Dialognetze zur Beschreibung von Dialogabläufen in graphisch-interaktiven Systemen

Christian Janssen
Fraunhofer-Institut für Arbeitswirtschaft und Organisation, Stuttgart

Zusammenfassung

Dialognetze wurden zur Beschreibung von Dialogabläufen in graphisch-interaktiven Systemen entwickelt. Sie können zur Spezifikation und Dokumentation der Dialogabläufe sowohl bei herkömmlicher Erstellung der Benutzungsschnittstelle als auch im Rahmen der automatischen Generierung aus höheren Beschreibungsformen eingesetzt werden. Im Gegensatz zu bisher in der Software-Entwicklung verwendeten Techniken sind Dialognetze für die Beschreibung paralleler Teildialoge, wie sie in graphischen Benutzungsschnittstellen auftreten, geeignet. Die Verwendung von Dialognetzen entspricht der Forderung von Software-Entwicklern nach einer Integration der Benutzungsschnittstellenentwicklung in software-technische Vorgehensweisen, die trotz prototyp-orientierten Vorgehens nicht vergessen werden sollte.

1 Einleitung

Eine bessere Integration benutzer- und benutzungsschnittstellenorientierter Methoden und Vorgehensweisen in die Software-Entwicklung wird von verschiedenen Seiten gefordert [5]. Unklar ist jedoch in vielen Einzelheiten, welche Methoden und Vorgehensweisen dem Praktiker in der betrieblichen Software-Entwicklung empfohlen werden können. Als einen Teilbereich greift diese Arbeit die Spezifikation der Dialogabläufe in graphischen Benutzungsschnittstellen heraus.

Die Notwendigkeit der Abkehr von den alten Phasenmodellen zugunsten einer prototyp-orientierten und iterativen Vorgehensweise ist in der Wissenschaft kaum umstritten und setzt sich auch in der Praxis mehr und mehr durch. Einige Beispiele geben Kieback et al. [8]. Zu einem software-technisch sauberen Vorgehen gehört jedoch auch die Dokumentation der Prototypen, damit die Spezifikation eines Systems nicht in – schlimmstenfalls unstrukturiert programmierter – Prototyp-Software verborgen bleibt. Die bisher gewohnten Beschreibungsmittel basieren in der Regel auf Zustandsübergangsdiagrammen, z.B. Interaktionsdiagramme [4], und sind für graphische Systeme mit parallelen Teildialogen nicht geeignet. Es werden also neue Techniken benötigt.

Die in dieser Arbeit beschriebenen Dialognetze können unabhängig von der Art der Erstellung der statischen Benutzungsoberfläche (Oberflächenobjekte mit statischen

Attributen und Layout) für die Spezifikation der Dynamik in graphischen Systemen eingesetzt werden. So ist es zum einen möglich, daß die statische Benutzungsoberfläche mit graphischen Oberflächeneditoren erstellt wird, die im Rahmen heutiger Entwicklungswerkzeuge üblich sind. Die groben Dialogabläufe können dann ebenfalls graphisch mit Dialognetzen beschrieben werden, was im Vergleich zu den textuellen Dialogbeschreibungssprachen heutiger User Interface Management Systeme (UIMS) [12] für die frühen Phasen des Entwurfes angemessener ist.

Zum anderen gibt es in neuerer Zeit Ansätze, die statische Benutzungsoberfläche aus Datenmodellen oder anderen höheren Beschreibungsformen weitgehend automatisch zu generieren [1, 15]. Hierbei können ebenfalls Dialognetze als höhere Beschreibungstechnik für die Dynamik verwendet werden, analog zu dem höheren Abstraktionsgrad für die statische Benutzungsoberfläche, den der Ansatz der automatischen Generierung gegenüber heutigen UIMS bietet.

