ViewNet - Konzeptionelle Gestaltung und Modellierung von Navigationsstrukturen

Jürgen Ziegler

Fraunhofer-Institut für Arbeitswirtschaft und Organisation

Zusammenfassung

Der Beitrag stellt einen Ansatz zur Modellierung von Navigationsstrukturen bei datenbankorientierten Anwendungen mit graphischen Benutzungsschnittstellen sowie hypertext-artigen Strukturen dar. Im Gegensatz zu existierenden Techniken, die sich eher auf die Spezifkation der Dynamik des Dialogablaufs konzentrieren, steht bei ViewNets die Darstellung der Semantik der einzelnen Knoten der Dialogstruktur im Vordergrund. Hierzu werden Sichtentypen eingeführt und durch graphische Symbole veranschaulicht. Bausteinkonzepte und komplexe Sichten bieten die Möglichkeit zur Darstellung auch umfangreicher Navigationsstrukturen in übersichtlicher Form.

1 Einleitung

Eines der Hauptprobleme bei der Entwicklung komplexer Anwendungssysteme mit graphischen Benutzungsschnittstellen liegt heute in zunehmendem Maß in der Auslegung der Dialogstruktur einer Anwendung „im Großen", d. h. der Gesamtarchitektur der Benutzungsschnittstelle. Während für die Verwendung und Gestaltung der Interaktionselemente graphischer Benutzungsschnittstellen Gestaltungsregeln z. B. in Form von Style Guides vorliegen, ist die Frage der Konzeption geeigneter Dialogstrukturen vom Entwickler für die jeweilige Anwendung meist spezifisch zu lösen und wird bestenfalls durch anwendungs- oder firmenspezifische Richtlinien unterstützt. Gerade die Auslegung der Dialogpfade durch die unterschiedlichen Bereiche des Systems ist aber für die Benutzbarkeit von zentraler Bedeutung, insbesondere wenn der Benutzer die grundlegende Handhabung graphischer Oberflächen bereits beherrscht.

Zur besseren Eingrenzung gegenüber den Dialogteilen, die zur Manipulation von Interaktionsobjekten (Listen, Felder, Buttons etc.) dienen, soll hier im folgenden der Begriff *Navigation* für die Bewegung des Benutzers durch die Sichten eines System verwendet werden. Unter einer *Sicht* soll eine Zusammenfassung von Elementen verstanden werden, die ein oder mehrere zugrundeliegende Anwendungsobjekte oder Aufgaben in zusammenhängender Weise z. B. in einem Fenster oder einer Maske für den Benutzer sichtbar macht.

Zur Modellierung von Navigationsabläufen wurde bislang eine Reihe von Techniken entwikkelt, die im wesentlichen eine genaue Beschreibung des Systemverhaltens in Abhängigkeit von Systemzustand und Benutzereingaben zum Ziel haben. Diese Techniken basieren typischerweise auf Zustandsübergangsdiagrammen [1], [2] oder Petrinetzen [3]. Durch die detaillierte Spezifikation von auslösenden Ereignissen, Bedingungen etc. bergen sie jedoch die Gefahr in sich, den Blick auf die Gesamtstruktur für den Entwickler eher zu verstellen als offenzulegen. Im Bereich von Hypermedia-Systemen gibt es zwar verschiedene Ansätze zum strukturellen Ent-

wurf der Navigation [4], [5], diese sind aber für datenbank-orientierte Anwendungen wenig geeignet, bei denen spezielle Abläufe aufgrund der Kardinalitäten von Objektklassen (viele Instanzen gleichen Typs) erforderlich werden.

