Ansatz eines Dialogmanagers für ein intensivmedizinisches Informationssystem

B. Thull, M. Langen, G. Rau
Helmholtz-Institut für Biomedizinische Technik an der RWTH Aachen

Zusammenfassung

Die Gestaltung der Benutzerschnittstelle für ein medizinisches Informationssystem stellt insbesondere an die Auslegung der Dialog- und Interaktionssequenzen hohe Anforderungen. Es wird ein Ansatz für einen Dialogmanager vorgestellt, der die Definition komplexer Dialoge unterstützt, Mechanismen für eine Benutzerführung auf der Basis einer Farbkodierung anbietet und schließlich eine Analyse von Dialogen an einer Benutzerschnittstelle mit Hilfe graphentheoretischer Algorithmen ermöglicht. Am Beispiel einer Kontrolleinheit für eine Infusionspumpe wird der beschriebene Ansatz erläutert. Nach den bisherigen Erfahrungen haben sich vor allem die Möglichkeit zur einfachen Definition komplexer Dialoge und die automatisierte Benutzerführung bewährt. Der graphentheoretische Ansatz zur Analyse von Dialogen muß für einen praktischen Einsatz bei der Gestaltung noch weiter ausgebaut werden.

1. Einleitung

Die Entwicklung eines Informationssystems für die Intensivstation oder den OP-Bereich erfordert die Berücksichtigung einiger besonderer Randbedingungen des medizinischen Umfelds (/Bernotat & Rau, 1980/, /Rau & Trispel, 1982/). Ein solches Informationssystem muß in der Lage sein, große, vielschichtige und inhomogene Informationsmengen (Patientendaten, Vitalparameter, Bilder, usw.) schnell und zuverlässig in einer geeigneten Form zur Verfügung zu stellen. Auf der anderen Seite haben Ärzte oder das medizinische Personal in der Regel nur wenig Erfahrung im Umgang mit komplexer Informationstechnik. Die Gestaltung der Benutzerschnittstelle für ein medizinisches Informationssystem beeinflußt daher entscheidend den effektiven Einsatz des Systems. Sowohl die Ergebnisse aus einem früheren Projekt zur Gestaltung eines Anästhesie-Informationssystems (AIS) für kardiochirurgische Operationen (/Klocke et al., 1986, 1987/) als auch laufende Arbeiten an einem Informationssystem für eine kardiologische Intensivstation zeigen, daß eine Integration von Darstellung und Manipulation aller nötigen Daten auf einem Monitor ein geeigneter Lösungsansatz ist. Dazu werden hochauflösende Farbgrafik und multimodale Interaktion (Berühreingabe, Spracheingabe, usw.) nach den Prinzipien der direkten Manipulation (/Shneiderman, 1982/) eingesetzt.

Die Gestaltung und Implementierung der Interaktions- und Dialogformen ist insbesondere bei direkt-manipulativen Benutzerschnittstellen sehr aufwendig (/Wilson & Rosenberg, 1988/). Zu-

sätzlich führen die Anforderungen des medizinischen Umfelds zur Einführung spezieller Interaktionsformen (z.B. modale Formen wie erzwungene Bediensequenzen, Quittier-Tasten, usw.) und zur Strukturierung großer Datenmengen in einzelne Arbeitsseiten, Ausschnitte oder Fenster, die einen hohen Interaktionsaufwand zur Kontrolle des Informationssystems benötigen. Daher können in einem umfassenden Informationssystem verzweigte, teils modale Interaktionssequenzen entstehen, die ein komplexes Dialogmanagement erfordern. Für den Entwickler der Benutzerschnittstelle sind dabei folgende Probleme zu bearbeiten:

- ein komplexes Dialogmanagement muß einfach und übersichtlich definiert werden können,
- die Konsistenz der Benutzerschnittstelle aus der Sicht des Benutzers muß gewährleistet sein (einheitliche Interaktionsformen und Systemreaktionen),
- Fehler im Dialogmanagement (wie Dialog-Sackgassen, fehlende Undo-Dialoge, usw.) müssen vermieden bzw. entdeckt werden können.

