Eine ergonomische Benutzerschnittstelle für den Anwendungsbereich der Bildfolgenauswertung

Volker Haarslev, Universität Hamburg

Zusammenfassung: Es wird ein neuer Vorschlag zur Gestaltung der Benutzerschnittstelle von Bildfolgenauswertesystemen vorgestellt. Dieser Vorschlag stützt sich auf einer Modellierung der zukünftigen Benutzer einer solchen Schnittstelle und auf einer systematischen Untersuchung der Mensch-Maschine-Kommunikation für die Bildfolgenauswertung und führt zu einer Benutzerschnittstelle für eine ganze Klasse von Bildfolgenauswertesystemen, die Benutzern eine objektorientierte Interaktion und eine direkte Manipulation graphischer Repräsentationen der Komponenten dieser Systeme gestattet. Abschließend wird die Implementation eines Prototyps in der Programmiersprache Ada vorgestellt.

Einleitung

Seit einigen Jahren werden verschiedene Ansätze zur Deutung von digitalisierten TV-Bildfolgen zeitveränderlicher Szenen untersucht (siehe *Nagel (1985)*). Die dabei auftretenden Probleme führen zur Entwicklung umfangreicher experimenteller Programmsysteme, die oft durch die Mitarbeit zahlreicher Autoren entstehen und deren Modifikation und Erweiterung zahlreiche Mitarbeiter über viele Jahre in Anspruch nehmen kann. Als Beispiel sei das MORIO-System (*Dreschler-Fischer et al., 1983*) genannt. Die Komplexität derartiger Experimentalsysteme wird weiterhin dadurch erhöht, daß sie aus einer Kombination von peripheren Geräten wie Bildsignalquellen und -senken (z.B. Kamera, Bildplatte, Bildspeicher), graphischen Ein- und Ausgabegeräten, speziellen Prozessoren sowie Rechnern und Programmkomponenten bestehen.

Auf diese Weise gewachsene Systeme berücksichtigen nur selten konsequent ergonomische Prinzipien für die Systemgestaltung und die Benutzerschnittstelle (*Maaß, 1984*). Es droht somit die Gefahr, daß auf die Dauer nur noch ein kleiner Teil der verfügbaren Personalkapazität effektiv für die Weiterentwicklung derartiger Experimentalsysteme eingesetzt werden kann. Die hier vorgestellte Arbeit (*Haarslev, 1986*) untersucht, inwieweit Konzepte zur Gestaltung von Benutzeroberflächen, die in anderen Anwendungsbereichen (z.B. Textverarbeitung) entwickelt wurden, angepaßt und erweitert werden können, um eine für den Benutzer von Bildfolgenauswertesystemen angemessene Benutzerschnittstelle zu schaffen.

Bei Systemen zur Auswertung von Bildern und Bildfolgen handelt es sich vorwiegend um interaktive Systeme, die dem Benutzer eine mehr oder weniger umfangreiche Überwachung und Beeinflussung des Verarbeitungsablaufs gestatten. Die Interaktion zwischen dem Benutzer und experimentellen Bildauswertesystemen dient insbesondere der Festlegung der Verarbeitungsdaten, des Ablaufs und der Steuerung des Systems. Der Benutzer hat dabei die Funktion eines Experten, der den Verarbeitungsablauf überwacht. Gleichzeitig versucht der Benutzer, ein adäquates System zur Lösung seiner Fragestellungen zu entwickeln. Zwei wichtige Bestandteile bei der Entwicklung solcher Systeme sind somit die Interaktion und die Systemgestaltung.

Aufgrund der Erfahrung vieler Wissenschaftler finden das menschliche Denken und die damit verbundenen Problemlösungsprozesse überwiegend in visuellen Kategorien statt (*Benzon,*

1985, Raeder, 1985). Die Strukturierung von Experimentalsystemen ist somit ein Bestandteil der Systementwicklung und unterliegt damit auch dem menschlichen Denken. Um den Entwickler bei diesen Vorgängen zu unterstützen, erscheint es sinnvoll, das Experimentalsystem und seine Struktur graphisch darzustellen. Gleichzeitig kann sich die Interaktion des Benutzers an dieser Systemdarstellung orientieren und ihm damit einen Bezug zwischen seinen Eingaben und den Systemreaktionen vermitteln. Die Strukturierung des Systems erfolgt durch die direkte Manipulation (*Shneiderman, 1983*) der graphischen Darstellung des Systems.

Für den Benutzer sind aber nicht nur die statischen, sondern auch die dynamischen Aspekte eines Programmsystems von Interesse. Durch die graphische Darstellung der Wirkungsweise von Algorithmen (algorithm animation) wird dem Benutzer Einsicht in die dynamischen Aspekte des Systemablaufs gegeben. Als Beispiele sind die Programmumgebung BALSA (*Brown & Sedgewick, 1985*), die Lisp-Programmierumgebung TINKER (*Lieberman, 1984*) und ein auf der Smalltalk-Umgebung basierendes System (*London & Duisberg, 1985*) zu nennen.

Bei der Entwicklung einer Benutzerschnittstelle sind weiterhin nicht nur die visuellen und strukturellen Aspekte der Schnittstelle wichtig, sondern auch die konsequente Konzeption der Schnittstelle ausgehend von dem zukünftigen Benutzer. Diese Entwicklung wird erleichtert, wenn Konzepte der methodischen Programmentwicklung auf den Entwurf von Benutzerschnittstellen angewendet werden (*Draper & Norman, 1985*).