Die angeführten Methoden sagen auf der anderen Seite wenig über Art und Eigenschaften der bei der Navigation zu durchlaufenden Sichten aus (in der Regel nur den Namen). Sie können daher als reich an Transitionsinformationen, aber arm an Zustandsinformationen bezeichnet werden. Diese Information ist aber für eine konsistente und verständliche Systemgestaltung wichtig, in den frühen Entwurfsphasen meist wichtiger als eine genaue Ablaufspezifikation. Es fehlt bislang eine Methode, die beim Navigationsentwurf beiden Aspekten Rechnung trägt und besonders die Bedeutung der Sichten, aus denen die Navigationsstruktur aufgebaut ist, deutlich macht. In diesem Beitrag wird die Methode **ViewNet** vorgestellt, die diese Anforderungen berücksichtigt und den Entwickler beim konzeptionellen Entwurf von Navigationsstrukturen unterstützt.

2 Prinzipien der Navigationsstrukturierung

Navigationsdialoge können nach unterschiedlichen Prinzipien strukturiert werden. Ausschlaggebend ist dabei der jeweils dominante Aspekt der Aufgabenstellung des Benutzers, der als Ausgangspunkt zur Auslegung des Dialogpfades für die Aufgabenerledigung dient [6]. Grundlegende Aufgabenaspekte sind das zu bearbeitende Objekt, der Zustand des Objektes oder die auszuführende Funktion. Hinzu kommen bei reinen Informationsbeschaffungsaufgaben Art und Beziehung der gewünschten Informationen. Entsprechend lassen sich folgende Navigationsprinzipien unterscheiden:

Funktionsorientierte Navigation: Ausgangspunkt für die Navigation bildet die Auswahl einer bestimmten Funktion (z. B. in einem konventionellen hierarchischen Menüsystem). Die eigentliche Datensicht wird erst nach einem oder mehreren Folgeschritten sichtbar bzw. manipulierbar.

Prozessorientierte Navigation kann als Erweitung des funktionsorientierten Prinzips gesehen werden und bietet Unterstützung für die Ausführung einer zu einem Arbeitsprozess gehörenden Menge von Aufgaben. Das System kann den Status des Prozesses kontrollieren und zustandsabhängig den Zugang zu den benötigten Objekten und Funktionen ermöglichen. Während solche Systeme früher den Benutzer oft zwangen, eine fixe Schrittfolge einzuhalten, erlauben heutige graphische Oberflächen eine flexiblere Gestaltung, z. B. durch Visualisierung der gegenwärtig verfügbaren Aufgaben in einer Task-Leiste.

Objektorientierte Navigation verwendet die Objekte der Anwendung und deren semantische Beziehungen zur Auslegung von Dialogpfaden. Die Funktionen werden erst bei Sichtbarkeit des zu bearbeitenden Objekts verfügbar. Objektorientierte Navigation ist eines der Grundprinzipien graphischer Benutzungsschnittstellen dar und zeichnet sich besonders durch hohe Einheitlichkeit und Flexibilität aus.

Assoziationsorientierte Navigation soll die typische Navigationsform in Hypertext-/Hypermedia-Systemen bezeichnen. Hierbei stellen die Knoten der Navigationsstruktur beliebige einzelne Informationseinheiten dar (im Gegensatz zu festen Objekttypen mit beliebig vielen Instanzen bei objektorientierter Navigation). Die möglichen Übergänge werden durch die Assoziationen zwischen diesen Informationseinheiten bestimmt.

Diese unterschiedlichen Navigationsformen haben bei gegebenen Aufgabenstellungen unterschiedliche Profile hinsichtlich Benutzbarkeitskriterien wie Effizienz, Verständlichkeit oder Flexibilität. In realen Systemen treten daher oft Mischformen auf. Auf der Basis der aufgeführten Navigationsgrundformen sollen nun im folgenden unterschiedliche Typen von Sichten entwickelt werden, die die Knoten der Navigationstruktur darstellen.