2. Beispiel: Kontrolleinheit für eine Infusionspumpe

Der im folgenden vorgestellte Ansatz für einen Dialogmanager soll am Beispiel des Entwurfs einer Kontrolleinheit für eine Infusionspumpe erläutert werden. Diese Kontrolleinheit ist ein Baustein, der z.B. beim Entwurf eines Informationssystems für eine Intensivstation eingesetzt werden kann. Er soll dazu dienen, den Verlauf der Infusion eines bestimmten Medikamentes zu überwachen und die Einstellung der dazu eingesetzten Infusionspumpe zu steuern (Förderrate, Alarme, usw.).

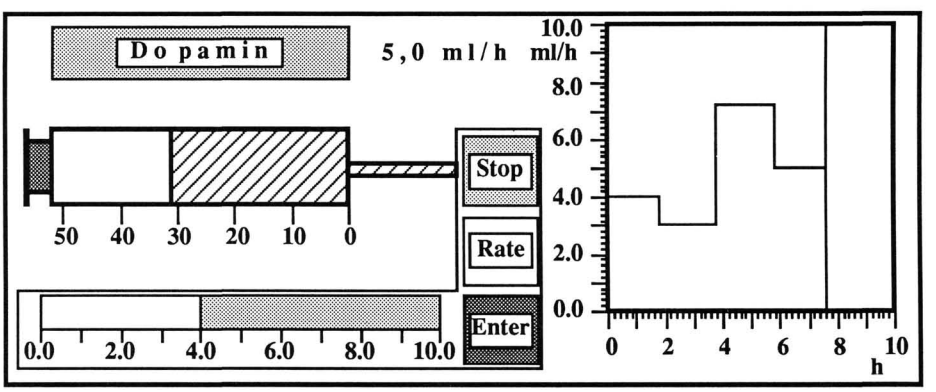

Abb. 1: Auslegung einer Kontrolleinheit für eine Infusionspumpe

Abbildung 1 zeigt den Entwurf für die Kontrolleinheit. Das verabreichte Medikament (Dopamin) wird in der Medikamentenanzeige zusammen mit der zur Zeit eingestellten Förderrate (5.0 ml/h) dargestellt. Die noch vorhandene Menge an Infusionslösung ist in der schematisch angedeuteten

Spritze angezeigt. Die Grafik auf der rechten Seite zeigt den Verlauf der verabreichten Raten für die zurückliegenden zehn Stunden. Mit der STOP-Taste kann die Infusion angehalten und wieder gestartet werden. Eine neue Förderrate wird eingestellt, indem zunächst die RATE-Taste angewählt (durch Anklicken mit der Maus oder Berühren mit dem Finger), danach der neue Wert mit Hilfe des virtuellen, analogen Schiebers unten links eingestellt und abschließend der eingestellte Wert über die ENTER-Taste bestätigt wird.

Die Benutzerschnittstelle wird mit Hilfe einer am Helmholtz-Institut entwickelten, objektorientierten Entwicklungsumgebung hierarchisch aufgebaut (/Langen et al., 1989/). Dabei wird eine Schnittstelle mit Hilfe von "Konstruktoren" aus elementaren Interaktionsobjekten wie Tasten, Schieber, Grafiken usw. zu (komplexen) Teilschnittstellen zusammengesetzt, die wiederum in neue Teilschnittstellen eingebettet werden können. Das Ergebnis ist eine hierarchische Struktur der Schnittstelle aus Elementarschnittstellen und zusammengesetzten (komplexen) Teilschnittstellen.

Abbildung 2 zeigt die hierarchische Struktur der hier vorgestellten Kontrolleinheit für eine Infusionspumpe.