Nachfolgend werden die wichtigsten zugrunde gelegten Prinzipien bei der Entwicklung einer Benutzerschnittstelle für experimentelle Bildfolgenauswertesysteme zusammengefaßt:

- Explizierung und graphische Darstellung der Struktur von Bildfolgenauswertesystemen unter Verwendung eines Farbrastergraphik-Systems;
- Möglichkeit zur interaktiven Strukturierung und Veränderung eines solchen Systems durch die direkte Manipulation der graphischen Repräsentation von Systemkomponenten;
- Graphische Darstellung der Wirkungsweise von Auswertealgorithmen;
- Strukturelle Trennung eines Bildfolgenauswertesystems in ein Dialogsystem und ein Anwendungssystem;
- Entwurf einer interaktiven Benutzerschnittstelle für die Bildfolgenauswertung auf der Grundlage eines vorher explizit definierten Benutzermodells;
- Gestaltung der Benutzerschnittstelle auf der Basis von Objektkonzepten für die einzelnen Systemkomponenten.

Benutzermodell

Der erfolgreiche Entwurf einer Benutzerschnittstelle für ein System setzt voraus, daß klare Vorstellungen von Eigenschaften und Intentionen des zukünftigen Benutzers vorliegen, d.h. daß der zukünftige Benutzer dieses Systems modelliert wird. Dieses Benutzermodell wird dann als Grundlage verwendet, um die Interaktion zwischen dem Benutzer und dem geplanten System zu gestalten. Ein wichtiger Schritt bei der Modellierung des Benutzers ist die Formulierung eines konzeptuellen Modells, das der zukünftige Benutzer von einem System hat. Ein *konzeptuelles Modell* besteht aus einer Menge von Konzepten, die eine Person bildet, um das Verhalten eines

Systems zu erklären. Dieses vom Benutzer entwickelte Modell erlaubt ihm, die Reaktion eines Systems zu antizipieren und mit diesem System zu kommunizieren. Es wird hauptsächlich durch die Interaktionsnormen des Systems geprägt. Es kann deshalb auch als *Systemmodell* (*Maaß, 1984*) bezeichnet werden.

Als *Benutzermodell* werden hier die Vorstellungen des Systementwicklers bezeichnet, die dieser vom Benutzer und dessen Systemmodell hat.

Aus dem vom Entwickler formulierten Systemmodell werden Konzepte zur Gestaltung einer Benutzerschnittstelle entwickelt. Diese Gestaltungskonzepte explizieren das vom Systementwickler angestrebte Benutzermodell. Ihre Verwendung beim Entwurf eines Systems soll bewirken, daß im Benutzer bei der Interaktion mit dem System das angestrebte Systemmodell entsteht. Der Systementwickler setzt somit seine Erwartungen bezüglich der Art der Benutzereingaben und Systemausgaben in die Dialoggestaltung des Systems um.

Diese vom Entwickler im System niedergelegten Erwartungen werden als Benutzerbild des Systems (*Maaß, 1984*) angesehen. Das *Benutzerbild* bezeichnet hier die Vorstellungen über den Benutzer, die der Systementwickler im System verankert hat. Das Benutzerbild ist damit das Benutzermodell des Systementwicklers, das sich im System wiederfindet und vom Benutzer nur indirekt abgeleitet werden kann. Das Benutzerbild eines Systems äußert sich in den Interaktionsnormen, die der Systementwickler setzt (*Maaß, 1984*).

Das Spektrum der Bildfolgenauswertung erstreckt sich vom Bereich der Bildfolgenerfassung über die Bildvorverarbeitung bis hin zur 3D-Modellierung von Objekten und der Interpretation von Änderungen. Ein dieses Aufgabengebiet umfassendes System wird deshalb in natürlicher Weise aus Komponenten bestehen, die jeweils eine Teilaufgabe des Gesamtsystems realisieren. Die Interaktion des Benutzers mit dem System findet auf der Basis der vorhandenen Komponenten statt. Sie sollen deshalb auch als *Interaktionskomponenten* bezeichnet werden.

Als Systemmodell des Benutzers wird ein *Datenflußmodell* zugrunde gelegt, das ein Bildfolgenauswertesystem in eine Menge von *Interaktionskomponenten* zerlegt, die über *Datenleitungen* miteinander verknüpft werden. Die Interaktionskomponenten teilen sich auf in *periphere Geräte*, *Datenverwaltung* und *Programmkomponenten*. Das Systemmodell beschreibt weiterhin die Ziele des Benutzers während der Interaktion. Dabei lassen sich die die Ermittlung des aktuellen *Systemzustandes*, die *Hilfestellung* und die *Manipulation des Systems* unterscheiden.

Es wurden Konzepte entwickelt, die als Grundlage für die Gestaltung einer Benutzerschnittstelle dienen sollen, und die damit das im System verankerte *Benutzerbild* prägen. Die *graphische Darstellung* des Bildfolgenauswertesystems und seiner Komponenten soll beim Benutzer die Bildung eines Systemmodells unterstützen. Zur Interaktion mit dem System und zur graphischen Darstellung werden fortgeschrittene Ein- und Ausgabegeräte wie eine *Maus* und ein *Farbrastergraphik-Gerät* gefordert. Die Kommunikation des Benutzers mit dem System soll auf der Basis der dargestellten Zerlegung des Systems stattfinden. Durch die *objektorientierte Interaktion* und den *objektgebundenen Handlungskontext* wird dieser Vorgang für den Benutzer erleichtert. Das Konzept der *direkten Manipulation* von graphisch dargestellten Objekten wird durch eine weitgehend *modusvermeidende Interaktion* und eine ständige *Reaktionsbereitschaft* der Benutzerschnittstelle unterstützt. Die Forderung nach einer *konsistenten* und *einfachen* Schnittstelle soll

deren Handhabung erleichtern.