Im folgenden zweiten Abschnitt werden aus der Literatur bekannte Ansätze für die graphische Dialogablaufbeschreibung besprochen und deren Grenzen aufgezeigt. Der dritte Abschnitt führt die Beschreibungselemente von Dialognetzen ein und gibt Einsatzmöglichkeiten und Einsatzerfahrungen wieder. Im Ausblick werden Entwicklungsperspektiven in bezug auf Dialognetze selbst sowie die Werkzeugunterstützung für deren Verwendung in der Softwareentwicklung aufgezeigt.

2 Beschreibungstechniken für Dialogabläufe

In der Literatur sind zahlreiche Beschreibungstechniken für Dialogabläufe bekannt, die auf Zustandsübergangsdiagrammen beruhen und zum Teil auch in der betrieblichen Software-Entwicklung verwendet werden [4, 6, 14, 16]. Mit Zustandsübergangsdiagrammen sind aber in der Grundform nur sequentielle Abläufe beschreibbar. Jacob [6] weicht diese Beschränkung auf, indem das Gesamtsystem durch mehrere Diagramme beschrieben wird, die zueinander quasi-parallel ablaufen können. Hierbei läßt sich dann aber der Zusammenhang zwischen den Diagrammen nicht graphisch veranschaulichen. Yunten und Hartson [16] führen eine Verzweigung des Steuerflusses ein, wodurch zwar parallele Abläufe beschreibbar werden, jedoch keine klare Semantik bezüglich der Parallelität und deren Einschränkungen entsteht.

Als geeigneter Formalismus zur Beschreibung von Dialogen in graphisch-interaktiven Systemen erscheinen daher Petri-Netze, die bereits im Grundansatz klar definierte Konzepte zur Beschreibung von Nebenläufigkeit (hier etwas unscharf auch Parallelität genannt) mitbringen [13]. Petri-Netze wurden bereits in verschiedenen Arbeiten zur Beschreibung von Dialogabläufen verwendet. RFA-Netze [9] können

unter anderem Dialogabläufe beschreiben, wurden aber bisher nicht für die Beschreibung von graphischen Dialogen verwendet.

Ereignisgraphen von Roudaud et al. [10] und Petri-Netz-Objekte nach Bastide und Palanque [2] sind speziell für die Spezifikation von graphischen Dialogen entwickelt worden. Petri-Netz-Objekte sind als höhere Netze mächtiger als Ereignisgraphen, andererseits ist aber der Detaillierungsgrad relativ hoch, so daß sie für die Verwendung in den ersten Entwurfs-Phasen weniger geeignet sind. Für die in dieser Arbeit beschriebenen Dialognetze wurden die einfacheren Ereignisgraphen als Grundlage genommen. Es wurden aber wesentliche Erweiterungen in bezug auf modale Teildialoge, Vereinfachung der Netzstruktur und hierarchische Gliederung durchgeführt.

Die Verwendung von Daten- und Steuerflußdiagrammen [11] wird hier nicht weiter verfolgt, da solche Diagramme nur Datenfluß- und Aufrufbeziehungen, nicht jedoch genaue Abläufe modellieren können.

3 Dialognetze

In der Praxis hängt der Erfolg jeder graphischen Modellierungstechnik von der Wahl einer angemessenen Abstraktionsebene für den Beschreibungsgegenstand ab. Wird diese zu detailliert festgelegt, wird der Aufwand für die Erstellung der Beschreibung zu groß und die Lesbarkeit wird erschwert. Aus diesem Grund werden im folgenden zwei verschiedene Ebenen für die Dialogabläufe in einer graphischen Benutzungsschnittstelle unterschieden.

- *Dialogabläufe auf Fensterebene* (grobe Dialogabläufe) beschreiben die Abfolge von Fenstern aufgrund von Benutzereingaben, wobei systemintern Anwendungsfunktionen aufgerufen werden können. Werden größere Teile innerhalb von Fenstern dynamisch geöffnet und geschlossen, so kann dies ebenfalls auf der Ebene der groben Dialogabläufe beschrieben werden.
- *Dialogabläufe auf Objektebene* (feine Dialogabläufe) beschreiben die Zustandswechsel der Oberflächenobjekte, die sich innerhalb von Fenstern befinden, z.B. die Änderung der Selektierbarkeit von Menü-Einträgen und die Selektion von Ikonen.