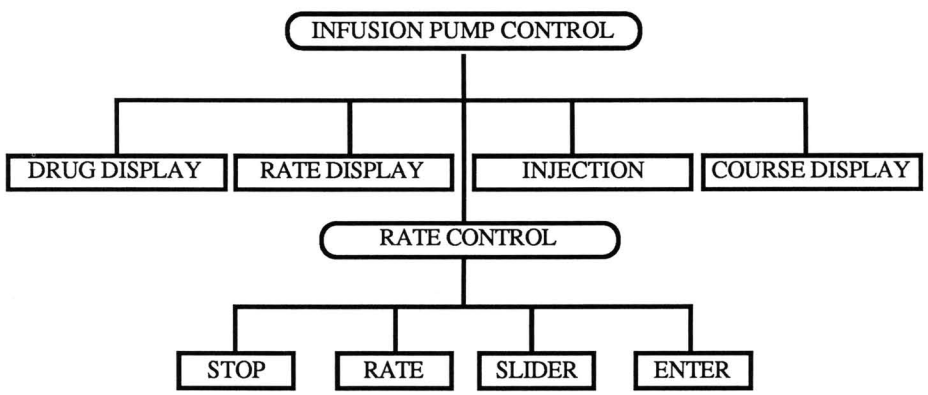

Abb. 2: Hierarchische Struktur für die Kontrolleinheit der Infusionspumpe

3. Ansatz: Erweiterung des Event-Modells

Die Funktionsweise der in Abbildung 1 dargestellten direkt-manipulativen Benutzerschnittstelle basiert auf dem Event-Modell (/Green, 1984/). Jede Funktion der Schnittstelle wird durch genau definierte Ereignisse (Events) ausgelöst. Diese Ereignisse können auf viele Arten erzeugt werden: Führen und Klicken einer Maus, Berühren einer Stelle auf dem Bildschirm mit einem Finger, Drücken von Tasten auf der Tastatur, Spracheingabesignale usw. Die Definition beliebiger Ereignisse zum Auslösen von Funktionen eröffnet einen großen Spielraum für die Gestaltung der Mensch-Maschine-Interaktion. Allerdings erfordern die vielfältigen Interaktionsmöglichkeiten eine

geeignete Benutzerführung, die für jedes elementare Interaktionsobjekt anzeigt, in welchem Aktivierungszustand es sich momentan befindet, also z.B. ob es zur Zeit bedienbar ist oder nicht bzw. ob es gerade bedient (aktiviert) wurde. Weiterhin sollte das Interaktionsobjekt auf eine Benutzereingabe mit einer Rückmeldung (Feedback) reagieren, um stets den aktuellen Zustand zu visualisieren und so eine Benutzerführung zu ermöglichen. Dies kann ein Farbumschlag, ein Signalton oder eine andere Reaktion sein. Mit Hilfe von Automaten, die in den elementaren Objekten implementiert sind, wird die Benutzerführung realisiert. Ein Interaktionsobjekt kann drei Aktivierungszustände annehmen: STANDBY als der Zustand, in dem das Objekt aktiviert werden kann, ACTIVE als der Zustand, in dem sich ein angesprochenes Objekt befindet und schließlich LOCKED als der Zustand, in dem ein Objekt vom Benutzer nicht bedient werden kann.

Als ein Beispiel für ein Interaktionsobjekt und sein Verhalten wird im folgenden ein virtueller, analoger Schieber vorgestellt, der über einen berührempfindlichen Bildschirm manipuliert werden kann.

