Farbsystem nur bestimmte Hintergrundfarben zulässig sind [Herczeg, Maier 83] oder die Größe von Schriftzeichen auf dem Bildschirm einen bestimmten Wert nicht unterschreiten darf.

- Merkt sich das System den Dialogablauf *(Dialoghistorie)*, so kann das System auf Wunsch des Benutzers eventuell zu früheren Systemzuständen zurückkehren *(Undo)* oder ausgeführte Aktionen in derselben oder leicht veränderten Form wiederholen *(Redo).* Wissen über die Abfolge von Systemfunktionsaufrufen, die Invertierbarkeit von Funktionen und das Verändern und Hinzukommen von Informationen sind dazu notwendig.

- *Wissen über die Hardware* (z.B. das verwendete Terminal) ist ein möglicher Ansatzpunkt, um oben genannten ergonomischen Forderungen Rechnung zu

tragen. Wenn das System beispielsweise in der Lage sein soll, dem Benutzer einen guten Kompromiß zwischen Schreib- und Hintergrundfarbe anzubieten, so muß es auch wissen, welche Farben auf dem Terminal darstellbar sind.

- Nicht zuletzt sollte *Wissen über die Anwendungssysteme* modelliert werden, um die Kommunikation zwischen Benutzerschnittstelle und Anwendungssystem so gut wie möglich zu gestalten. Dazu können Kommunikationsprotokolle definiert werden oder auch Quellen für ein Hilfesystem, das die Funktionen der Anwendung erklären soll.

Es gibt noch sehr viel mehr Wissen, das in der Benutzerschnittstelle modelliert werden sollte, um dem Benutzer eine angenehme und seinen Anforderungen adäquate Kommunikation mit verschiedenen Anwendungssystemen zu erlauben. Das Ziel ist eine Benutzerschnittstelle, die wir als wissensbasiert bezeichnen wollen.[14]

Die Isolierung verschiedener Eigenschaften der Benutzerschnittstelle in Form von Bausteinen ist ein erster Schritt zu einem solchen wissensbasierten System. Die Objekte und ihre Vererbungshierarchien dienen als Strukturierungsprinzip und machen das Wissen dieser Schnittstelle besser kontrollierbar. Objekte, die andere Objekte beschreiben (Metaobjekte), machen die Benutzerschnittstelle für Mensch und Computer gestaltbarer.

## 5. Schlußbemerkungen

Es gibt viele Gründe, die für die Modularisierung von Benutzerschnittstellen in möglichst anwendungsneutrale Bausteine sprechen. Ausgehend von einer Architektur des Anwendungssystems mit eng verflochtener Benutzerschnittstelle wurde anhand von vier Modellen der Übergang zu einer vom Anwendungssystem weitgehend losgelösten Benutzerschnittstelle in Form eines Netzwerks von kommunizierenden Objekten vollzogen.

Der objektorientierte Ansatz bringt - wie wir gesehen haben - viele Vorteile beim Bau solcher Benutzerschnittstellen:

- Die Aufteilung (der Benutzerschnittstelle) in Objekte ist eine "natürliche" Modularisierungsmethode.
- Die Koppelung von Objekten innerhalb der Benutzerschnittstelle ist auf vielfältige Weise möglich (vertikale und horizontale Schichtung, Synchronisation). Gute Strukturierung wird dadurch unterstützt.

---

[14] Man könnte sie auch intelligente Benutzerschnittstelle nennen, allerdings gibt es sehr viele Vorbehalte zu dieser Namensgebung.

- Individualisierung der Bausteine (durch das Konzept von Klassen und Instanzen sowie durch Vererbungshierarchien) ist sehr einfach möglich.

- Es gibt eine Vielzahl von Methoden für die Objekte der Benutzerschnittstelle, um mit dem Anwendungssystem zu kommunizieren (Funktionsaufruf, Senden von Botschaften, Constraints).

Am Beispiel von Icons wurde die Konzeption einer anwendungsneutralen Schnittstelle und die Realisierung als ein Vererbungsnetzwerk von ObjTalk-Klassen angerissen.

