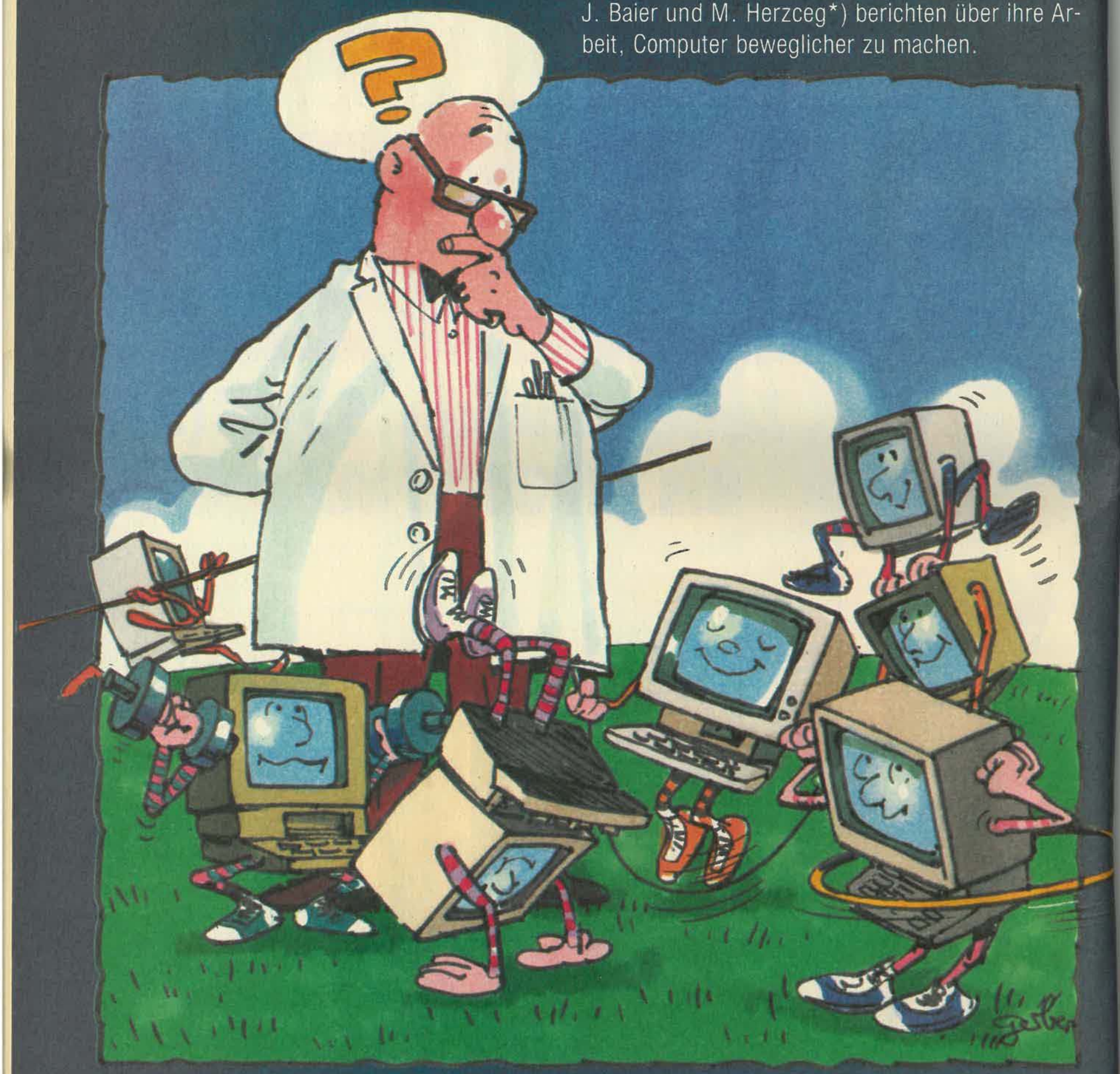
Nicht nur Endgerätehersteller, sondern auch Wissenschaftler widmen sich verstärkt der neuen Teildisziplin „Software-Ergonomie". Denn bei künftigen Kommunikations- und Informationssystemen wird der Computer sich zunehmend nach dem Benutzer und seinen Bedürfnissen richten müssen. J. Baier und M. Herzceg*) berichten über ihre Arbeit, Computer beweglicher zu machen.