3 Sichtentypen

Sichten bilden Anwendungsobjekte, Aufgaben oder allgemeine Informationseinheiten in einer zusammenhängenden Darstellung ab, wobei die Ausgangsobjekte ganz oder teilweise repräsentiert werden können. Mehrfache Sichten auf ein Objekt sind möglich. Sichten können sich hierarchisch aus Teilsichten zusammensetzen. Im folgenden werden die wesentlichen Typen von Sichten beschrieben und entsprechende graphische Symbole eingeführt, aus denen ein ViewNet-Diagramm aufgebaut wird.

Objektsichten stellen eine Instanz einer bestimmten Objektklasse (z. B. Kunde) in unterschiedlichen Formen dar (Abb. 1). Zu unterscheiden sind hier die Komplettsicht, meist in Icon-Form (Icon-Sicht), die Attributsicht, die alle oder einige der Attribute darstellt und die Graphiksicht, die zur Darstellung beliebiger graphischer Repräsentationsformen einer Instanz herangezogen wird (z.B. als Karte, Geschäftsgrafik etc.).

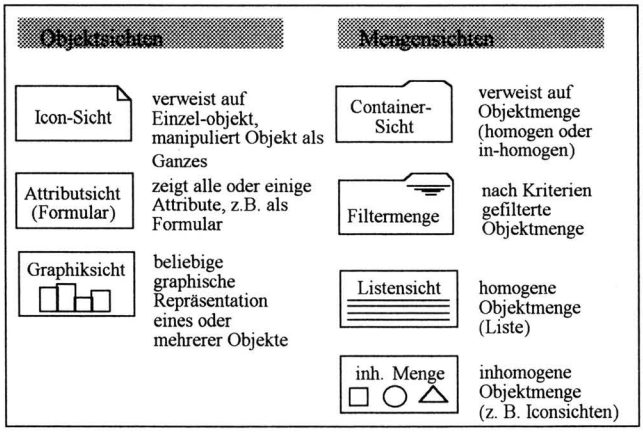

Abb.1: Objekt- und Mengensichten

Zur Strukturierung des objektorientierten Dialogs sind *Mengensichten* erforderlich, die entweder eine Gesamt- oder Teilmenge der Instanzen einer Objektklasse abbilden (homogene Mengen) oder Objekte unterschiedlichen Typs zusammenfassen (inhomogene Menge). Ein spezieller Fall einer Mengensicht sind gefilterte Mengen, durch die der Benutzer auf vor- oder eigendefinierte Untermengen der Datenobjekte zugreifen (z. B. alle fälligen Rechnungen). Ggf. kann der Benutzer solche Filterobjekte an der Benutzungsschnittstelle in flexibler Weise selbst anlegen und verwalten.

Funktionssichten zeigen eine oder mehrere Funktionen, ggf. mit Parametern, wie sie z. B. durch Menümasken oder modale Dialogboxen repräsentiert werden (Abb. 2). Objektattribute können in einer Funktionssicht mit angezeigt werden, sind aber nur in Verbindung mit der gewählten Funktion zu sehen (z. B. Suchattribute in einer Dialogbox 'Suchen'). *Aktorensichten* sind komplexe Funktionssichten, die den Benutzer durch einen zusammenhängenden Vorgang führen und in sehr unterschiedlicher Formen realisiert werden können(To-Do-Listen, Task-Leiste oder ein Assistenten-Fenster wie z.B. heute bei vielen Setup-Programmen verwirklicht).