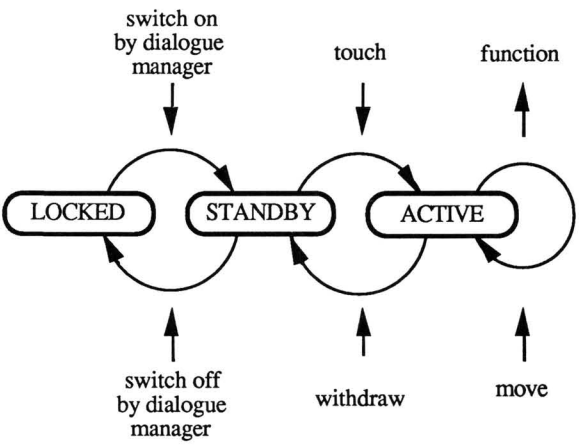

Abb. 3: Automat für einen virtuellen Schieber

Abbildung 3 zeigt den Automaten für einen virtuellen Schieber, mit dem der Benutzer einen Wert einstellen kann, indem er mit dem Finger auf dem Schieber entlangfährt. Der Benutzer aktiviert den Schieber durch die erste Berührung. Die Bewegung des Fingers auf dem Schieber resultiert in einer Schleife über den Zustand ACTIVE, in der der Schieber eine Funktion, wie z.B. das Aktualisieren eines bestimmten, einzustellenden Wertes, wiederholt durchführt. Der Schieber wird in den Zustand STANDBY zurückgesetzt, indem der Finger zurückgezogen wird. Die verschiedenen Zustände werden mit Hilfe einer für alle Interaktionsobjekte aus der Entwicklungsumgebung einheitlichen Farbkodierung angezeigt (dunkelblau für LOCKED, hellblau für STANDBY, weiß für ACTIVE). Der Übergang zwischen den Zuständen LOCKED und STANDBY wird prozeßbezogen über den Dialogmanager gesteuert.

Die Steuerung der direkt-manipulativen Benutzerschnittstelle erfolgt also durch die wiederholte Abfolge:

Event - Rückmeldung - Funktion

Diese Abfolge wird nun zur Definition und Modellierung eines Dialogs an einer Benutzerschnittstelle erweitert, indem nach Auslösen der mit dem Ereignis verbundenen Funktion ein weiterer Schritt ausgeführt wird. In diesem Schritt wird definiert, welchen Zustand die anderen Schnittstellenobjekte nach einer Aktivierung oder Deaktivierung eines Interaktionsobjekts annehmen. Die Abarbeitung eines Ereignisses sieht demnach wie folgt aus:

Event - Rückmeldung - Funktion - Einstellen neuer Zustände

Die Definition neuer Zustände erfolgt dabei über die Definition von Verbindungen zwischen Interaktionsobjekten, die von einem Dialogmanager bearbeitet werden.

4. Dialogmanager

Eine Verbindung zwischen zwei Interaktionsobjekten obj1 und obj2, die den Folgezustand des Objektes obj2 bestimmt, wird wie folgt definiert:

connect obj1 with obj2 on {activating | deactivating}:
if <condition> then switch on [else switch off].

Dabei beschreibt <condition> einen Zustand, in dem sich bestimmte Interaktionsobjekte befinden müssen, damit das verbundene Objekt obj2 in den Zustand STANDBY (switch on) geschaltet wird (z.B.: "KEY1 in LOCKED or KEY2 in STANDBY"). Durch diese Bedingung kann die Ausführung von Verbindungen zwischen Interaktionsobjekten von bestimmten Situationen, in der sich eine Benutzerschnittstelle befindet, abhängig gemacht werden. Für den Fall, daß der <condition>-Teil konstant TRUE oder FALSE ist, werden folgende abkürzende Schreibweisen eingeführt:

connect obj1 with obj2 on {activating | deactivating}: switch on

für den Fall, daß <condition> = TRUE und:

connect obj1 with obj2 on {activating | deactivating}: switch off,

falls <condition> = FALSE.

Eine solche Verbindung kann grafisch ausgedrückt werden, wie in Abbildung 4 an einem Beispiel

gezeigt wird. Sei zu Beginn Taste KEY1 im Zustand STANDBY und Taste KEY2 im Zustand LOCKED. Wenn KEY1 durch ein dafür geeignetes Ereignis aktiviert wird, dann prüft der Dialogmanager nach, welche Verbindungen zwischen KEY1 und anderen Tasten (inklusive KEY1 selbst) definiert sind. In diesem Fall wird aufgrund der Verbindung "link A" KEY2 in den Zustand STANDBY geschaltet und KEY1 selbst aufgrund der Verbindung "link B" in den Zustand LOCKED.