Verstärkte Bemühungen um Benutzerschnittstellen

Computer auf dem Tr

Der Grund für die verstärkten Bemühungen um Benutzerschnittstellen liegt an deren zentraler Stellung als Bindeglieder zwischen Anwendungsprogrammen eines Computers und deren Benutzern. Sie definieren Regeln und Methoden der Kommunikation zwischen Benutzer und Computer. Die Eigenschaften von Benutzerschnittstellen beeinflussen entscheidend die Komplexität und Effizienz der Arbeitsabläufe. Sie bestimmen, wie stark sich der Benutzer an die Arbeit mit dem Computer anzupassen hat und wie groß die Belastung des Benutzers durch die Arbeit mit dem System ist. Die heutige und zukünftige Bedeutung von Computersystemen im Arbeits- und Privatleben und die Schwierigkeit, Benutzerschnittstellen zu spezifizieren, zu realisieren und zu erproben, machen deutlich, wie wichtig eine neue Wissenschaft ist, die sich mit diesen Schnittstellen befaßt. Software-Ergonomie muß unter anderem physiologische und informationstechnische Aspekte sowie das Arbeitsumfeld einbeziehen. Diese Aufgabe macht die Software-Ergonomie zu einer inderdisziplinären Wissenschaft, die zumindest Medizin, Psychologie, Informatik und Arbeitswissenschaften zu bemühen hat. Im folgenden soll durchleuchtet werden, welche Eigenschaften Benutzerschnittstellen haben sollten, die das Prädikat ergonomisch verdienen.

Der unterschiedliche Erfahrungsstand von Benutzern ist ein Kriterium, das die Gestalt von Benutzerschnittstellen beeinflussen sollte. Es gibt meist nicht den Benutzer eines Systems, sondern jeder Benutzer hat einen speziellen Kenntnisstand in der Bedienung des Systems sowie spezielles Fachwissen. Er hat bestimmte Fähigkeiten in der Aufnahme visueller Information und der Ausführung motorischer Tätigkeiten, wie der Bedienung einer Tastatur oder eines Zeigeinstruments (zum Beispiel Maus, Lichtgriffel). Außerdem haben unterschiedliche Benutzer meist unterschiedliche Aufgaben und Kompetenzen. All diese speziellen Eigenschaften sind nicht unveränderlich und damit nicht fest einplanbar. Benutzer entwickeln sich erst während der Arbeit mit dem System von unerfahrenen zu erfahrenen Benutzern, vielleicht sogar zu Experten. Die meisten Systeme sind auf die Eigenschaften ihrer Benutzer nicht abgestimmt. Die Folge davon ist am Anfang der Benutzung eine mehr oder weniger starke Überforderung, die häufig der Grund für die Ablehnung des Systems ist. Hat der Benutzer jedoch Erfahrung in der Benutzung gesammelt, stößt er oft schon früh an die Grenzen eines Systems. Der Eindruck von Eingeschränktheit und Monotonie bei der Arbeit kann dann kaum ausbleiben.

Was müssen wir also von zukünftigen benutzergerechten Systemen verlangen? Wir wollen Systeme, die sich an die speziellen Eigenschaften und Erfahrungen des jeweiligen Benutzers anpassen lassen. Solche adaptierbaren Systeme sollen ihm insbesondere Möglichkeiten zur Gestaltung der Benutzerschnittstelle anbieten. Wie inzwischen bekannt ist, eignen sich Menüs zur Auswahl von Aktionen mehr für Anfänger, während Experten häufig Kommandoschnittstellen bevorzugen, daß sie sich die Kommandos über lange Zeit sicher eingeprägt haben und auf ein Menü aller möglichen Kommandos gerne verzichten. Die Entscheidung zwischen Menü-Interaktion oder Kommandoschnittstellen könnte also vom Benutzer selbst gefällt werden. Die Methode der Informationskodierung, zum Beispiel die Verwendung von Farbe, kann vom Systemdesigner im voraus nicht sinnvoll festgelegt werden. Persönliche Präferenzen spielen hierbei eine große Rolle. Beispielsweise kann das Angebot einer Farbpalette zur Auswahl der Farben für die verschiedenen Objekte auf dem Bildschirm die Akzeptanz eines Systems wesentlich erhöhen.