In *Haarslev (1986)* werden mehrere Systeme aus dem Bereich der Bildverarbeitung und -auswertung vorgestellt. Die Benutzerschnittstellen dieser Systeme orientieren sich überwiegend an Kommandos. Zur Bewertung der Benutzerschnittstellen wurde ein neues Klassifikationsschema entwickelt, das zwischen Interaktionsformen und Interaktionskonzepten unterscheidet. Als *Interaktionsform* werden die zur Durchführung einer Interaktionshandlung verwendete Technik und ihre Form der graphischen Darstellung bezeichnet. Ein *Interaktionskonzept* hingegen beschreibt die Bedeutung einer Interaktionshandlung, die der Benutzer zur Durchsetzung seiner Ziele vollführt.

Die Benutzerschnittstellen der untersuchten Systeme gestatten die Ableitung von vier Interaktionsformen. Die Form der *einfachen Kommandos* wird häufig verwendet und stellt sich aus der Sicht des Benutzers als textuelle Eingabe einer linearen Befehlsfolge dar. Eine Erweiterung tritt in Form von *Unix-Kommandos* auf, die einen interaktiven Aufbau einer Fließbandverarbeitung mithilfe von Filtern und Datenleitungen gestatten. Eine andere Form äußert sich durch *Menüs*, die alle verfügbaren Eingabealternativen anführen und dem Benutzer die Auswahl einer Option ermöglichen. Die vierte, häufig auftretende Interaktionsform besteht aus einem festen *Frage-Antwort-Schema.*

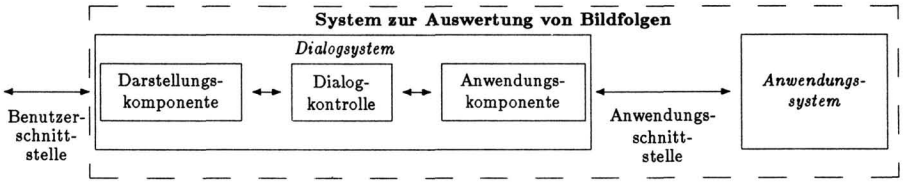
Die Benutzerschnittstellen der untersuchten Systeme lassen vier Interaktionskonzepte erkennen, die dem Benutzer die Durchsetzung seiner Interaktionsziele gestatten. Die *lokale* und *globale Ablaufsteuerung* dient der benutzerkontrollierten Beeinflussung des Systems. Die *Datenflußsteuerung* regelt die Angabe der Datenquellen und -senken und den Datenaustausch zwischen den Systemkomponenten. Die *Parameterinteraktion* verwendet der Benutzer u.a. zur lokalen (d.h. innerhalb einer Komponente) Steuerung des Ablaufs. Zur Überwachung des Verarbeitungsablaufs wird eine *Protokollinteraktion* angeboten. Zur Realisierung dieser Konzepte werden die beschriebenen Interaktionsformen verwendet.

Konzeption einer Benutzerschnittstelle

Die auf der Grundlage des Benutzermodells konzipierte Benutzerschnittstelle faßt die Interaktionskomponenten als eigenständige Objekte auf, die mit dem Benutzer direkt und ohne Beeinflussung anderer Komponenten in Wechselwirkung treten können und auf dem Bildschirm durch Piktogramme erscheinen. Der Benutzer kann den Status der Objekte abfragen und gegebenenfalls Parameter verändern. Hierzu muß jedes Objekt eine angemessene Hilfestellung anbieten. Die Benutzerschnittstelle unterscheidet zwischen *Verarbeitungsobjekten*, die die für die Bildfolgenauswertung benötigten Operationen ausführen, und *Datenobjekten*, die über *Datenleitungen* zwischen den Verarbeitungsobjekten ausgetauscht werden. Mithilfe von *Gruppenobjekten* kann der Benutzer das Erscheinungsbild des Systems interaktiv verändern und auf einer für ihn jeweils angemessenen Abstraktionsebene konfigurieren, benutzerspezifische Merkmale an Objekte bzw. Objektmengen binden und eine gleichzeitige Interaktion mit mehreren Objekten durchführen. Der Benutzer erhält somit eine graphische Systemdarstellung, die ihm ständig die gewählte *Konfiguration*, den *Status* der Objekte, den *Fortschritt* ihrer Verarbeitung und den momentanen *Datenfluß* anzeigt.

Es werden vier Konzepte zur Steuerung des Systems verwendet. Der *Systemablauf* wird

Abb. 1: Struktur eines Bildfolgenauswertesystems



durch die Aktivierung bzw. Deaktivierung von Objekten kontrolliert. Die *Datenflußsteuerung* erfolgt durch die Schaffung von Datenquellen und Datensenken sowie über Datenleitungen zur Verbindung von Objekten. Alle Objekte können *Parameter* und Voreinstellungen für deren Werte besitzen, die der Benutzer verändern darf. Die flexible Überwachung des Systems über *Protokollausgaben* wird durch spezielle Protokollobjekte möglich, die der Benutzer zur Erzeugung der Protokolle interaktiv in das Verarbeitungsnetz des Systems einbinden kann.