Dialognetze sind prinzipiell für beide Abstraktionsebenen geeignet, sollten aber schwerpunktmäßig für die Dialogabläufe auf Fensterebene verwendet werden, da die vollständige Beschreibung der Dialogabläufe auf Objektebene graphisch zu aufwendig ist. Dialognetze beschreiben dabei die Nebenläufigkeit von Dialogen in Fenstersystemen, wobei allerdings von der Sequentialisierung der Eingaben durch das Fenstersystem, das ja nicht der Modellierungsgegenstand ist, abstrahiert wird.

3.1 Beschreibungselemente von Dialognetzen

Grundelemente

Wie alle Petri-Netze bestehen Dialognetze aus Stellen (in der Graphik Kreise oder Ovale), Transitionen (Rechtecke) und einer Flußrelation (Pfeile) zwischen Stellen und Transitionen oder umgekehrt. Zur Beschreibung von groben Dialogabläufen wird die Sichtbarkeit von Fenstern (bzw. Fensterteilen) mit Hilfe einer dem Fenster zugeordneten Stelle modelliert. Ein Fenster ist genau dann sichtbar, wenn die zugehörige Stelle markiert ist. In der heutigen Version sind Dialognetze Bedingungs-/Ereignisnetze [13], so daß ein Fenster nur einmal zur Zeit geöffnet sein kann, es also gleichzeitig nicht mehr als eine Inkarnation eines Fensters geben kann.

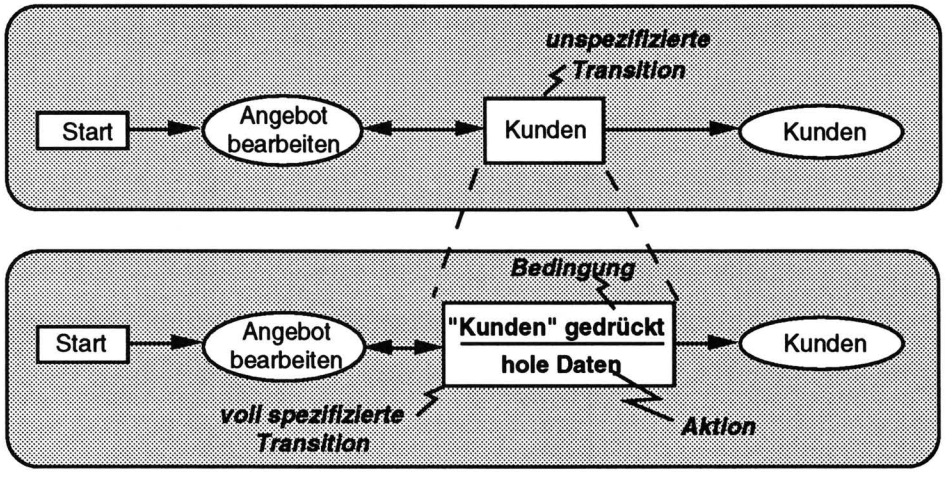

Abb. 1: Dialognetz mit unspezifizierter und voll spezifizierter Transition

Dialogschritte werden in Transitionen beschrieben. An Transitionen können Bedingungen und Aktionen gebunden werden. Die Bedingung ist in erster Linie ein Eingabeereignis, kann aber auch weitere Nebenbedingungen enthalten. Aktionen beinhalten den Aufruf von Anwendungsfunktionen oder verändern Eigenschaften von Oberflächenobjekten. Die *Schaltregel* für Transitionen in Dialognetzen ist wie folgt definiert:

(1) Eine Transition kann nur schalten, wenn alle Eingangs- und Nebenstellen markiert sind und keine Ausgangsstelle markiert ist. Beim Schalten einer Transition werden die Marken in den Eingangsstellen entfernt und die Ausgangsstellen markiert.