Die Namensgebung für die Objekte und Aktionen des Systems muß nicht unabänderlich sein. Der jeweilige Benutzer hat oft einen geeigneteren Wortschatz im Rahmen seiner Tätigkeit, als dies der Systemdesigner in seiner relativen Unkenntnis der eigentlichen Anwendung haben kann. Wenn dies nicht der Ausführung der Tätigkeit auf andere Weise schadet, sollte der Benutzer diesen Wortschatz (etwa Synonyme) auch in das System einbringen und diesen dann verwenden können.

Oft gibt es (zum Beispiel in Formularen) häufig wiederkehrende Eingabetexte. Diese können sich zeitweise ändern, so daß sie nicht unabänderlich vorgesehen werden können. Der Benutzer sollte sie definieren können. Es gibt immerwiederkehrende Abfolgen von Aktionen mit dem System, die stark aufgaben- und benutzerspezifisch sind. Diese Abfolgen sollten abspeicherbar sein, um jederzeit wieder darauf zurückgreifen zu können, um sich so die ständige komplette Wiederholung zu ersparen.

Adaptierbare Systeme können dem Benutzer das Gefühl einer persönlichen Arbeitsumgebung vermitteln, die er sich nach seinen Fähigkeiten und Präferenzen gestaltet hat und jederzeit wieder umgestalten kann. Ein über derartige Möglichkeiten hinausgehendes Ziel sind sich an den Benutzer anpassende Systeme. Adaptive Systeme kontrollieren die Art und Weise der Interaktion eines bestimmten Benutzers, um ihn bei verschiedenen Problemen besser unterstützen zu können. Einige Beispiele automatischer Anpassung sind:

- Das System bemerkt, daß sich ein Benutzer ständig in der gleichen Art und Weise vertippt, und korrigiert dies dann. In Zukunft korrigiert das System diesen Tipfehler automatisch, sofern es eindeutig möglich ist.
- Der Benutzer macht ständig denselben Fehler bei der Verwendung eines Kommandos trotz bereits ausgegebener Hilfe-Information. Das System informiert ihn beim nächsten Mal ausführlicher über dieses Kommando und gibt ihm Beispiele.
- Sobald der Benutzer mit den für Anfänger angebotenen Funktionen sicher umzugehen gelernt hat, bietet ihm das System komplexere Funktionen an, die zwar schwieriger zu bedienen sind, aber schneller zum Ziel führen. Der Benutzer kann sich dann langsam weiterentwickeln, ohne durch die offengelegte gesamte Funktionalität von Anfang an überfordert zu werden.
- In einem Formularfeld verwendet der Benutzer ständig denselben Eingabetext. Das System bietet daraufhin diesen Text in dem Feld zur Bestätigung an und erspart dem Benutzer das wiederholte Eintippen.

Das Anbieten von Gehaltsfunktionen zur Adaptierbarkeit, die automatische Adaption sowie die Verwaltung benutzerabhängiger Unterschiede in der Präsentation des System bringen einige schwierige psychologische und technische Probleme mit sich. Systeme müssen Wissen sowohl über den Benutzer als auch über sich selbst besitzen, um zu diesen Leistungen fähig zu sein.

Arbeitet ein Benutzer mit einem System, so erwartet er, daß gleichartige Vorgänge durch gleichartige Interaktionen hervorgerufen werden können. Diese Verläßlichkeit des Dialogverhaltens ist eine Voraussetzung für unbelastetes Arbeiten. Ändern sich die Durchführungsbedingungen der Interaktion, so ist der Benutzer einem ständigen Anpassungszwang unterworfen, der ihn von der eigentlichen Arbeit ablenkt. Die innere Konsistenz von Systemen ist leider nicht immer gewährleistet, da das Verhalten des Systems häufig im Programm an mehreren Stellen festgelegt wird, die sich nur mühevoll konsistent halten lassen. Bei der Konzeption wissenbasierter Systeme wird darauf geachtet, daß das Wissen (und damit auch das Verhalten) des Systems in klar isolierbaren Teilen festgehalten ist, auf die von verschiedenen Stellen Bezug genommen werden kann. Die Konsistenz läßt sich dadurch leichter gewährleisten.