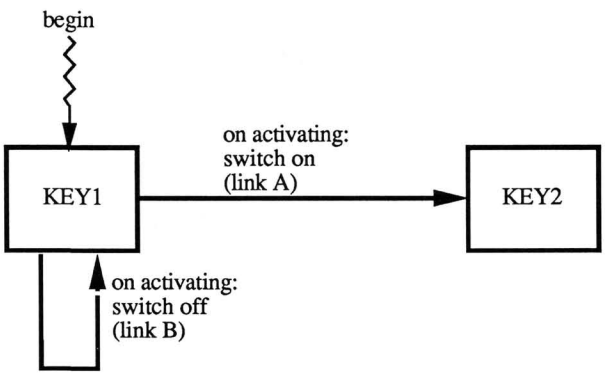

Abb. 4: Beispiel für eine Dialog-Verbindung zwischen zwei Tasten

Das Dialogmanagement ist so angelegt, daß Verbindungen nur zwischen Objekten definiert werden können, die zu einer (komplexen) Teilschnittstelle gehören, d.h. dem gleichen Konstruktor angehören (z.B. RATE, SLIDER und ENTER im Konstruktor RATE CONTROL in Abbildung 2). Auf diese Weise wird eine hierarchische Strukturierung des Dialogs in elementare Dialoge, Teildialoge, usw. analog zur hierarchischen Struktur der Schnittstelle erreicht.

Am Beispiel der Definition der Interaktionssequenzen zur Einstellung der Rate an der Kontrolleinheit der Infusionspumpe soll dies verdeutlicht werden (Abbildung 5). Folgende Anforderungen werden an die zu definierenden Interaktionssequenzen gestellt:

- Die Rate wird eingestellt, indem die RATE-Taste aktiviert, dann der Wert auf dem Werteschieber SLIDER eingestellt und anschließend der eingestellte Wert mit Hilfe der ENTER-Taste quittiert wird.
- Der Benutzer wird bei dieser Sequenz mit Hilfe einer Farbkodierung geführt.
- Durch Deaktivieren der RATE-Taste soll die Interaktionssequenz jederzeit abgebrochen werden können (Undo-Funktion).

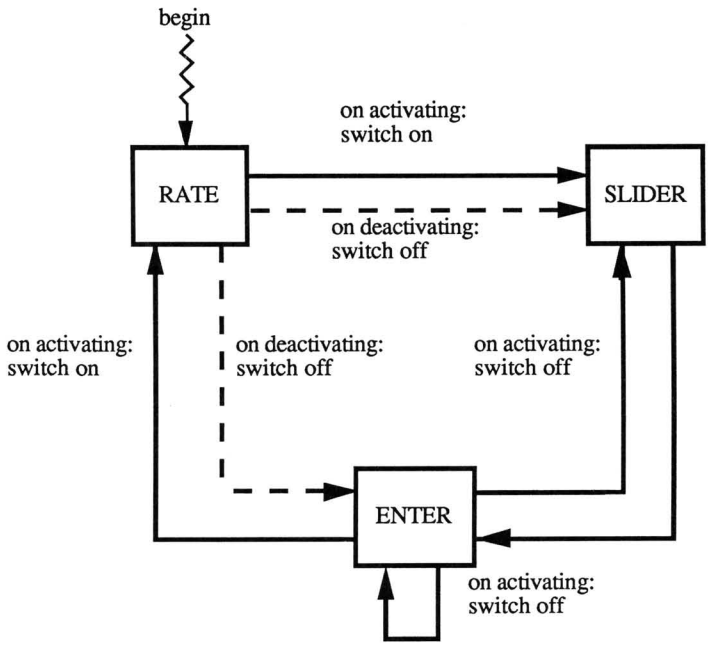

Abb. 5: Definition der Interaktionssequenzen für die Infusionspumpenkontrolleinheit

5. Erfahrungen und Diskussion

In der Einleitung wurden folgende Anforderungen an einen Dialogmanager formuliert:

- einfache Definition eines komplexen Dialogmanagements,
- (syntaktische) Konsistenz der resultierenden Interaktion,
- Möglichkeit zur Entdeckung von Fehlern im Dialogmanagement.