Zur Unterstützung der Interaktion bieten die Objekte von ihnen geführte *Interaktionshistorien*, die in einer *Systemhistorie* vereint werden. Es werden Operationen zur Inspektion, Veränderung und erneuten Ausführung der (Teil-) Historien angeboten. Die durch die Erfassung der Interaktionshistorien bereitgestellten Daten bilden eine Grundlage dafür, daß der Benutzer zur Durchführung komplexer Systemmanipulationen *graphische Ablaufpläne* formulieren und diese für eine *automatische Steuerung* des Gesamtsystems nutzen kann. Die automatische Überwachung und Fehlerbehandlung kann mithilfe von *Objektdämonen* durchgeführt werden.

Realisierung eines Dialogsystems

Für die oben konzipierte Benutzerschnittstelle wurde ein Dialogsystem für Bildfolgenauswertesysteme entwickelt. Als Grundlage für die Implementation des Dialogsystems wurden bekannte Vorschläge oder Systeme zur Verwaltung von Benutzerschnittstellen (user interface management system) analysiert. Die meisten dieser Systeme können in drei logische Komponenten unterteilt werden (*Green, 1984*). Diese Komponenten sind zwar nicht in allen Systemen explizit vorhanden, sie lassen sich aber meistens im konzeptuellen Entwurf wiederfinden. Dabei wird das Dialogsystem in eine Darstellungskomponente, eine Dialogkontrolle und eine Anwendungskomponente gegliedert (siehe Abbildung 1).

Die *Darstellungskomponente* ist für die externe oder graphische Darstellung gegenüber dem Benutzer verantwortlich. Sie erzeugt die graphische Ausgabe auf dem Bildschirm und liest die vom Benutzer bedienten Eingabegeräte.

Die *Dialogkontrolle* definiert die Struktur des Dialoges zwischen dem Benutzer und dem Anwendungssystem. Sie empfängt eine lineare Folge von Eingabewörtern von der Darstellungskomponente und eine lineare Folge von Ausgabewörtern von der Anwendungskomponente. Mithilfe dieser Daten wird eine Dialogsteuerung vorgenommen.

Die *Anwendungskomponente* definiert die Schnittstelle zwischen dem Dialog- und dem Anwendungssystem. Sie kann direkt mit dem Anwendungssystem kommunizieren. Dabei werden die Daten des Anwendungssystems in eine dem Dialogsystem angemessene interne Darstellung

überführt und umgekehrt die Aktionen des Benutzers auf die Anwendungsdaten abgebildet. Die Anwendungskomponente erzeugt aus der internen Darstellung der Anwendungsdaten Informationen, die die Dialogkontrolle bei der Durchführung ihrer Aufgaben unterstützen. Weiterhin führt sie eine semantische Überprüfung der Benutzereingaben durch.

Neben der detaillierten Struktur des Dialogsystems ist ein weiterer wichtiger Punkt die Schnittstelle zwischen dem Dialogsystem und dem Anwendungssystem. Die Struktur der Anwendungsschnittstelle wird entscheidend durch die Form des Daten- und Kontrollflusses zwischen dem Anwendungs- und dem Dialogsystem bestimmt (*Draper & Norman, 1985, Green, 1984, Hayes et al., 1985*). Der Datenfluß und -austausch kann sich entweder an den Bedürfnissen des Dialogsystems oder an denen des Anwendungssystems orientieren. Dabei ist eine Abstraktion auf Anwendungsniveau vorzuziehen, da die Kommunikation in Form von Einheiten beschrieben wird, die das Anwendungssystem vom Benutzer benötigt oder ihm vermitteln will. Die Abstraktion auf dem Niveau des Dialogsystems spezifiziert die Daten in Form von Einheiten aus der Benutzerschnittstelle.

Für den Kontrollfluß zwischen dem Anwendungs- und dem Dialogsystem, und damit für den Einfluß des Anwendungssystems auf den Dialogablauf, können drei Formen unterschieden werden.

In der ersten Form wird das Dialogsystem als Unterprogrammpaket angesehen. Die Kontrolle über den Dialogablauf liegt damit ausschließlich bei dem Anwendungssystem. Diese aus der Sicht des Anwendungssystems *interne Kontrolle* hat mehrere Nachteile. Die Anwendungsschnittstelle ist nur in Form von Unterprogrammaufrufen spezifiziert. Dadurch wird die konzeptuelle Trennung zwischen Dialog- und Anwendungssystem nicht im Anwendungssystem sichtbar. Das Dialogsystem hat keinen Einfluß auf den Dialogablauf, es kann immer nur lokal in Form von Unterprogrammen arbeiten. Eine fehlerhafte Benutzung des Dialogsystems von der Anwendung kann nur schwer vermieden werden.

Die zweite Form der Dialogkontrolle ist eine völlige Umkehrung der ersten Form. Das Anwendungssystem wird als Unterprogrammpaket angesehen. Dadurch wird die Kontrolle über den Dialogablauf nur durch das Dialogsystem ausgeübt. Diese aus der Sicht des Anwendungssystems *externe Kontrolle* besitzt die oben erwähnten Nachteile der internen Kontrolle nicht. Dafür ist bei der externen Kontrolle von Nachteil, daß das Anwendungssystem keinerlei Möglichkeit erhält, den Dialog zu beeinflussen. Weiterhin muß das Dialogsystem eine vollständige Spezifikation des Kommunikationsablaufs zwischen dem Benutzer und dem Anwendungssystem besitzen. Diese Spezifikation muß alle möglichen Benutzerinteraktionen vorsehen.