Nach der beschriebenen Architektur wird in unserem Projekt ein "Baukasten" entwickelt, mit dessen Hilfe man sehr schnell Benutzerschnittstellen zu beliebigen Anwendungssystemen erstellen kann. Die Bausteine werden für prototypische Anwendungssysteme verwendet und damit ständig durch kritische Bewertung weiterentwickelt oder verworfen. Neben den beschriebenen Icons gibt es weitere Elemente in diesem Baukasten:

- *Fenstersystem:*
  Das Fenstersystem[15] dient zur Unterteilung des Bildschirms in Fenster (rechteckige Bereiche, Teilbildschirme), die sich gegenseitig beliebig überlappen können [Fabian, Rathke 83]. Innerhalb dieser Fenster können sich z.B. die eigentliche Anwendung, Menüs und Icons befinden.

- *Menüs:*
  Das Menüsystem bietet eine Reihe verschiedenartiger Menütypen an, die bei Bedarf in ihrem Verhalten und Aussehen modifiziert werden können. Es gibt neben Standardmenüs zum Beispiel Pop-Up-Menüs (siehe 2.4.2), Scrolling-Menüs und mehrspaltige Menüs. Meistens sind Menüs mit bestimmten Fenstern oder den darin laufenden Anwendungen verknüpft.

- *Editoren:*
  Editoren[16] für unterschiedliche Arten von Texten sind ein wichtiger Bestandteil einer Benutzerschnittstelle. Als weitere Komponente wurde deshalb ein universeller, fensterorientierter Editor entwickelt, der leicht an verschiedene Anwendungen angepaßt werden kann.

- *Fonts:*
  Auf Rasterbildschirmen kann man verschiedene Zeichensätze (Schriftarten) darstellen. Die Nutzung dieser Möglichkeit kann die Benutzerschnittstelle entscheidend verbessern. Aus diesem Grund wurde eine Systemkomponente für den einfachen Umgang mit mehreren Zeichensätzen und einzelnen Bitmaps entwickelt.

---

[15] Siehe Kapitel V.

[16] Siehe Kapitel VI.

- *Formularsysteme:*
  In laufenden Arbeiten werden zur Zeit verschiedene Formularsysteme entworfen und realisiert [Herczeg 83]. Formulare sind eine altbewährte Interaktionstechnik, die durch die Verwendung von Rasterbildschirmen noch entscheidend verbessert werden kann.

- *Property-Sheets:*
  Zur Visualisierung und Modifikation der Eigenschaften von Objekten der Anwendung und der Benutzerschnittstelle dienen sogenannte Property-Sheets. Man kann sie als eine Mischung aus Menü und Formular betrachten. Sie dienen dem Benutzer zum Beispiel zur Modifikation von Icons auf dem Bildschirm und sind damit Metaobjekte zu Icons.

Mit diesen Ausführungen sollte nicht der Anschein erweckt werden, daß Benutzerschnittstellen völlig losgelöst von Anwendungssystemen betrachtet werden können. Dies ist genausowenig möglich, wie sie unabhängig vom Benutzer betrachtet werden können. Das Ziel ist in erster Linie, Methoden, die ständig in gleicher oder leicht veränderter Form Anwendung finden, so weit wie möglich zu isolieren und als individualisierbare Bausteine bereitzustellen. Das Ziel ist auch nicht die starre Standardisierung von Computerdialogen, denn dies würde zwangsläufig zu einer Verarmung der Kommunikation führen. Genau das Gegenteil ist anzustreben, eine vom Menschen handhabbare Bereicherung der Mensch-Computer-Kommunikation durch isolierte, kontrollierbare und leicht veränderbare Konzepte.

# Methoden und Werkzeuge zur Gestaltung benutzergerechter Computersysteme

Herausgegeben von
Gerhard Fischer und Rul Gunzenhäuser

Sonderdruck

Walter de Gruyter · Berlin · New York 1986