Im folgenden wird gezeigt, wie die Anforderungen durch den vorgestellten Ansatz für einen Dialogmanager umgesetzt werden, wo die Grenzen liegen und welche Erfahrungen damit gesammelt werden konnten.

Die Definition eines Dialogs durch Verbindungen zwischen einzelnen Interaktionsobjekten ermöglicht im Gegensatz zu der globalen Definition von Zustandsübergängen einer ganzen Schnittstelle (/Farooq & Dominick, 1988/, /Jacob, 1986/) eine lokale Sichtweise auf den Dialog. Eine weitere Strukturierung erfolgt durch die hierarchische Zergliederung der Dialogdefinition analog zur Hierarchie der Interaktionsobjekte der Benutzerschnittstelle. Daraus ergibt sich eine an semantischen Einheiten orientierte Sichtweise auf den Dialog (z.B. Einstellen der Rate - Kontrollieren der Infusionspumpe - Kontrolle aller für einen Patienten eingesetzten Geräte - Verwalten aller Patientendaten).

Die geforderte (syntaktische) Konsistenz der Benutzerschnittstelle wird erreicht durch Steuerung der Interaktionsobjekte mit Automaten und die einheitliche Darstellung der Zustände der Automaten mit Hilfe einer Farbkodierung. Die Verkapselung der implementierten Automaten in Objekten einer objektorientierten Umgebung bedeutet weiterhin, daß die angezeigten Zustände auch mit den tatsächlich möglichen Interaktionen übereinstimmen, da eine "falsche" Zuordnung der Zustände zu ihrer Visualisierung durch den Entwickler der Benutzerschnittstelle ausgeschlossen ist.

Die Entdeckung von Fehlern im Dialogmanagement wird durch die Ableitung eines "Dialoggraphen" ermöglicht. Der Dialoggraph eröffnet prinzipiell die Möglichkeit, ergonomische Anforderungen an eine Benutzerschnittstelle graphentheoretisch auszudrücken und mit Hilfe geeigneter Algorithmen zu analysieren. Mit Hilfe des folgenden Verfahrens läßt sich aus den Definitionen der Verbindungen zwischen den Interaktionsobjekten ein (gerichteter) Dialoggraph ableiten, der alle möglichen Interaktionssequenzen an der entwickelten Schnittstelle repräsentiert (multimodale Interaktion kann durch einen getypten Graphen ausgedrückt werden). Dazu wird der Zustand der Schnittstelle als Tupel der Zustände aller Interaktionsobjekte definiert. Durch systematisches Durchspielen der in diesem Zustand möglichen Interaktionen werden alle möglichen Folgezustände für diese Schnittstelle erzeugt. Die rekursive Fortsetzung dieses Verfahrens auf die so erhaltenen neuen Zustände erzeugt schließlich einen Graphen, der als Dialoggraph bezeichnet werden kann. Abbildung 6 zeigt den Dialoggraphen für die Interaktionsobjekte RATE, SLIDER und ENTER der Infusionspumpenkontrolleinheit. (Zu Beginn befinden sich RATE im Zustand STANDBY und SLIDER und ENTER im Zustand LOCKED. Daher ist zunächst die Aktivierung der RATE-Taste die einzig mögliche Interaktion.)