(2) Eine Transition kann nur schalten, wenn das in der Bedingung spezifizierte Ereignis eingetreten ist und die spezifizierten Nebenbedingungen wahr sind. Beim Schalten einer Transition werden die spezifizierten Aktionen ausgeführt.

(3) Zu Beginn eines Dialoges schaltet eine besonders beschriftete Start-Transition.

In einer ersten Beschreibung eines Dialogsystems auf höherer Abstraktionsebene brauchen Bedingungen und Aktionen nicht spezifiziert zu sein. Es genügt die sinnvolle Benennung der Transitionen. Die Schaltregel reduziert sich dann entsprechend auf die Regel (1). Ferner kann trotz Vorhandensein einer Bedingung die Aktion weggelassen werden, was die Regel (2) entsprechend reduziert.

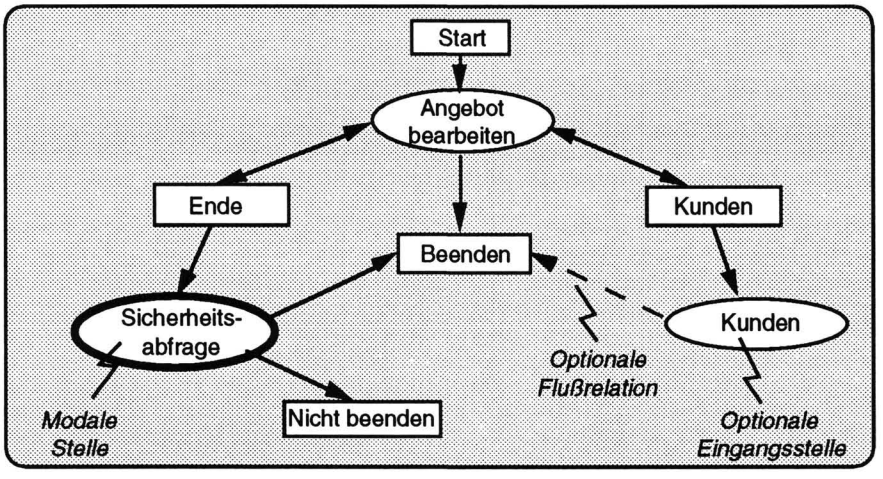

Abb. 2: Modale Stelle und optionale Flußrelation

Abb. 1 zeigt Dialognetze für einen Ausschnitt aus einem Dialog zur Angebotsbearbeitung mit unterschiedlich detaillierten Transitionen. Zu Beginn des Dialoges wird ein Fenster "Angebot bearbeiten" geöffnet, in dem der Benutzer Daten zum Angebot eintragen kann. Durch die Transition "Kunden" kann zu einem Fenster "Kunden" verzweigt werden. Die zugehörige Stelle ist Ausgangsstelle (ausgehender Pfeil) der Transition. Das Fenster "Angebot bearbeiten" bleibt geöffnet, da die zugehörige Stelle als Nebenstelle (eingehender und ausgehender bzw. Doppelpfeil) dargestellt wurde. In dem unteren Netz ist die Transition voll spezifiziert, indem eine Bedingung (Drücken einer Schaltfläche "Kunden") und eine Aktion (holen von Daten für einen evtl. schon eingetragenen Kunden) angegeben wurde.

Modale Stellen

In dem Beispiel in der Abb. 1 stehen alle geöffneten Fenster parallel zur Verfügung und können beliebig vom Benutzer bearbeitet werden. Bei modalen Dialogfenstern

dagegen, wie sie z.B. bei Sicherheitsabfragen verwendet werden, muß auch eine Einschränkung der Parallelität spezifiziert werden können. Zur Modellierung von modalen Dialogfenstern werden *modale Stellen* eingeführt (graphisch durch fetten Rand gekennzeichnet). Ist eine modale Stelle markiert, können systemweit nur noch Transitionen schalten, die diese als Eingangsstelle haben. Solche Transitionen werden auch *modale Transitionen* genannt. Dabei wird festgelegt, daß zur Zeit nur eine modale Stelle im Netz markiert sein darf, d.h. jede Transition höchstens eine modale Stelle als Ausgangs- oder Nebenstelle haben darf. Es gilt dann die folgende Schaltregel:

(4) Eine nichtmodale Transition kann nur schalten, wenn keine modale Stelle im Netz markiert ist.