Die dritte Form gestattet sowohl dem Dialogsystem als auch dem Anwendungssystem einen Einfluß auf die Dialogkontrolle. Sie wird deshalb auch als *verteilte Kontrolle* bezeichnet. Dabei werden das Dialogsystem und das Anwendungssystem als kooperierende Prozesse angesehen, die über die Anwendungsschnittstelle eine verteilte Kontrolle des Dialogs ausüben. Die Nachteile der internen und der externen Kontrolle werden von dieser Form der Dialogkontrolle aufgehoben. Weiterhin können mit ihr die beiden ersten Kontrollformen nachgebildet werden.

Zur Dialogkontrolle gehört ebenfalls eine Kontrolle über die Form der Darstellung der Anwendungsdaten und die Reihenfolge von gültigen Benutzeraktionen. Bei der *expliziten Dar-*

stellungskontrolle spezifiziert das Anwendungssystem die Reihenfolge der Darstellung von Anwendungsdaten und die gültigen Folgen von Eingabeaktionen durch den Benutzer, während bei der *impliziten Darstellungskontrolle* von dem Anwendungssystem nur die Daten und gültigen Mengen von Benutzeraktionen angegeben werden. Jede Ordnung über die Reihenfolge der Aktionen und Datendarstellung wird durch das Dialogsystem induziert. Die implizite Kontrolle hat gegenüber der expliziten den Vorteil, daß das Anwendungssystem die Behandlung von Fehlern und Ausnahmebedingungen, die im Dialog mit dem Benutzer auftreten können, nicht vorsehen muß.

Als weitere Grundlage für die Implementation des Dialogsystems wurden bekannte Vorschläge und Verwaltungssysteme für Benutzerschnittstellen aufgrund ihrer Realisierung der Dialogkontrolle klassifiziert. Das Spektrum der dabei betrachteten Systeme und der von ihnen verwendeten Methoden reicht von *kontextfreien Grammatiken* (z.B. *Shneiderman (1982)*), *Zustandsübergangsdiagrammen* (z.B. *Wasserman (1985), Jacob (1985)*), *speziellen Dialogbeschreibungssprachen* (z.B. *Hayes (1984), Shu (1985)*) über eine *interaktive Dialoggestaltung* (z.B. *Buxton et al. (1983), Bass (1985)*) bis hin zur *Modellierung auf der Grundlage von Prozessen* (*Green, 1985, Cardelli & Pike, 1985, Olsen et al., 1985*).

Die *kontextfreien Grammatiken* werden meistens in einer erweiterten *Backus-Naur-Form* (BNF) beschrieben. Systeme, die eine BNF-Notation verwenden, definieren mithilfe einer Grammatik die gültigen Eingabeaktionen des Benutzers. Da Benutzerschnittstellen durch eine reine BNF-Notation nur ungenügend beschrieben werden können, besteht eine übliche Erweiterung darin, die Regeln der Grammatik mit semantischen Aktionen zu assoziieren. Diese Aktionen (meistens als Prozeduren formuliert) rufen das Anwendungssystem auf, um die vom Benutzer gewünschten Operationen auszuführen. Ein Beispiel dafür ist die Erweiterung der BNF-Notation zu einer *Aktionsgrammatik* (*Reisner, 1981*).

Durch die Verwendung von kontextfreien Grammatiken realisieren die Verwaltungssysteme eine externe Kontrolle des Dialogs. Damit sind die Möglichkeiten des Anwendungssystems, den Dialog zu beeinflussen, als äußerst gering anzusehen. Weiterhin ist von Nachteil, daß die zeitliche Abfolge der Benutzeraktionen implizit definiert wird. Bei genügend komplexen Schnittstellen wird eine BNF-Beschreibung sehr umfangreich und für den menschlichen Betrachter schwer zu handhaben. Eine verständliche Grammatik ist ferner von der guten Namensgebung der Meta-Symbole sowie von einer übersichtlichen Regelstruktur abhängig.

Die *Zustandsübergangsdiagramme* oder auch *Übergangsnetze* werden von Systemen verwendet, die die Benutzerschnittstelle als eine Sammlung von Zuständen modellieren. Die Aktionen des Benutzers bewirken Übergänge in dem Zustandsnetz.

Die Verwendung von Übergangsnetzen als formales Beschreibungsmittel bietet dieselben Vorteile wie die Verwendung kontextfreier Grammatiken. Gegenüber den Grammatiken definieren die Übergangsnetze eine explizite zeitliche Abfolge der Benutzeraktionen. Ebenso wie bei der vorherigen Methode dienen die Übergangsnetze einer externen Kontrolle des Dialogs. Sie sind besonders gut zur Beschreibung von linearen, alphanumerischen Schnittstellen geeignet. Die Erweiterungen zur Handhabung von graphischen Schnittstellen (siehe *Wasserman (1985)*) sind jedoch mühsam und wenig überzeugend. Für die Netze und Grammatiken gilt, daß bei Schnittstellen mit vielen vom Benutzer jederzeit durchführbaren Operationen (modusvermeidende Interaktion)

die formale Beschreibung sehr umfangreich und mit vielen Details behaftet ist. Analog zu den Flußdiagrammen für die Programmentwicklung besteht bei den Netzen die Tendenz, eine zu sehr am Detail und an Modi orientierte Schnittstelle zu entwerfen.