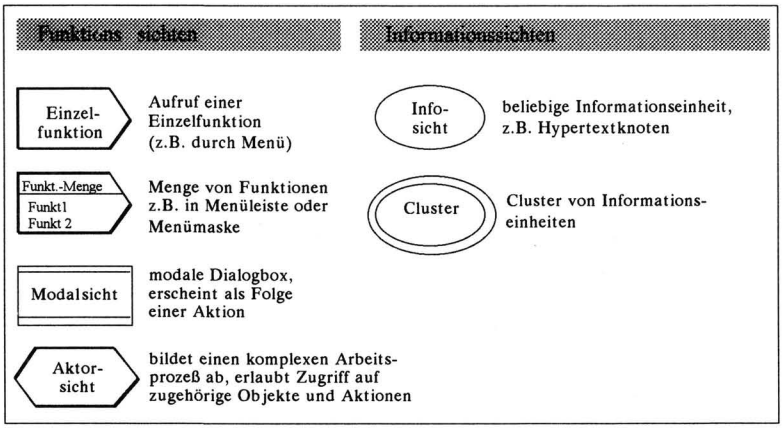

Abb.2: Dialog- und Informationssichten

Informationssichten werden in assoziationsorientierten Navigationen verwendet und können als Einzelknoten oder als Cluster dargestellt werden. Beispiele sind Inhalte eines Hilfesystems oder eines Hypermedia-Lernprogramms.

Abb. 3 zeigt *zusammengesetzte Sichten*. Typische, bei graphischen Schnittstellen häufig verwendete Konstrukte wie Master-Detail-Sichten oder Notebooks sind zwecks besserer Anschaulichkeit mit einem eigenen Symbol aufgeführt, das sich aus der konkreten visuellen Darstellung einer solchen Sicht ableitet. *Generische Bausteine* erlauben es, Teile einer Navigationsstruktur in einem Element zusammenzufassen und mit einem Parameter zu versehen. So kann z.B. ein aus mehreren Sichten bestehende Suchdialog für Instanzen einer Klasse mit dem Klassennamen parametrisiert werden. Gleichartige Suchdialoge für unterschiedliche Klassen (z.B. Auftrag, Kunde, Produkt) können dann durch ein einzelnes Bausteinsymbol mit Angabe der entsprechenden Klasse dargestellt werden. Hierdurch wird die Übersichtlichkeit verbessert und eine konsistente Navigationsentwicklung deutlich unterstützt.

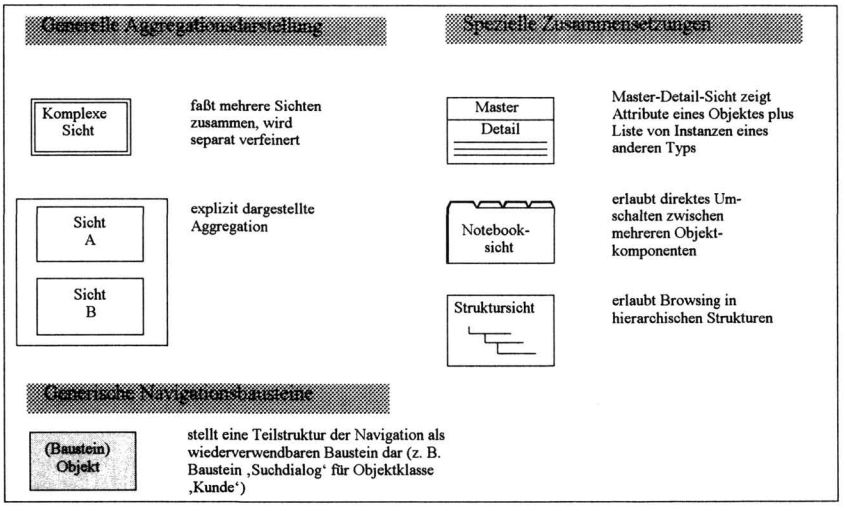

Abb. 3: Zusammengesetzte Sichten

4 Navigationsdarstellung mit ViewNets

Mit Hilfe der definierten Sichtentypen können Navigationsstrukturen in Diagrammform dargestellt werden. Für eine übersichtliche Darstellung der Navigation werden ViewNets in ihrer einfachen Form als Erreichbarkeitsgraphen dargestellt. Hierbei werden Ereignisse, die die Navigationsschritte auslösen, sowie eventuelle Bedingungen und Aktionen nicht dargestellt. Erweiterungen für eine detaillierte Spezifikation der Transitionen, etwa in Form von Dialognetzen [3], lassen sich jedoch in die Darstellung ohne Schwierigkeiten einführen. Für den Zweck der konzeptionellen Gestaltung des Dialoges ist es aber meist als günstiger anzusehen, zunächst von den Details der Dialogführung zu abstrahieren.