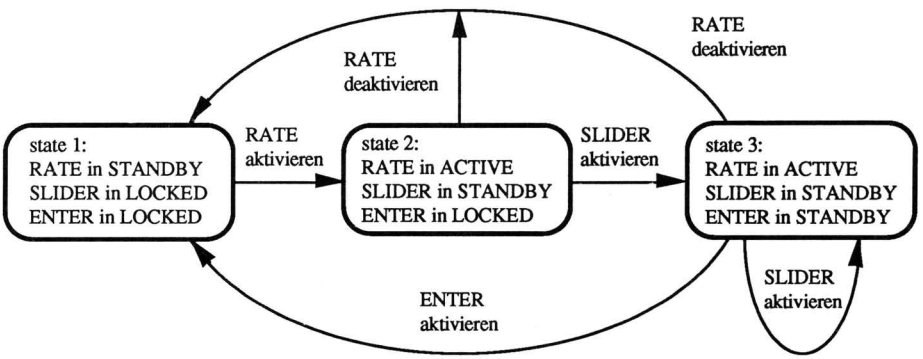

Abb. 6: Dialoggraphen für die Interaktionsobjekte RATE, SLIDER und ENTER

An diesem Graphen drücken sich Dialog-Sackgassen durch Blätter aus, von denen aus keine neuen Zustände mehr erreicht werden können; nicht-umkehrbare Interaktionssequenzen bilden sich ab auf Knoten, die nur einmal erreichbar sind, usw.

Der hier vorgestellte Dialogmanager wurde bisher dazu benutzt, die Interaktionssequenzen von drei sehr unterschiedlichen, komplexen Mensch-Maschine-Schnittstellen zu gestalten: eine Benutzerschnittstelle für ein entscheidungsunterstützendes System (/Schecke et al., 1988/), ein Werkzeug zur interaktiven Farbgestaltung (/Langen et al., 1989/) sowie Studien für ein intensivmedizinisches Informationssystem. Die als Beispiel verwendete Kontrolleinheit für eine Infusionspumpe ist ein Ausschnitt aus einer dieser Benutzerschnittstellen.

Der praktische Einsatz hat gezeigt, daß die besondere, lokale Sichtweise auf den Dialog (Definition von Verbindungen zwischen einzelnen Interaktionsobjekten im Gegensatz zu globalen Zustandsbeschreibungen einer Schnittstelle und ihre Zustandsübergänge) in Verbindung mit der hierarchischen Strukturierung des Dialogs sowie die automatisierte Benutzerführung die Gestaltung der Dialogsequenzen vereinfacht, die Übersichtlichkeit der Dialogdefinition erhöht und eine Fehlersuche erleichtert.

Ein Schwachpunkt des vorgeschlagenen Ansatzes zur Definition von Dialogen stellt die "Kontextfreiheit" der definierten Sequenzen dar, d.h. der Dialog ist nur abhängig vom Zustand der Benutzerschnittstelle und berücksichtigt keine Zustände innerhalb der durch die benutzerinitiierten Ereignisse ausgelösten Funktionen (der eigentlichen Applikation). Auf der anderen Seite läßt sich nach unseren Erfahrungen ein gewisser Anteil an kontextabhängigen (eigentlich applikationsabhängigen) Dialogsequenzen nicht vermeiden. Dies bedeutet, daß einige Verbindungen zwischen Interaktionsobjekten nur unter bestimmten (applikationsabhängigen) Umständen gültig sind, was in dem bisher definierten Modell des Dialogs nicht berücksichtigt wurde. Eine Verallgemeinerung der <condition>-Anweisung bei der Definition der Verbindungen auf beliebige Bool´sche Funktionen kann in diesem Fall Abhilfe schaffen. Dann kann allerdings der Dialoggraph nicht mehr applikationsunabhängig erstellen werden und ist daher nur für bestimmte Zeitpunkte gültig.

Ein Problem bei der Analyse von Dialoggraphen stellt die Definition weiterer geeigneter ergonomischer Kriterien und ihre Abbildung auf graphentheoretische Begriffe dar. Welche Kriterien hierfür wirklich relevant sind, hängt von dem betrachteten Gestaltungsproblem ab und kann nach unserer Erfahrung bisher nur heuristisch ermittelt werden. Für die Erstellung und Analyse des Dialoggraphen gibt es weiterhin Komplexitätsgrenzen, die recht schnell erreicht werden. Daher lassen sich mit diesem Verfahren bisher nur Ausschnitte einer Benutzerschnittstelle betrachten und analysieren. Um den in einer praktischen Anwendung nötigen Zusammenhang zu größeren Komponenten herzustellen, bedarf es einer noch zu entwickelnden Analysemethodik. Mit Hilfe der hierarchischen Struktur der Benutzerschnittstelle sollen dabei die Ergebnisse der Analysen von Ausschnitten zu einem Gesamtbild verknüpft werden.