Ein weiteres wichtiges Problem bei beiden Ansätzen ist die Behandlung von gleichzeitigen Handlungen. Dies soll am Beispiel der oben konzipierten Benutzerschnittstelle verdeutlicht werden. Der Benutzer kann dort mit einem Objekt kommunizieren (z.B. Parameter besetzen), während andere Objekte ihre Darstellung auf dem Bildschirm aktualisieren. Diese Aktionen erfolgen sowohl gleichzeitig als auch unabhängig voneinander. Ein entsprechendes Übergangsnetz müßte aus mehreren Teilen bestehen, die parallel zueinander arbeiten können.

Es gibt einige Verwaltungssysteme, die dem Benutzer die *interaktive Gestaltung* einer Benutzerschnittstelle ermöglichen. Der Benutzer kann mithilfe eines speziellen Editors oder auch über ein Menüsystem die Bildschirmaufteilung und die Darstellung von Anwendungsdaten direkt in ihrer graphischen Repräsentation erstellen. Diese Verwaltungssysteme beschränken sich üblicherweise auf bestimmte Interaktionsformen und ihre Darstellung. Eine interaktive Erstellung und Erprobung von graphischen Benutzerschnittstellen stellt durchaus eine Alternative zur formalen Spezifikation dar. Diese Systeme haben jedoch meistens den Nachteil, daß sie nur wenige Interaktionsformen unterstützen (z.B. MENULAY (*Buxton et al., 1983*) für Menüs, KSBASS (*Bass, 1985*) für formularähnliche Masken).

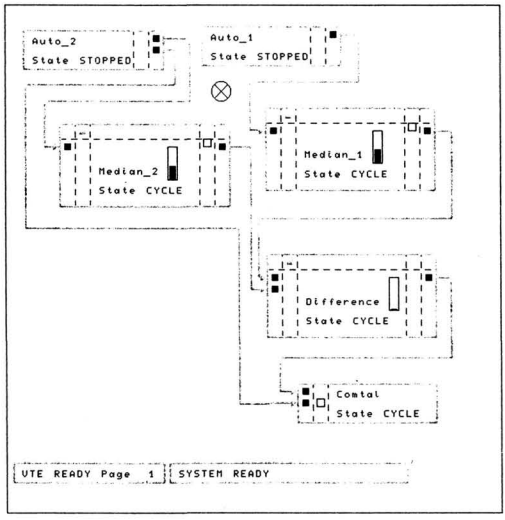
Die meisten der bisher vorgestellten Methoden oder Systeme zur Verwaltung von interaktiven Systemen haben im Rahmen der in Abbildung 1 skizzierten Struktur den Schwerpunkt auf die Dialogkontrolle gelegt. Die gerätenahe Behandlung von graphischen Ein- und Ausgaben sowie die flexible Einbettung unterschiedlicher Geräte (siehe auch *Rosenthal (1983)*) wird in diesen Systemen meistens nur sehr eingeschränkt behandelt.

Ein Konzept, welches sowohl eine detaillierte Dialogkontrolle als auch die Lösung der oben angedeuteten Schwächen erlaubt, modelliert ein Dialogsystem mithilfe von *Prozessen* bzw. *Objekten* (auch als *abstrakte Geräte* bezeichnet). Dieses Konzept besitzt eine starke Verwandtschaft mit dem von Smalltalk (*Goldberg & Robson, 1983*) begründeten objektorientierten Ansatz. Die Kommunikation dieser Prozesse kann synchron oder asynchron über Nachrichten bzw. Prozeduraufrufe erfolgen. Die asynchrone Kommunikation mit Nachrichten wird in dem Kontext der Verwaltungssysteme auch als *Ereignismodell* bezeichnet (siehe auch *Green (1984), Green (1985)*).

Die Beschreibung eines Dialogsystems auf der Grundlage von Prozessen hat den Vorteil, daß eine flexible Behandlung von graphischen Geräten möglich ist. Von Nachteil ist dagegen, daß diese Notation weniger formal als die Grammatik- oder Übergangsnetznotation ist. Die Verwendung von Prozessen gestattet dafür eine angemessene Behandlung von Bildschirmfenstern und eine Strukturierung der Fenster und Geräte in Form von Hierarchien (siehe auch *Rosenthal (1983)*). Die Verwendung formaler Methoden erscheint nur dann sinnvoll, wenn Systeme zur Handhabung derselben verfügbar sind. Als Beispiele seien das System RAPID/USE (*Wasserman, 1985*) und die in *Jacob (1985)* vorgestellte Programmumgebung für Übergangsnetze genannt.

Da zur Realisierung eines Dialogsystems für die oben beschriebene Schnittstelle derartige Werkzeuge nicht zur Verfügung standen, wurde das Dialogsystem mithilfe von Objekten implementiert. Die zu jedem Zeitpunkt möglichen Benutzereingaben werden aus der Sicht der Ob-

Abb. 2: Graphische Systemdarstellung



jekte als asynchron ausgelöste Ereignisse modelliert. Diese Entscheidung wird durch die Erkenntnis unterstützt, daß als Grundlage eines derartigen Verwaltungssystems eine asynchrone Ereignisverarbeitung mithilfe eines Nachrichtensystems empfehlenswert ist (*Olsen et al., 1984*). Weiterhin sollte die Implementationssprache des Verwaltungssystems diese Konzepte voll unterstützen.