Schon heute arbeitet ein Computerbenutzer nicht nur mit einem einzigen Anwendungssystem. Viele, auch sehr unterschiedliche, Tätigkeiten werden durch dieselbe Person mit dem Computer erledigt. So kommt es beispielsweise vor, daß ein Büroangestellter mit einem Buchungs-, einem Auskunfts- und einem Textsystem arbeitet. Es ist durchaus nicht die Ausnahme, daß er sich dazu drei verschiedene Bedienungsmethoden aneignen muß; er lernt jedes System wieder nahezu von Grund auf. Die Konsistenz zwischen mehreren Anwendungssystemen ist ein großes Problem. Was für die Einheitlichkeit der Bedienung innerhalb eines Systems gilt, sollte natürlich auch systemübergreifend garantiert sein. Was wir eigentlich brauchen, sind integrierte Systeme, die sich derselben Interaktionstechnik, unabhängig von der Anwendung, bedienen. Dazu benötigt man weitgehend anwendungsneutrale Benutzerschnittstellen, die den Zugang zu allen benutzten Anwendungssystemen in derselben Art und Weise ermöglichen. Bekannte Ansätze dazu sind Masken- und Menügeneratoren. ▶

mmdich-Pfad

*)Die Autoren Joachim Bauer und Michael Herczeg sind wissenschaftliche Mitarbeiter der Forschungsgruppe INFORM am Institut für Informatik der Universität Stuttgart. Ihre Arbeitsschwerpunkte sind der Entwurf von Benutzerschnittstellen und Methoden zur Wissensrepräsentation. Die Forschungsgruppe INFORM wird im Rahmen des Verbundprojekts WISDOM vom Bundesminister für Forschung und Technologie gefördert.

Neue anwendungsneutrale Interaktionstechniken werden entwickelt. Da sind zum Beispiel die Systeme, die dem Benutzer Konsistenz mit der bekannten manuellen Arbeitsumgebung (metaphorische Systeme) anbieten. Dazu werden auf dem Bildschirm die gewohnten Arbeitsobjekte (Dokumente, Ordner, Schränke, Papierkorb) als kleine Bilder, sogenannte Icons, dargestellt. Die Bildschirmfläche stellt den Schreibtisch und die Arbeitsumgebung dar. Mit Hilfe eines Zeige-Instruments (üblicherweise eine Maus) können diese Objekte dann ausgewählt, bewegt und mit einem Editor bearbeitet werden. Die Wirkungen sind sofort sichtbar und damit kontrollierbar. Man spricht deshalb auch von direkter Manipulation oder dem WYSWYG-Prinzip (What You See Is What You Get). Diese Systeme müssen ständig wissen, wo sich welche Objekte auf dem Bildschirm befinden. Manipulation der Bildschirmobjekte muß sich entsprechend auf die dadurch repräsentierten Arbeitsobjekte niederschlagen. Das Bewegen eines Dokuments in den Papierkorb muß also auch die Löschung der entsprechenden internen Informationsstruktur bewirken.

Es gibt inzwischen eine Reihe bereits auf dem Markt angebotener Systeme, die sich derartige Metaphern zur Interaktion zunutze machen (zum Beispiel Xerox Star, Apple Lisa und Macintosh). Das Prinzip der direkten Manipulation ist schon seit längerer Zeit bei Computerspielen zu beobachten, die ihre Attraktivität im wesentlichen dieser Technik verdanken.

Die Konstruktion als wissensbasierte Systeme verhindert, daß metaphorische Systeme mehr scheinen, als sie wirklich sind. Sie verhelfen dem Benutzer zu einer angenehmen Arbeitsumgebung und kombinieren die Vorteile manueller Systeme mit den Vorteilen moderner Informationsverarbeitung. So erreichen wir leichte Bedienbarkeit und integrierte Lösung der Arbeitsaufgaben. Einen großen Teil seiner Zeit am Rechner verbringt der Benutzer damit, Fehler zu verbessern. Oft sind diese von einfacher Art wie Tippfehler, falsche Benennung von Schlüsselwörtern oder vertauschte Parameterreihenfolge. Eine andere Art von Fehlern sind konzeptionelle Fehler. Diese treten auf, wenn der Benutzer die Konzepte und Begriffe, auf denen ein System beruht, nicht versteht.

Konzeptionelle Fehler können durch einen klaren Aufbau des Systems und konsequente Verwendung derselben Begriffe für dieselben Konzepte verringert werden. Wenn es trotzdem zu Fehlern kommt, muß der Benutzer durchschauen können, warum diese auftraten. Das System muß die Ursache von Fehlern erklären können, muß fehlertransparent sein.

