Ein Vorgehensmodell zur integrierten Anforderungsanalyse von Benutzungsschnittstelle und Anwendung

Georg Kösters, Josef Voss
FernUniversität Hagen

Zusammenfassung

Wir stellen in diesem Artikel ein Vorgehensmodell für die frühen Phasen der Softwareentwicklung vor, das in besonderer Weise die Entwicklung direkt-manipulativer graphischer Benutzungsschnittstellen einbezieht. Anforderungen an Anwendung und Benutzungsschnittstelle werden aufeinander abgestimmt und parallel entwickelt. Zentrales Beschreibungsmittel für Anforderungen an die Benutzungsschnittstelle sind abstrakte Sichten, die realisierungsunabhängig formuliert werden. Aus der vollständigen Anforderungsanalyse kann die Softwarearchitektur von Benutzungsschnittstelle und Anwendung systematisch entwickelt werden.

1 Einleitung

Die Entwicklung von direkt-manipulativen graphischen Benutzungsschnittstellen ist bekanntermaßen mit erheblichem Aufwand verbunden. Es ist also dringend geboten, diesen Teil der Software in geeigneter Weise in den Softwareentwicklungsprozeß zu integrieren. Vielfach wird in diesem Zusammenhang die Idee des Prototyping propagiert, um durch die frühzeitige Erstellung einer lauffähigen Benutzungsschnittstelle gemeinsam mit dem Auftraggeber die Anforderungen an die Funktionalität und die Benutzungsoberfläche validieren zu können. Bei Systemen mit komplexem Funktionsumfang ergeben sich aber erhebliche Probleme: Wie gelangt man schnell und systematisch zu einem Prototypen und wie gewinnt man aus einem Prototypen präzise und verbindliche Vorgaben für die weitere Entwicklung? Zur Lösung derartiger Probleme schlagen wir ein Vorgehen für die frühen Phasen der Entwicklung vor, in dem die Prototyperstellung als ein nachgeordneter Schritt innerhalb eines differenzierten Vorgehensmodells auftritt.

Wir stellen im folgenden ein Vorgehen vor, das die traditionelle Aufteilung in Anforderungsanalyse und Softwareentwurf mit entsprechenden Dokumenten im wesentlichen beibehält, aber die jeweiligen Tätigkeiten und Ergebnisse neu definiert. Wir plädieren dabei insbesondere für die Formulierung von Anforderungen an die Benutzungsschnittstelle bereits in der Anforderungsanalyse.

Wir formulieren Benutzungsschnittstellenanforderungen realisierungsunabhängig und lassen insbesondere Details von Bildschirmlayout und Dialogablauf noch offen. Die Anforderungen werden allerdings so weit präzisiert, daß die wesentliche Struktur der Benutzungsoberfläche vorgegeben ist, so daß die Softwarearchitektur von Benutzungsschnittstelle und Anwendung daraus systematisch entwickelt werden kann, und zwar unabhängig von und parallel zur weiteren detaillierten Festlegung von Bildschirmlayout und Dialogablauf.

2 Verwandte Arbeiten

Die methodische Integration der Benutzungsschnittstellenentwicklung in den Software Life Cycle ist bislang kaum systematisch verfolgt worden. Die meisten Ansätze beschäftigen sich mit der möglichst effizienten Konstruktion von Benutzungsschnittstellen. Eine Forschungsrichtung ist die modellbasierte Generierung. Im allgemeinen wird dabei zunächst ein bestimmtes Modell der Anwendung erstellt, das dann zur (halb-) automatischen Konstruktion der Benutzungsschnittstelle genutzt wird (z.B. [2], [5], [7]). Auf zwei dieser Ansätze gehen wir kurz ein:

- Bei den Arbeiten von Foley u.a. (z.B. [5]) besteht das Anwendungsmodell aus einer präzisen Definition von Operationen, die mit der Oberfläche realisiert werden, einschließlich deren Semantik in Form von Vor- und Nachbedingungen. Kritik: In der Analyse hat man es zunächst mit unscharfen Anforderungen zu tun. Ein solches Anwendungsmodell kann also nicht gleich zu Beginn der Entwicklung in einem Schritt bestimmt werden.