Optionale Flußrelationen

In fensterorientierten Dialogen kann es vorkommen, daß eine Transition mehrere Fenster schließen soll, falls diese offen sind, aber auch schalten soll, falls einige der Fenster geschlossen sind. Analog kann es Transitionen geben, die Fenster öffnen sollen, aber nur, falls sie noch nicht offen sind. Zur Vereinfachung des Netzes bei diesen Situationen werden *optionale Flußrelationen* (gestrichelte Pfeile) eingeführt. Stellen, die mit Transitionen durch optionale Flußrelationen verbunden sind, heißen entsprechend optionale Eingangs-, Ausgangs- bzw. Nebenstellen. (Optionale Nebenstellen sind nur sinnvoll in Verbindung mit Subdialogen in Transitionen, s.u.) Die Schaltregel (1) wird wie folgt ergänzt:

(1a) Optionale Eingangs-, Neben- und Ausgangsstellen beeinflussen nicht die Schaltbedingung. Beim Schalten einer Transition werden aber evtl. vorhandenene Marken in optionalen Eingangsstellen entfernt und die optionalen Ausgangsstellen markiert, falls sie es noch nicht sind.

In dem Beispiel in Abb. 2 führt das Schalten der Transition "Ende" zum Öffnen der Sicherheitsabfrage. Von diesem Augenblick an sind nur noch die Transitionen "Beenden" und "Nicht beenden" möglich. Beim Schalten der Transition "Beenden" wird eine eventuell vorhandene Marke aus der Stelle "Kunden" entfernt (das Kundenfenster geschlossen). Die Marke in der Stelle ist aber nicht Voraussetzung für das Schalten, da die Flußrelation als optional gekennzeichnet ist.

Hierarchische Strukturierung von Dialognetzen

In der Praxis treten in den meisten Dialogen so viele Fenster und Dialogschritte auf, daß der gesamte Ablauf nicht mit einem einzigen Dialognetz beschrieben werden kann. Es ist also eine Unterteilung in mehrere Netze erforderlich. Hierzu dienen in Dialognetzen *komplexe Stellen.* Eine komplexe Stelle ist eine Vergröberung eines Teildialoges mit einem oder mehreren Fenstern und wird graphisch durch doppelte Umrandung dargestellt.