Abb. 4 zeigt eine typische objektorientierte Dialogstruktur, wie sie z. B. beim Entwurf eines Auftragsverwaltungsystemes auftreten könnte. Die oberste Ebene der Navigation (z. B. ein Desktop) entspricht zwecks Vereinfachung der Darstellung der Zeichenfläche. Ausgehend von einer Icon-Darstellung von Objektmengen (Kunden) auf dem Desktop kann über entsprechende Listenauswahl ('Öffnen') oder Suchmechanismen ('Suchen') ein Zugriff auf die Attributsicht eines bestimmten Objektes erreicht wird, in der lokal entsprechende Operationen (z. B. als Buttons oder Menüeinträge) zur Verfügung stehen. Die Attributsicht für Kunde ist hier als Notebook-Sicht dargestellt, da angenommen werden soll, daß es sich um umfangreiche Daten handelt. Von den Details dieser zusammengesetzten Struktur wird in diesem Schritt noch abstrahiert. Die Beziehungen zwischen Objektklassen (z. B. Kunde-Auftrag) werden durch Navigationspfade realisiert, wobei die Kardinalitäten der Beziehungen ggf. durch Mengensichten aufgelöst werden (hier z. B. durch die Liste 'Aufträge').

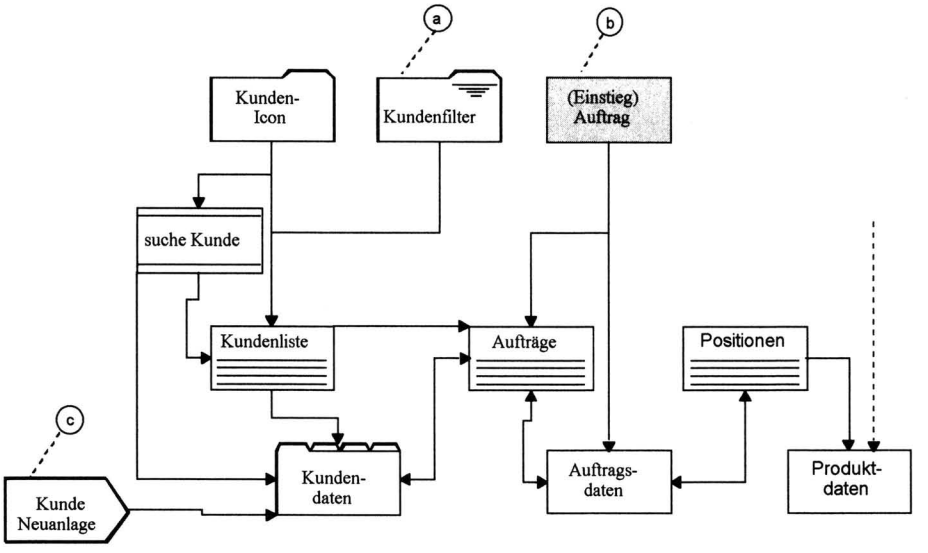

Abb. 4: Beispiel für die Darstellung einer Navigationsstruktur mit einem ViewNet

Das Element (a) zeigt zusätzlich die Möglichkeit, über gefilterte Objektmengen (z. B. Kunden Süddeutschland), die der Nutzer selbst definieren kann, wiederholt auf interessierende Objekte zuzugreifen. Die gesamte Substruktur für den Einstiegsdialog für Kunden (Icon-Suchdialog-Filter) kann zu einem generischen Baustein zusammengefaßt und in gleicher Weise für die Klasse 'Auftrag' zur Verfügung gestellt werden (b). Bausteindefinitionen erfolgen in separaten Diagrammen, so daß sie für unterschiedliche Dialoge wiederverwendet werden können. (c) zeigt einen zusätzlich zur objektorientierten Navigation eingeführten funktionalen Einstieg (z. B. über ein Icon oder einen Menüeintrag 'Kunde Neuanlage'). Funktionale Einstiege werden meist auf Effizienzgründen für häufig wiederkehrende Aufgaben eingeführt und können eine objektorientierte Grundstruktur überlagern.