- Ein differenzierteres Vorgehen wird von Ziegler u.a. verfolgt (z.B. [7]): Ausgehend von einer Problemweltanalyse in Form eines ER-Modells werden Sichten zur Aufgabenbearbeitung definiert und mit Hilfe angereicherter ER-Modelle dokumentiert. Anschließend wird mit Dialognetzen der globale Dialogablauf definiert. Ablauf und Sichten dienen als Ausgangsmaterial für die Generierung der Benutzungsschnittstelle. Kritik: Einzelne Aufgaben werden nicht explizit formuliert. Dialognetze zur Darstellung des globalen Ablaufs sind wenig intuitiv und basieren zum Teil schon auf syntaktischen Details der Oberfläche.

Ein allgemeines Problem dieser Ansätze besteht darin, daß die ersten Schritte der Entwicklung stark auf die jeweiligen Generierungswerkzeuge zugeschnitten sind, wobei insbesondere zu früh Details festgelegt werden. Außerdem ist das Vorgehen jeweils auf ein eng umrissenes Anwendungsgebiet begrenzt.

3 Das Vorgehensmodell im Überblick

Zur Formulierung von Anforderungen an graphisch-interaktive Programme werden, vereinfacht ausgedrückt, Gegenstände und Tätigkeiten untersucht. Wir behandeln beide Aspekte gleichrangig und verfolgen in der Anforderungsanalyse zwei parallele Entwicklungslinien: Eine an den Gegenständen orientierte Problemweltmodellierung sowie eine an den Tätigkeiten orientierte Aufgabenanalyse.

Der zeitliche Ablauf der einzelnen Entwicklungsschritte wird in Abb. 1 idealisiert dargestellt. Geänderte Anforderungen oder erkannte Fehler können natürlich jederzeit Rückkopplungen erfordern. Ein Entwicklungsschritt in einer bestimmten Ebene ist von den Ergebnissen der darüberliegenden Schritte abhängig. Ein erster stabiler Zwischenzustand – angedeutet durch die gestrichelte Linie – ist nach der Aufgabenanalyse und bei der Problemweltmodellierung vor der Bestimmung der Operationen erreicht. Das Ergebnis der gesamten Anforderungsanalyse setzt sich aus dem vollständigen Problemweltmodell, benannten Objekten sowie abstrakten Sichten zusammen. Auf die einzelnen Punkte werden wir im folgenden genauer eingehen.

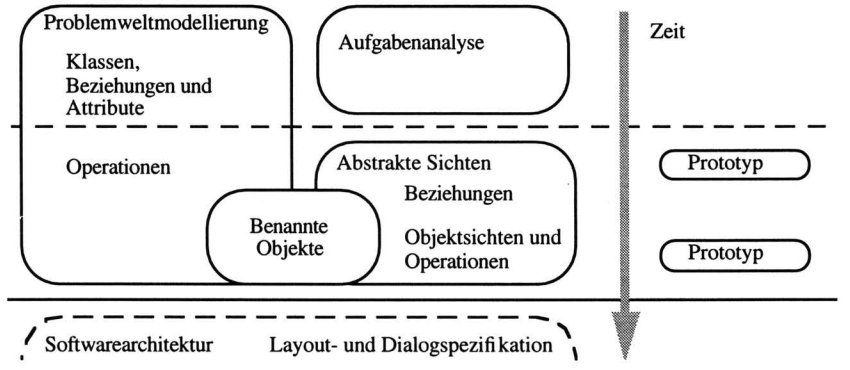


Abb.1: Die Entwicklungsschritte des Vorgehensmodells im Überblick.