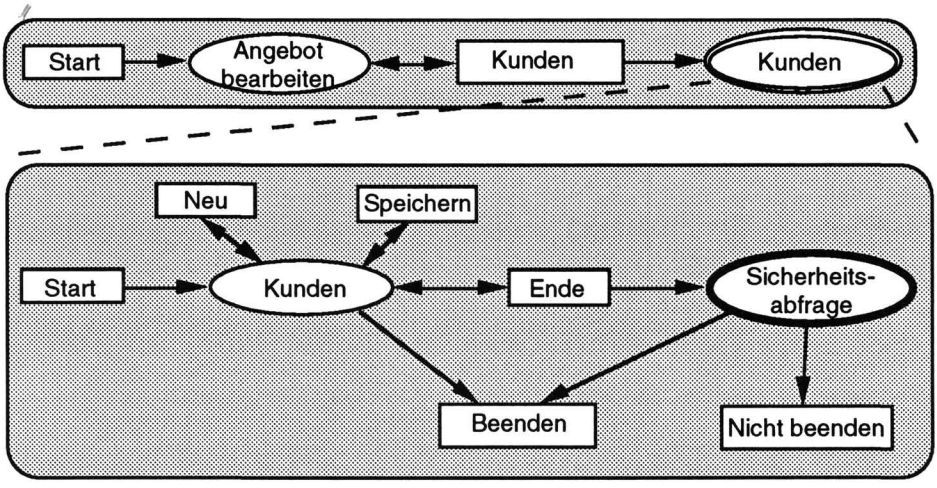

Abb. 3: Vergröberung und Verfeinerung einer Stelle in einem Dialognetz

Die Beziehungen zwischen einer komplexen Stelle und ihrer Verfeinerung sind wie folgt definiert: Wird eine komplexe Stelle markiert, so tritt das Start-Ereignis für das Unternetz ein und somit wird das Unternetz aktiviert. Wird die Marke aus einer komplexen Stelle abgezogen, so führt dies automatisch zum Abzug aller Marken aus dem Unternetz, also zu dessen Deaktivierung. Ebenso wird die Marke in einer komplexen Stelle entfernt, wenn alle Marken aus dem Unternetz entfernt werden. Für die Beendigung eines Unterdialoges in einem Unternetz gibt es also zwei Möglichkeiten: Abzug aller Marken durch eine Transition im Unternetz oder Abzug der Marke aus der zugehörigen komplexen Stelle im Obernetz. Die Schaltregel (4) wird wie folgt auf hierarchisch gegliederte Netze erweitert:

(4) Eine nichtmodale Transition kann nur schalten, wenn keine modale Stelle im Gesamtsystem markiert ist.

In dem Beispiel in Abb. 3 wurde der gesamte Kundendialog zu einer komplexen Stelle vergröbert und in einem separaten Unternetz beschrieben. Die für das Kunden-Fenster lokalen Dialogschritte "Speichern" usw. wurden ebenso wie der Dialogablauf beim Beenden des Kundendialoges in das Unternetz verlagert.

Weitere Elemente in Dialognetzen ermöglichen es, je nach Aufrufkontext mit unterschiedlichen Start-Aktionen in Unternetze einzusteigen und beim Beenden von Unternetzen situationsabhängig im Hauptdialog unterschiedliche Aktionen auszulösen. Ferner ist die Vergröberung von Dialogen (insbesondere für Meldungen und Hilfe) in Transitionen möglich. Aus Platzgründen werden diese Konstrukte hier nicht beschrieben.

3.2 Einsatz von Dialognetzen in der Software-Entwicklung

In betrieblichen Software-Projekten wird unserer Erfahrung nach eine formalisierte und dennoch leicht verständliche Beschreibungstechnik für die Dialogabläufe in graphisch-interaktiven Systemen von Entwicklern gewünscht. Daher wurden in Praxisbeispielen Dialognetze für die Beschreibung der groben Dialogabläufe verwendet. Die Erfahrung zeigte, daß die Konzepte schnell von Software-Entwicklern erlernt wurden und der Nutzen von Dialognetzen, z.B. zum Festhalten des Ergebnisses von Projektsitzungen im groben Dialogentwurf, von ihnen bestätigt wurde. Die frühe Visualisierung der Dialogabläufe konnte Inkonsistenzen und Unvollständigkeiten aufzeigen. Ferner war es in einigen Fällen möglich, zu erwartende Probleme hinsichtlich der Transparanz von Dialogen für den Benutzer aufzudecken.

Gewisse Anfangsschwierigkeiten hatten Entwickler in bezug auf die nicht-sequentiellen Abläufe in Dialognetzen. So muß z.B. beim Abbrechen der Sicherheitsabfrage in Abb. 3 das Kundenfenster nicht wieder aktiviert werden, da es vorher nicht geschlossen wurde.

Dialognetze sind hauptsächlich für die Dialogabläufe in datenbankgestützten Anwendungen entworfen worden. Es gibt andere Klassen von Anwendungen, z.B. Standard-Anwendungen der Dokument-Verarbeitung, für die sie als Beschreibungstechnik weniger nützlich sind, da eine Repräsentation des Dokumentes im Zentrum des Dialoges steht. Die interessanten Dialogabläufe liegen hier auf der Objektebene und der Wechsel zu anderen Fenstern ist von sekundärer Bedeutung.