6. Ergebnis

Der beschriebene Ansatz eines Dialogmanagers beruht auf einer Erweiterung eines Event-Modells, bei der für jedes benutzerinitiierte Ereignis ein "Dialog-Schritt" ausgeführt wird. In diesem Dialog-Schritt wird mit Hilfe von Verbindungen zwischen den elementaren Interaktionsobjekten

definiert, welchen Zustand die anderen Schnittstellenobjekte nach einer Aktivierung oder Deaktivierung eines Interaktionsobjekts annehmen. Dieser Ansatz ermöglicht die Definition eines komplexen Dialogmanagements, die Einrichtung einer automatisierten Benutzerführung und in gewissen Grenzen die Erstellung von "Dialoggraphen" zur Analyse von Dialogen. In der Praxis haben sich die besondere, lokale Sichtweise auf den Dialog, seine hierarchische Strukturierung und die automatisierte Benutzerführung bewährt. Der graphentheoretische Ansatz zur Analyse von Dialogen muß dagegen noch weiter ausgebaut werden.

Literatur

Bernotat, R., Rau, G. (1980): Ergonomics in Medicine. In: H. Reul, D.N. Ghista, G. Rau (eds.): Perspectives in Biomechanics, New York: Harwood Academic Publishers, pp. 381-398.

Farooq, M.U., Dominick, W.D. (1988): A Survey of Formal Tools and Models for Developing User Interfaces. Int. Journal of Man-Machine Studies 29/5, pp. 479-496.

Green, M. (1984): Report on Dialogue Specification Tools. Computer Graphics Forum, 3 (1984), pp. 305-313.

Jacob, J.K.J. (1986): A Specification Language for Direct-Manipulation User Interfaces. ACM Transactions on Graphics, Vol. 5, No. 4, pp. 283-317.

Klocke, H., Trispel, S., Rau, G., Hatzky, U., Daub, D. (1986): An Anesthesia Information System for Monitoring and Record Keeping during Surgical Anesthesia. Journal of Clinical Monitoring 2(4), pp. 246-261.

Klocke, H., Rau, G., Schecke, Th. (1987): Direkte Manipulation durch Berühreingabe bei einem Anästhesie-Informations- und Entscheidungsunterstützungssystem. W. Schönpflug, M. Wittstock (Hrsg.): Software-Ergonomie '87, Bericht des German Chapter of the ACM, Teubner: Stuttgart, 146 - 155.

Langen, M., Thull, B., Schecke, Th., Rau, G., Kalff, G. (1989): Prototyping methods and tools for the human-computer interface design of a knowledge-based system. In: G. Salvendy, M.J. Smith (eds.): Designing and Using Human-Computer Interfaces and Knowledge Based Systems, Amsterdam: Elsevier Science Publishers.

Rau, G., Trispel, S. (1982): Ergonomic Design Aspects in Interaction Between Man and Technical Systems in Medicine. Medical Progress through Technology 9, pp. 153-159.

Schecke, Th., Langen, M., Rau, G., Käsmacher, H., Kalff, G. (1988): Knowledge-Based Decision Support for Monitoring in Anesthesia: Problems, Design and User Interaction. In: O. Rienhoff, U. Piccolo (eds.): Expert Systems and Decision Support in Medicine, Springer, 256-263.

Shneiderman, B. (1982): The future of interactive systems and the emergence of direct manipulation. Behavior and Information in Technology 1(3), pp. 237-256.

Wilson, J., Rosenberg, D. (1988): Rapid Prototyping for User Interface Design. In: M. Helander (ed.): Handbook of Human-Computer Interaction, Amsterdam: North-Holland, pp. 859-875.

Dipl.-Inform. B. Thull
Helmholtz-Institut für Biomedizinische Technik
an der RWTH Aachen
Pauwelsstraße 30
D-5100 Aachen