4 Aufgabenanalyse und Problemweltmodellierung

In der Aufgabenanalyse gehen wir der Frage nach, was ein Benutzer mit dem Programm bewerkstelligen will. Ausgehend von einer informellen Problemstellung

oder einer Grobanalyse, die eine Sammlung potentiell zu realisierender Anforderungen enthält, wird festgelegt, welche Aufgaben tatsächlich mit dem Programm bearbeitet werden, und welche nicht – z.B. weil ihre Realisierung zu aufwendig ist, sie durch existierende Programme oder von Hand erledigt werden.

In der Regel wird ein Softwareprodukt nicht ein völlig neues Aufgabengebiet definieren, sondern existierende Arbeitsvorgänge sollen sicherer und effizienter bearbeitet werden. Zur Bestimmung der Aufgaben orientieren wir uns daher an (existierenden) Arbeitsvorgängen und Arbeitsschritten sowie an den zu bearbeitenden Gegenständen. Die Aufgaben werden umgangssprachlich formuliert ggf. weiter zerlegt oder zusammengefaßt (vgl. [3], [9]). Einzelne Aufgaben werden nicht bis auf eine syntaktische oder lexikalische Ebene heruntergebrochen. Diese Ebenen werden erst in späteren Entwicklungsphasen betrachtet. Ihre Beschreibungen bleiben vielmehr auf einer konzeptuellen Ebene und benutzen das Vokabular der Problemwelt.

Zur Strukturierung werden Aufgaben hierarchisch angeordnet: Komplexere Aufga-

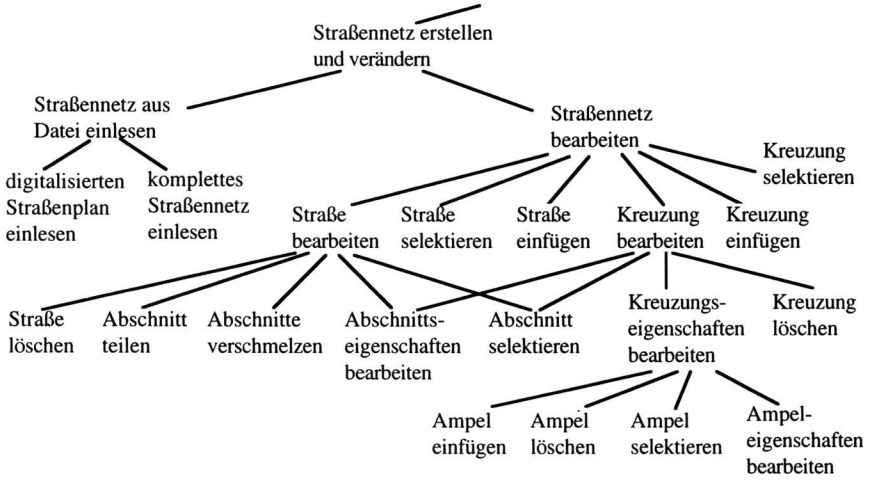


Abb.2: Auszug aus der Aufgabenhierarchie des Verkehrssimulationssystems.

ben setzen sich aus einfacheren Teilaufgaben zusammen. Zusammengefaßte Aufgaben sind durch inhaltliche Nähe und insbesondere durch die Bearbeitung zusammengehöriger Gegenstände gekennzeichnet. Als Beispiel betrachten wir ein Verkehrssimulationssystem, dessen Aufgabenhierarchie Abb. 2 auszugsweise darstellt. Die Beziehungen zwischen den Aufgaben haben hier noch keine präzise

Semantik und werden als „ist Teilaufgabe von" oder auch als expliziter Wechsel zur Bearbeitung einer anderen Aufgabe interpretiert.