Viele Fehler treten nur deswegen auf, weil das System ziemlich starr ist und nur eine genau definierte Eingabeform akzeptiert. Sieht ein erfahrener Benutzer jemandem über die Schulter, der einen Fehler macht, sieht er in vielen Fällen schnell, wo der Fehler ist und kann ohne Rückfrage erklären, wie die Eingabe richtig ausgeführt werden müßte. Diese Hilfe durch einen erfahrenen Benutzer sollte eigentlich nicht nötig sein. Viele Tippfehler kann auch das System selbst korrigieren. Auch Dinge wie in falscher Reihenfolge eingegebene Parameter oder vergessene Teilaktionen können vom System gut erkannt werden. Das System soll fehlertolerant sein. Wenn nicht automatisch korrigiert werden kann, so bringt eine flexible Fehlerbehandlung doch noch den Vorteil, daß der Benutzer gezielt nur seinen Fehler verbessern muß und nicht eine ganze Eingabefolge. Dieses gezielte Korrekturangebot macht den Fehler auch transparenter.

Fehler können dadurch vermieden werden, daß es keine starren Eingabereihenfolgen gibt, es sei denn, sie sind unbedingt nötig. Dort, wo starre Reihenfolgen notwendig sind, etwa weil für eine Aktion bestimmte Voraussetzungen erfüllt sein müssen), sollte der Benutzer vom System geführt werden. Falsche Eingaben sollten möglichst frühzeitig (nicht erst nach Eingabe einer langen Eingabefolge) erkannt werden und zu Erklärungen führen. Das System sollte zunächst kurze und prägnante Fehlermeldungen geben und dem Benutzer weitere Hilfe

anbieten. Bei der Beantwortung seiner Fragen muß der Systemzustand während der Fehlersituation mitberücksichtigt werden.

Bei wissensbasierten Systemen sind Aktionen nicht unbedingt in fester Reihenfolge einprogrammiert, sie können als Erfordernisse spezifiziert sein, die der Benutzer nach und nach erfüllt. Wenn Voraussetzungen nicht erfüllt sind, führt dies nicht zu einem Fehler, sondern das System erfragt sie. Durch die explizite Repräsentation aller Aktionen und Wissen darüber, in welchen Situationen diese ausgelöst werden können, stehen alle möglichen Aktionen immer zur Verfügung, so daß eine automatische Korrektur möglich ist oder die Initiierung momentan nicht erlaubter Aktionen verhindert werden kann. Wissen über Konzepte und den Ablauf von Aktionen kann dazu verwendet werden, Fehler zu erklären. Wissensbasierte Systeme erleichtern nicht nur die Fehlerbehandlung, sondern verhindern Fehler durch konsistenten Aufbau, Anpassung an den Benutzer und Hilfeleistungen.

Der Dialog mit einem System sollte leicht verständlich sein. Da man aber aus einem Namen oder einem Icon die ihm zugeordnete Aktion oft nur erahnen kann und sich nicht alle Details einer Aktion in ihrer Bezeichnung widerspiegeln können, sind weitere Hilfeleistungen durch das System unumgänglich. Fragen nach der Existenz bestimmter Aktionen kommen bei Anfängern am häufigsten vor. Geübte Benutzer wissen von der Existenz der meisten Aktionen, erinnern sich aber nicht mehr daran, wie man diese auslöst. Wenn man eine neue Aktion entdeckt hat (sie können in einem Menü stehen) oder Details einer Aktion wissen. will, die man schon grob kennt, muß das System eine Beschreibung der Aktion geben können. Schwierige Situationen beim Arbeiten mit dem Computer sind Anfragen des Computers, die man nicht ganz versteht und bei denen man nicht so recht weiß, was man eingeben kann. In diesen Fällen sollte das System Auskunft über den momentanen Systemzustand geben können: Wie kam man dorthin? Welche Benutzeraktionen erwartet das System? Oft denkt ein Benutzer gar nicht daran, daß eine bestimmte Aktion möglich sein könnte. Hier sollte das System selber aktiv werden und Ratschläge geben. Ein von uns entwickeltes, aktives Hilfesystem stellt zum Beispiel fest, wenn ein Benutzer eine Zeile immer umständlich Zeichen für Zeichen löscht, und schlägt ihm das Kommando "lösche-Zeile" vor.