Zur Implementation eines Prototyps wurde deshalb die Programmiersprache Ada* gewählt. Ada besitzt u.a. ein Prozeßkonzept, Pakete, generische Programmeinheiten sowie private Typen. Ein Vergleich zwischen Smalltalk-80 und Ada hat gezeigt, daß Ada nicht die Flexibilität von Smalltalk besitzt (*Nagl, 1984*). Dafür wird in Ada ein größeres Maß an Sicherheit und eine gute Unterstützung der methodischen Programmentwicklung angeboten. Ada ist deshalb ebenfalls wie Smalltalk sehr gut zur objektorientierten Programmierung geeignet.

Dieses Dialogsystem ist als ein Werkzeug im Rahmen einer objektorientierten Programmierumgebung für die Bildfolgenauswertung anzusehen (*Faasch & Haarslev, 1985*) und definiert eine für eine *ganze Klasse* von Anwendungssystemen *adaptive* Benutzerschnittstelle. Diese Anwendungssysteme können mit einem zweiten Werkzeug generiert werden, das eine objektorientierte Systemgestaltung von Experimentalsystemen unter Verwendung komplexer Daten- und Programmabstraktionen bietet (*Faasch, 1987*).

Dieser Prototyp beruht darauf, daß Anwendungssysteme über die vorhandene Anwendungsschnittstelle mit ihm kooperieren (siehe Abbildung 1). Der Kontrollfluß zwischen dem Anwendungs- und dem Dialogsystem basiert auf einer verteilten Kontrolle mit einer impliziten Darstellungskontrolle.

Abbildung 2 zeigt ein Beispiel für die durch den Prototyp realisierte graphische Systemdarstellung. Als Anwendungsbeispiel wurde das Problem der Erkennung von Änderungsbereichen

*Ada ist eine registrierte Handelsmarke der US-Regierung (Ada Joint Program Office)

zwischen jeweils einem Bildpaar aus einer Bildfolge — bestehend aus digitalisierten TV-Bildern — gewählt. Unter Änderungsbereichen werden dabei Bildbereiche verstanden, die Abbildungen von bewegten Objekten einer Straßenverkehrsszene sind. Eine ähnliche Problemstellung findet sich beispielsweise im MORIO-System (*Dreschler-Fischer et al., 1983*).

Die in Abbildung 2 dargestellte Konfiguration umfaßt sechs über Datenleitungen verknüpfte Anwendungsobjekte. Es handelt sich um zwei Objekte als Datenquellen, die das erste ("Auto 1") bzw. das zweite Bild ("Auto 2") eines Bildpaars bereitstellen, um drei Verarbeitungsobjekte, wobei zwei eine Medianfilterung auf einem Bild durchführen ("Median 1", "Median 2") und das dritte elementweise die betragsmäßige Differenz zweier Bilder mit einer anschließenden Binarisierung anhand eines Schwellwertes vornimmt ("Difference"), und um ein Objekt als Datensenke, das ein Rastersichtgerät zur überlagerten Darstellung eines Bildes und einer binären Maske der Änderungsbereiche repräsentiert. Die Darstellung der Objekte enthält u.a. den Objektnamen, auf der linken bzw. rechten Seite Leitungseingänge bzw. -ausgänge, einen Steuerparameter ("STATE") und evtl. einen Indikator (als Säule), der den Fortschritt der Verarbeitung anzeigt. Eine ausführliche Erörterung der graphischen Systemdarstellung ist in *Haarslev (1986)* zu finden.

Literatur

Bass (1985): *An Approach to User Specification of Interactive Display Interfaces*, L.J. Bass, IEEE Transactions on Software Engineering, Vol. SE-11, No. 8, Aug. 1985, pp. 686–698.

Benzon (1985): *The Visual Mind and the MacIntosh*, B. Benzon, Byte, Vol. 10, No. 1, Jan. 1985, pp. 113–130.

Borman & Curtis (1985): *Human Factors in Computing Systems*, L. Borman, B. Curtis (eds.), CHI'85 Conference Proceedings, April 14–18, San Francisco, 1985.

Brown & Sedgewick (1985): *Techniques for Algorithm Animation*, M.H. Brown, R. Sedgewick, IEEE Software, Vol. 2, No. 1, Jan. 1985, pp. 28–38.

Buxton et al. (1983): *Towards A Comprehensive User Interface Management System*, W. Buxton, M.R. Lamb, D. Sherman, K.C. Smith, ACM Computer Graphics, Vol. 17, No. 3, July 1983, pp. 35–42.

Cardelli & Pike (1985): *Squeak: a Language for Communicating with Mice*, L. Cardelli, R. Pike, ACM Computer Graphics, Vol. 19, No. 3, July 1985, pp. 199–204.

Draper & Norman (1985): *Software Engineering for User Interfaces*, S.W. Draper, D.A. Norman, IEEE Transactions on Software Engineering, Vol. SE-11, No. 3, Mar. 1985, pp. 252–258.

Dreschler-Fischer et al. (1983): *Lernen durch Beobachtung von Szenen mit bewegten Objekten: Phasen einer Systementwicklung*, L.S. Dreschler-Fischer, W. Enkelmann, H.-H. Nagel, 5. DAGM-Symposium Karlsruhe, 11.–13. Okt. 1983, H. Kazmierczak (Hrsg.), Mustererkennung, VDE-Fachberichte Nr. 35, VDE-Verlag Berlin Offenbach, 1983, pp. 29–34.