Parallel zur Aufgabenanalyse wird in der Problemweltmodellierung die Struktur der vom Programm bearbeiteten Gegenstände untersucht. Die anfängliche Problemstellung und ggf. auch schon Ergebnisse der Aufgabenanalyse können bereits insofern einfließen, als daß sie bestimmen, welche Gegenstände überhaupt relevant sind und bis zu welchem Detaillierungsgrad deren Struktur untersucht wird. Zur Problemweltmodellierung verwenden wir die OOA-Methode und -Notation von Coad/Yourdon [4]. Im ersten Schritt verzichten wir allerdings auf die Einbeziehung von Operationen (Diensten). Abb. 3 zeigt einen Auszug der Problemweltmodellierung des Verkehrssimulationssystems.

Die gegenseitige Beeinflussung von Problemweltmodellierung und Aufgabenanalyse ist hauptsächlich durch die Tatsache begründet, daß zur Formulierung von Aufgaben Objekte, deren Attribute, Beziehungen und Komponenten aus dem OOA-Modell verwendet werden. Gegenstände, die in der Aufgabenanalyse auftreten, aber im OOA-Modell nicht berücksichtigt werden, deuten auf ein unzureichendes OOA-Modell hin. Umgekehrt müssen Objekte des OOA-Modells, die sich keiner Aufgabe zuordnen lassen, hinterfragt werden. Im Idealfall werden beide Schritte

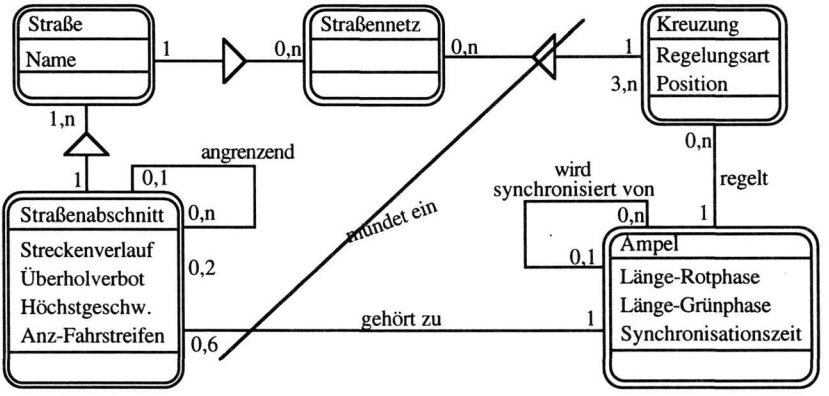


Abb.3: Auszug aus dem OOA-Modell des Verkehrssimulationssystems.

deshalb parallel und mit häufigen Rückkopplungen durchgeführt. Mit Beendigung dieser Schritte erreicht die Anforderungsanalyse einen im Hinblick auf Konsistenz und Vollständigkeit zu validierenden Zwischenstand.

5 Abstrakte Sichten

Als Analyseendergebnis ist der vorliegende Zwischenstand noch unzureichend. Aus Sicht des Systementwicklers fehlt die Präzisierung der operativen Anforderungen als Vorgabe für die weitere Entwicklung. Aus Sicht des Auftraggebers fehlen Skizzen oder ein Oberflächenprototyp, um die Vollständigkeit bzw. Richtigkeit der Anforderungsanalyse beurteilen zu können. Wir ergänzen deshalb die Anforderungsanalyse und präzisieren die bislang lediglich umgangssprachlich formulierten Aufgaben und halten dies in geeignet formalisierter Weise fest. Zu diesem Zweck stellen wir mit abstrakten Sichten und benannten Objekten im folgenden zwei wichtige Hilfsmittel vor, die den Dialogablauf strukturieren und den Zusammenhang zu den dabei behandelten Gegenständen bzw. Objekten herstellen.