Im AuT-Projekt "Unterstützungswerkzeuge zur benutzergerechten Gestaltung der Mensch-Computer-Schnittstelle" [15] werden Dialognetze eingesetzt, um die Dynamik von Oberflächenobjekten zur spezifizieren, welche im Rahmen eines regelbasierten Ansatzes automatisch generiert werden. Hierfür wurde eine Formalisierung der Bedingungen und Aktionen in den Transitionen durchgeführt, so daß aus der Netzbeschreibung direkt Regeln für ein User Interface Management System generiert werden können.

4 Ausblick

Zwei Erweiterungen von Dialognetzen gegenüber der hier dargestellten Form sind denkbar. Zum einen kann durch zusätzliche Arten von Stellen und Flußrelationen neben dem Steuerfluß auch der Datenfluß zwischen Fenstern und Datenobjekten der Anwendung dargestellt werden, ähnlich wie in [13]. Allerdings würde hierdurch die graphische Übersichtlichkeit leiden und es ist fraglich, ob der Datenfluß aus der Sicht des Dialogablaufes wirklich benötigt wird.

Zum anderen werden Erweiterungen erforderlich, falls man von der Beschränkung abgehen will, daß nur eine Inkarnation eines Fensters zur Zeit gezeigt wird. Hier ist der Übergang zu höheren Petri-Netzen möglich, wobei allerdings die Einfachheit des Formalismus leiden würde.

Als Werkzeugunterstützung für Dialognetze steht gegenwärtig ein graphischer Editor mit einer Generierungsmöglichkeit für ausführbare Dialoge zur Verfügung. Die Mechanismen zur Codegenerierung sind noch zu erweitern, so daß auch manuelle Änderungen des Codes bei folgenden Generierungen nachgeführt werden können. Eine weitere Neuerung wäre, neben dem "top-down"-Ansatz der Generierung von Prototypen aus Dialognetzen auch den "bottom-up"-Ansatz der Synthese von Dialognetzen aus Dialogbeschreibungen, die von UIMS benutzt werden, zu unterstützen. Darüberhinaus wird die Integration mit Werkzeugen für andere Beschreibungstechniken im Rahmen einer computerunterstützten Software-Entwicklung angestrebt. Im AuT-Projekt TASK [3] wird eine Gesamtmethode entwickelt, die neben dem Dialogmodell ein Aufgabenmodell, ein konzeptionelles Objekt- und Funktionsmodell sowie ein statisches Oberflächenmodell (Sichten) vorsieht.

Insgesamt wird davon ausgegangen, daß Dialognetze im Rahmen einer aufgaben- und benutzerorientierten Vorgehensweise schwerpunktmäßig ein Darstellungsmittel für Software-Entwickler sind, da netzbasierte Beschreibungen nicht ohne weiteres von Benutzern interpretiert werden können [10]. Zur Förderung der Benutzer-Entwickler-Kommunikation sind daher lauffähige Prototypen, die u.a. aus Netzbeschreibungen generiert werden, weitaus besser geeignet. Dialognetze haben aber ihren Nutzen für die Spezifikation und Dokumentation von graphisch-interaktiven Anwendungen. Sie führen zu besser durchdachten Dialogen und sind ein Hilfsmittel für eine bessere Integration der Entwicklung der Benutzungsschnittstelle mit software-technischen Methoden.

Diese Arbeit wurde im Rahmen der Projekte "TASK - Technik der aufgaben- und benutzerangemessenen Software-Konstruktion" und "Unterstützungswerkzeuge zur benutzergerechten Gestaltung der Mensch-Computer-Schnittstelle" vom Projektträger Arbeit und Technik des BMFT gefördert (Förderkennzeichen 01 HK 849 A2 bzw. 01 HK 409 0).

Ich danke den Mitgliedern des Programmkomitees, deren konstruktive Anmerkungen zur Verbesserung dieses Berichts beigetragen haben.