Basierend auf einer anfänglichen ViewNet-Darstellung können die Navigationsabläufe im Bezug auf die Benutzeraufgaben weiter optimiert werden, wobei sich auch der Aufbau einzelner Sichten noch verändern kann. So wird man im Beispiel bezüglich der Darstellung von Auftragsdaten und Auftragspositionen bei üblichen Mengengerüsten zu dem Schluß kommen, daß eine Darstellung in einer Sicht (Master-Detail-Sicht) übersichtlicher ist und Dialogschritte sparen kann (Abb. 5). Die ViewNet-Darstellung stellt für solche Optimierungen eine anschauliche Grundlage dar, insbesondere auch für die Überführung von Objektbeziehungen mit Kardinalitäten 1:n und m:n in entsprechende Navigationsabläufe.

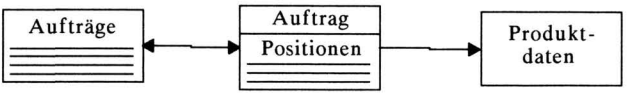

Abb. 5: Optimierung von Sichten und Abläufen für eine Teilstruktur des Beispiels in Abb. 4

ViewNets können nicht nur für Anwendungen mit Klasse-Instanz-Konstellationen, bei denen feste Sichtenstrukturen (Masken) mit wechselnden Inhalten gefüllt werden, sondern auch für Navigationen zwischen hypertext-organisierten Informationseinheiten oder Mischformen aus diesen Ansätzen verwendet werden. Abb. 6 zeigt einen möglichen Ausschnitt aus einem Internet-basierten Produktinformationssystem, bei dem Datenbank- und Hypertext-Komponenten gemeinsam auftreten.

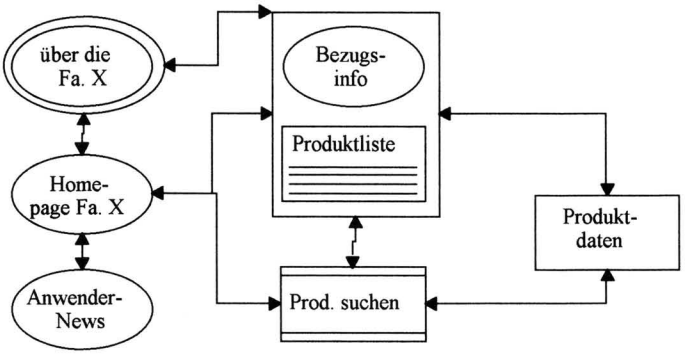

Abb. 6: Beispiel für eine gemischte Hypertext-/Datenbank-Navigation

Der Inhalt der einzelnen Sichten wird nicht in der graphischen Darstellung definiert, sondern z. B. in einem separaten Sichtendefinitionsschema aufgeführt. Dieses gibt die in einer Sicht dargestellten Attribut, ggf. die Darstellungsform sowie die für die Sicht verfügbaren Operationen an.

Diskussion

Die vorgestellte Modellierungstechnik ViewNet liefert im Gegensatz zu existierenden Dialogmodellierungstechniken eine reichhaltigere Beschreibung der Semantik der Sichten. Die Einführung unterschiedlicher Typen von Sichten unterstützt die Entwicklung der konzeptionellen Struktur einer Anwendung. Durch die Verwendung möglichst anschaulicher piktogrammhafter Symbole für die Darstellung der Sichten werden dem Entwickler (wie auch dem beteiligten Benutzer) visuelle Merkmale angeboten, die für ein schnelles Erfassen der angezeigten Struktur hilfreich sind. Der Entwickler kann in systematischer Weise Art und Funktion der verwendeten Sichten reflektieren und das der Navigationsstruktur zugrundeliegende Prinzip leichter erkennen.