Unter abstrakten Sichten verstehen wir Fenster bzw. Teilfenster, die dadurch charakterisiert sind, daß sie bestimmte Attribute, Beziehungen und Komponenten von benannten Objekten darstellen sowie Operationen anbieten, die zu Veränderungen an abstrakten Sichten oder benannten Objekten führen. Wir sprechen von abstrakten Sichten, weil das Augenmerk auf dem Informationsgehalt und den logischen Verbindungen der zugehörigen Objekte und nicht auf deren konkretem Aussehen liegt. Den Ausgangspunkt zur Bestimmung der abstrakten Sichten bilden die zentralen, voraussichtlich am häufigsten auszuführenden Aufgaben der Aufgabenanalyse, die einen in sich geschlossenen Arbeitsschritt repräsentieren. Wir assoziieren mit jeder zentralen Aufgabe eine abstrakte Sicht.

Eine abstrakte Sicht ist durch die enthaltenen benannten Objekte und angebotenen Operationen charakterisiert. Dabei können nicht nur benannte Objekte in mehreren abstrakten Sichten benutzt werden, auch Operationen und bestimmte Darstellungen benannter Objekte – Objektsichten genannt – finden sich in unterschiedlichen abstrakten Sichten wieder. So wird beispielsweise dieselbe Darstellung des Straßennetzes sowohl beim Verändern des Netzes als auch beim Beobachten einer Simulation verwendet. Wir definieren deshalb benannte Objekte, Objektsichten, sowie Operationen zunächst unabhängig von einer bestimmten abstrakten Sicht. Jede abstrakte Sicht ist dann durch die Namen der enthaltenen benannten Objekte, der zugehörigen Objektsichten und der angebotenen Operationen definiert.

5.1 Benannte Objekte

Eine genauere Betrachtung der Aufgabenbeschreibungen liefert die Erkenntnis, daß dort im Unterschied zur Problemweltmodellierung von bestimmten Gegenständen gesprochen wird, und zwar nicht von abstrakten Begriffen oder Objekttypen oder

-klassen, sondern von konkreten Exemplaren, die im Kontext einer Aufgabe eine bestimmte Bedeutung haben. Wir extrahieren diese Gegenstände als sog. benannte Objekte.

Für jedes benannte Objekt wird festgehalten, welchen Typ es hat, in welchen abstrakten Sichten es bekannt ist, bzw. benutzt wird und ggf. von welchen anderen benannten Objekten es abhängig ist (als Komponente, Element oder Teilmenge). Beim Typ der benannten Objekte handelt es sich im allgemeinen um Strukturen, etwa Listen oder Mengen, die Objekte aus den Klassen der Problemweltmodellierung enthalten. Existieren mehrere Ausprägungen einer Sicht, so existieren auch von den in dieser Sicht lokal bekannten Objekten Ausprägungen in entsprechender Anzahl. Zu jeder Ausprägung der abstrakten Sicht „Abschnittsbearbeitung" existiert beispielsweise eine Ausprägung des benannten Objekts „bearbeiteter Abschnitt". Die folgende Aufstellung beschreibt einige benannte Objekte des Verkehrssimulationssystems.

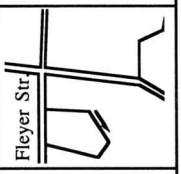
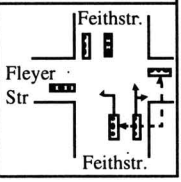
benanntes Objekt: Name	*Menge und Typ*	*bekannt/benutzt in abstrakter Sicht*	*Teilkomponente von*
Straßennetz	ein Straßennetz	global bekannt	
Abschnittselektion	0-2 angrenzende Straßenabschnitte	Netz- und Kreuzungsbearbeitung	Straßennetz
bearbeiteter Abschnitt	ein Straßenabschnitt	Abschnittbearbeitung	Straßennetz

5.2 Objektsichten – Skizzen

In einer abstrakten Sicht werden genau die benannten Objekte dargestellt, die für die assoziierten Aufgaben relevant sind, allerdings nicht mit all ihren Eigenschaften, sondern vielmehr auf Sicht-spezifische Aspekte reduziert. Eine Objektsicht definiert, welche Attribute und Beziehungen des benannten Objekts auf dem Bildschirm dargestellt werden. Bei zusammengesetzten Objekten wird zusätzlich festgelegt, welche Komponenten sichtbar sind, d.h. bis zu welcher Ebene und mit welchen Attributen und Beziehungen auf der jeweiligen Ebene ein Objekt dargestellt wird.