Faasch (1987): *Systemgestaltung einer Experimentalumgebung für die Bildverarbeitung in Ada*, H. Faasch, Universität Hamburg, Fachbereich Informatik, 1987 (in Vorbereitung).

Faasch & Haarslev (1985): *Konzeption einer neuen Ada-Programmierumgebung für die Bildfolgenauswertung*, H. Faasch, V. Haarslev, Mustererkennung 1985, 7. DAGM-Symposium Erlangen, 24.-26. Sep. 1985, Informatik-Fachberichte Nr. 107, H. Niemann (Hrsg.), pp. 191–196.

Goldberg & Robson (1983): *Smalltalk-80: The Language and its Implementation*, A. Goldberg, D. Robson, Addison-Wesley, Reading, Mass., 1983.

Green (1984): *Report on Dialogue Specification Tools*, M. Green, Computer Graphics Forum, Vol. 3, No. 4, 1984, pp. 305–313.

Green (1985): *The University of Alberta User Interface Management System*, M. Green, ACM Computer Graphics, Vol. 19, No. 3, July 1985, pp. 205–213.

Haarslev (1986): *Interaktion in Systemen zur Bildfolgenauswertung basierend auf einem objektorientierten Ansatz*, V. Haarslev, *Dissertation*, Universität Hamburg, Fachbereich Informatik, Juli 1986.

Hayes (1984): *Executable Interface Definitions using Form-based Interface Abstractions*, P.J. Hayes, *Technical Report* CMU-CS-84-110, Carnegie-Mellon University, Pittsburgh/PA, March 1984.

Hayes et al. (1985): *Design Alternatives for User Interface Management Systems Based on Experience with Cousin*, P.J. Hayes, P.A. Szekely, R.A. Lerner, In: *Borman & Curtis (1985)*, pp. 169–175.

Jacob (1985): *A State Transition Diagram Language for Visual Programming*, R.J.K. Jacob, IEEE Computer, Vol. 18, No. 8, Aug. 1985, pp. 51–59.

Lieberman (1984): *Seeing what your programs are doing*, H. Lieberman, International Journal of Man-Machine Studies, Vol. 21, No. 4, Oct. 1984, pp. 311–331.

London & Duisberg (1985): *Animating Programs using Smalltalk*, R.L. London, R.A. Duisberg, IEEE Computer, Vol. 18, No. 8, Aug. 1985, pp. 61–71.

Maaß (1984): *Mensch-Rechner-Kommunikation — Herkunft und Chancen eines neuen Paradigmas*, S. Maaß, *Dissertation*, Universität Hamburg, Fachbereich Informatik, Juli 1984. Auch erschienen als FBI-HH-B-104/84.

Nagel (1985): *Analyse und Interpretation von Bildfolgen*, H.-H. Nagel, Informatik-Spektrum, Vol. 8, No. 4, Aug. 1985, pp. 178–200.

Nagl (1984): *Ada und Smalltalk-80: Ein summarischer Vergleich*, M. Nagl, Technical Report, Osnabrücker Schriften zur Mathematik, Reihe I, Heft 16, Fachbereich Mathematik, Universität Osnabrück, 1984.

Olsen et al. (1984): *A Context for User Interface Management*, D.R. Olsen, Jr., W. Buxton, R. Ehrich, D.J. Kasik, J.R. Rhyne, J. Sibert, IEEE Computer Graphics and Applications, Vol. 4, No. 12, Dec. 1984, pp. 33–42.

Olsen et al. (1985): *Input/Output Linkage in a User Interface Management System*, D.R. Olsen, Jr., E.P. Dempsey, R. Rogge, ACM Computer Graphics, Vol. 19, No. 3, July 1985, pp. 191–197.

Raeder (1985): *A Survey of Current Graphical Programming Techniques*, G. Raeder, IEEE Computer, Vol. 18, No. 8, Aug. 1985, pp. 11–24.

Reisner (1981): *Formal Grammar and Human Factors Design of an Interactive Graphics System*, P. Reisner, IEEE Transactions on Software Engineering, Vol. SE-7, No. 2, Mar. 1981, pp. 229–240.

Rosenthal (1983): *Managing Graphical Resources*, D.S.H. Rosenthal, ACM Computer Graphics, Vol. 17, No. 1, Jan. 1983, pp. 38–45.

Shneiderman (1982): *Multiparty Grammars and Related Features for Defining Interactive Systems*, B. Shneiderman, IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-12, No. 2, March/April 1982, pp. 148–154.

Shneiderman (1983): *Direct Manipulation: A Step Beyond Programming Languages*, B. Shneiderman, IEEE Computer, Vol. 16, No. 8, Aug. 1983, pp. 57–69.

Shu (1985): *FORMAL: A Forms-Oriented, Visual-Directed Application Development System*, N.C. Shu, IEEE Computer, Vol. 18, No. 8, Aug. 1985, pp. 38–49.

Wasserman (1985): *Extending State Transition Diagrams for the Specification of Human-Computer Interaction*, A.I. Wasserman, IEEE Transactions on Software Engineering, Vol. SE-11, No. 8, Aug. 1985, pp. 699–713.

Dr. Volker Haarslev
Universität Hamburg
Fachbereich Informatik
Arbeitsbereich Kognitive Systeme
Bodenstedtstraße 16
D-2000 Hamburg 50