Weitgehende Hilfeleistungen erfordern viel Wissen im System. Fragen nach der Existenz von Kommandos lassen sich aufgrund von Wissensbasen, in denen verschiedene Aktionen konzeptionell zusammengefaßt sind, besser beantworten als aus einer Ansammlung von mehr oder weniger unabhängigen Unterprogrammen. Konzeptionell gegliedertes Wissen erleichtert die Suche nach einer bestimmten Information. Erklärungen können bei entsprechender Darstellung des Wissens teilweise aus der Wissensbasis erzeugt werden. Das Problem der Inkonsistenz zwischen Programm und Erklärung, das auftritt, wenn Programmänderungen durchgeführt, aber nicht dokumentiert werden, entfällt dann. Wenn Wissen über den Ablauf von Aktionen explizit repräsentiert ist, lassen sich Erklärungen über die einzelnen Dialogschritte leicht erzeugen.

Aus den bisherigen Ausführungen geht hervor, daß ein System, das benutzergerecht sein soll, ausreichendes Wissen benötigt. Seine Wissensbasis wird sowohl zur Beantwortung von Benutzerfragen, als auch zur Steuerung des Systemablaufs verwendet. Im folgenden fassen wir das Wissen, das diese enthalten müßte, noch einmal zusammen:

Wissen über die Interaktion

Das System sollte die grundlegende Interaktionsform und die für eine bestimmte Aktion nötigen Interaktionsschritte kennen. Bei unseren experimentiellen Systemen dient der linke Mausknopf zum Selektieren eines Bildschirmobjekts, der rechte Mausknopf aktiviert immer das zu einem Bildschirmobjekt gehörende Menü.

Wissen über die Funktionalität

Das System sollte Wissen darüber haben, welche Funktionen es gibt, in ▶

Fortsetzung von Seite 25

welchem Zusammenhang diese von Bedeutung sind, was sie bewirken und wie sie aktiviert werden können.

Wissen über den Systemzustand

Dies beinhaltet Wissen über die Dialoggeschichte und den augenblicklichen Systemzustand.

Wissen über den Anwendungsbereich

Damit bezeichnen wir Wissen über den Arbeitsbereich, in dem ein System zum Einsatz kommt. Es beinhaltet Wissen über geltende Konventionen (zum Beispiel „wie sieht üblicherweise ein Briefkopf aus?") und über Zustände, Ereignisse und deren Verknüpfung (zum Beispiel „ein Brief an den Chef geht immer über die Sekretärin").

Wissen über den Benutzer (Benutzermodell)

Für adaptive und adaptierbare Systeme ist Wissen über den Benutzer Voraussetzung. Fehlerbehandlung und Hilfeleistungen sind besser, wenn sie den individuellen Benutzer berücksichtigen. Zum Benutzermodell gehören die Informationen, zu welcher Benutzergruppe er gehört (Programmierer, Sekretärin, Manager), welche Ziele er hat (etwa bei Editiergeräten: einen Brief, ein Dokument oder ein Programm erstellen), welche Erfahrungen mit bestimmten Systemen oder Systemtypen er hat, was er über das System und den Anwendungsbereich weiß und welche Vorlieben er hat (zum Beispiel, ob er lieber mit Menüs arbeitet oder eine Kommandoschnittstelle bevorzugt).

Eine gute Möglichkeit, wissensbasierte Systeme zu erstellen, bieten objektorientierte Programmiersprachen. Bei diesen wird das vorhandene Wissen in Objekten zusammengefaßt. Objekte sind aktive Datenstrukturen mit Eigenschaften, die entlang einer Generalisierungshierarchie von einem Objekt auf andere vererbt werden können. Wenn ein Objekt eine Nachricht erhält, reagiert es darauf. Dies kann in der Veränderung seines lokalen Zustandes, dem Senden von Nachrichten an weitere Objekte oder der Rückgabe einer Antwort bestehen.

Die Konstruktion von ergonomischen Systemen steckt noch in den Anfängen. Sofern ergonomische Richtlinien in Systeme einfließen, geschieht dies meist nur in Teillösungen, da sich die Ziele mit herkömmlichen Programmiertechniken und den vorhandenen Hardwareeinschränkungen nicht oder nur mit sehr großem Aufwand erreichen lassen. Durch verbesserte Hardware (schnellere CPUs, hochauflösende Bildschirme, Zeige-Instrumente etc.) und den Bau von wissensbasierten Systemen können wir diesem Ziel entscheidende Schritte näherkommen.