5 Literatur

[1] de Baar, D.J.M.J., Foley, J., Mullet, K.E. (1992). Coupling Application Design and User Interface Design. In: CHI'92 Conference on Computer and Human Interaction, 259-266.

[2] Bastide, R. und Palanque, P. (1991). Petri Net Objects for the Design, Validation and Prototyping of User-Driven Interfaces. In: Diaper et al. (Eds.) Human-Computer Interaction - INTERACT'90. Amsterdam: North-Holland, 625-631.

[3] Beck, A., Ziegler, J. (1991). Methoden und Werkzeuge für die frühen Phasen der Software-Entwicklung. In: Ackermann, D., Ulich, E. (Hrsg.). Software-Ergonomie '91. Stuttgart: Teubner, 76-85.

[4] Denert, E. (1977). Specification and Design of Dialog Systems with State Transition Diagrams. In: Morlet, E., Ribbens, D. (Eds.). Proc. of the International Computing Symposium. Amsterdam: North-Holland, 417-427.

[5] Eberleh, E., Arend, U., Kasten, C., Strothotte, T., Ziegler, J. (1991). Integration software-ergonomischer Forschungsergebnisse in die betriebliche Software-Entwicklung. Diskussionsunterlagen. In: Ackermann, D., Ulich, E. (Hrsg.). Software-Ergonomie '91. Stuttgart: Teubner, 141-151.

[6] Jacob, R.J.K. (1986). A Specification Language for Direct-Manipulation User Interfaces. ACM Transactions on Graphics 5 (4), 283-317.

[7] Keil-Slawik, R. (1989). Systemgestaltung mit Aufgabennetzen. In: Maaß, S., Oberquelle, H. (1989). Software-Ergonomie'89. Stuttgart: Teubner, 123-133.

[8] Kieback, A., Lichter, H., Schneider-Hufschmidt, M., Züllighoven, H. (1992). Prototyping in industriellen Software-Projekten. Informatik Spektrum 15 (2), April 1992, 65-77.

[9] Oberquelle, H. (1987). Sprachkonzepte für benutzergerechte Systeme. Berlin, Heidelberg: Springer.

[10] Roudaud, B., Lavigne, V., Lagneau, O., Minor, E. (1990). SCENARIOO: A New Generation UIMS. In: Diaper et al. (Eds.). Human-Computer Interaction - INTERACT'90. Amsterdam: North-Holland, 607-612.

[11] Schend, B. (1991). Verwendbarkeit von SE-Methoden bei der Entwicklung dialogorientierter Systeme. In: Ackermann, D. und Ulich, E. (Hrsg). Software-Ergonomie'91. Stuttgart: Teubner, 86-95.

[12] Trefz, B. Ziegler, J. (1989). DIAMANT - Ein User Interface Management System für graphische Benutzerschnittstellen. In: Maaß, S., Oberquelle, H. (1989). Software-Ergonomie'89. Stuttgart: Teubner, 264-273.

[13] Valk, R., Jessen, E. (1987). Modelle für Rechensysteme. Berlin, Heidelberg: Springer.

[14] Wasserman, A.I. (1985). Extending Transition Diagrams for the Specification of Human-Computer Interaction. IEEE Trans. Software Engineering 11, 8, 699-713.

[15] Weisbecker, A. (1993). Integration von software-ergonomischem Wissen in die System-entwicklung. In diesem Band.

[16] Yunten, T. und Hartson, H.R. (1985). A Supervisory Methodology and Notation (SUPERMAN) for Human-Computer System Development. In: Hartson, H.R. (ed.). Advances in Human-Computer Interaction. Norwood (NJ): Ablex Publishing, 243-281.

Christian Janssen
Fraunhofer-Institut für Arbeitswirtschaft und Organisation (IAO)
Nobelstraße 12
7000 Stuttgart 80.

Tel.: 0711/970-2330
Fax: 0711/970-2300
Email: C_Janssen@iao.fhg.de