Das Problem der Dialogstrukturierung kann damit leichter in einzelne Teilschritte zerlegt werden. So kann z. B. die Entwicklung der objektorientierten Navigationsgrundstruktur einer An-

wendung (die bei graphischen Oberflächen meist sinnvoll ist) getrennt werden von der Erstellung zusätzlicher aufgaben- oder prozeßbezogener Dialogpfade. Die Ableitung des Navigationsmodells aus Objekt- und Aufgabenmodellen der Anforderungsanalyse kann in einer systematischen, regelgestützten Weise erfolgen [7], [8].

Der Übergang zu einer detaillierten Beschreibung des dynamischen Dialogverhaltens kann in die verwendete Diagrammtechnik ohne Schwierigkeiten integriert werden, z. B. mit den Konstrukten von (Petri-Netz-basierten) Dialognetzen. Der vorgestellte Ansatz geht allerdings von der Annahme aus, daß insbesondere in den frühen Gestaltungsphasen die konzeptionelle Struktur des Dialogs Vorrang vor der Dynamik des Dialogablaufs hat.

Die ViewNet-Technik bzw. Vorläuferversionen wurden in einer Reihe von Entwicklungsprojekten angewendet und qualitativ bewertet. Als besonders nützlich erwies sich die Darstellung dabei in Gruppendiskussionen, in denen gemeinsam die Struktur einer Anwendung erarbeitet wurde. Zukünftige Arbeiten werden sich auf eine entsprechende Werkzeugunterstützung, gerade auch für Gruppensituationen konzentrieren.

Literatur

[1] Denert, E. (1977): Specification and design of dialog system with state transition diagrams. In Morlet, E. & Ribbens, D. (Eds.), Proc. Int. Computing Symposium. Amsterdam: Noth-Holland, 417-427.

[2] Jacob, R.J.K. (1986): A specification language for direct manipulation user interfaces. ACM Transactions on Graphics, Vol. 5, 283-317.

[3] Janssen, Chr. (1993): Dialognetze zur Beschreibung von Dialogabläufen in graphisch-interaktiven Systemen. In K.-H. Rödiger (Hrsg.): Software-Ergonomie '93. Stuttgart: Teubner, 67-76.

[4] Nielsen, J. (1990): The art of navigating through hypertext. Communications of the ACM, March 1990, Vol. 33, No. 3, 296-310

[5] Berk, E. & Devlin, J. (1991): Hypertext/Hypermedia Handbook. New York: McGraw-Hill.

[6] Ziegler, J. & Janssen, Chr. (1995): Aufgabenbezogene Dialogstrukturen für Dialogsysteme. In: Böcker, H.-D. (Hrsg.): Software-Ergonomie '95 - Mensch-Computer-Interaktion - Anwendungs-bereiche lernen voneinander. Fachtagung Germ. Chapt. ACM , GI u. a. (20.-23.2.1995, Darmstadt), 395-406.

[7] Greutmann, Th. (1993): Datenmodellierung und aufgabengerechte Dialoge: ein Synchronisierungsproblem. In: Rödiger, K.-H. (Hrsg.): Software-Ergonomie '93 (Bremen 15.-17.3.1993), Stuttgart: Teubner.

[8] Janssen, C.; Weisbecker, A. & Ziegler, J. (1993): Generating user interfaces from data models and dialogue net specifications. In Proceedings of INTERCHI (Amsterdam, 24-29 April), New York: ACM, 418-423.

Adresse des Autors

Dr.-Ing. Jürgen Ziegler
Fraunhofer-Institut für Arbeitswirtschaft und Organisation
Nobelstraße 12, 70569 Stuttgart
Email:juergen.ziegler@iao.fhg.de