Auch bei der Festlegung der Objektsichten geht es uns lediglich um den Informationsgehalt und nicht um deren Präsentation auf dem Bildschirm. Falls erforderlich ergänzen wir die textuelle Beschreibung einer Objektsicht um eine Skizze, z.B. wenn die Entwicklung eines geeigneten Bildschirmlayouts anwendungsspezifische Kenntnisse erfordert. Eine Skizze beschreibt also Anforderungen hinsichtlich einer bestimmten graphischen Darstellung, die allein durch die textuelle

Beschreibung der Objektsicht nicht ausgedrückt werden können. Die folgende Aufstellung gibt zwei durch Skizzen ergänzte Objektsichten unseres Beispiels wieder.

Name der Objekt-sicht	*dargest. benanntes Objekt*	*dargest. Komponenten, in Beziehung stehende Objekte*	*dargest. Beziehungen*	*dargest. Attribute*	*Skizze, Darstellungs-vorgabe*
Netz-topologie-sicht	Straßen-netz	Straßen und deren Abschnitte, Kreuzungen	angrenzend, mündet-ein	Straße: Name, Abschnitt: Streckenverlauf Kreuzung: Position	
Ampel-kreuzungs-sicht	bearbeitete Kreuzung	Straßen und Ampeln an der Kreuzung	wird synchronisiert von	Straße: Name, Ampel: Richtung	

5.3 Operationen

Bisher beschreiben Aufgaben die Tätigkeiten, die Benutzer mit dem System durchführen. Die in den Aufgaben bisher nur implizit enthaltenen vom System bereitzustellenden Operationen werden jetzt explizit beschrieben. Zur Beschreibung einer Operation werden im wesentlichen nur die beteiligten benannten Objekte angegeben. Beteiligt sind zum einen die Argumente der Operation, zum anderen die als Folge ihrer Ausführung veränderten benannten Objekte.

Die folgende Aufstellung beschreibt einige Operationen der abstrakten Sicht „Netzbearbeitung". Das Löschen von Straßenabschnitten wird von verschiedenen Sichten angeboten, deshalb ist bei der Operation „Abschnitt löschen" die Verwendung eines Parameters vom Typ „Abschnitt" angezeigt.

Operation: Name und Beschreibung	*Argumente/ Parameter*	*veränderte benannte Objekte*	*veränderte abstrakte Sichten*
Abschnitt (de)selektieren		Abschnittselektion	
Wechsel zur Abschnittbearbeitung	Abschnitt-selektion		erzeugt neue Abschnitt-bearbeitung mit Abschnittselektion

Abschnitt löschen	Straßennetz, ein Abschnitt	Straßennetz, Abschnitt-selektion, bearbeitete Kreuzungen	zerstört Abschnittbear-beitung, die den gelöschten Abschnitt bearbeitet

6 Beziehungen zwischen abstrakten Sichten

Die einzelnen Fenster einer Oberfläche sind nicht völlig unabhängig voneinander. Wir geben im folgenden die wichtigsten Abhängigkeiten an und modellieren sie als Beziehungen zwischen abstrakten Sichten.

- Wechsel zwischen sichtbaren Fenstern: Im Normalfall können Benutzer zwischen aktiven Sichten bzw. der Bearbeitung der assoziierten Aufgaben beliebig wechseln. Solche Wechsel bezeichnen wir als implizit.

- Explizites Sichtbarmachen von Fenstern: Ein expliziter Wechsel wird durch eine in der Ausgangssicht vorgegebenen Operation ausgelöst. Explizite Wechsel sind durch die hierarchische Anordnung der Aufgabenanalyse vorstrukturiert.

- Variabel viele Ausprägungen von Fenstern: Sichten können dynamisch erzeugt und zerstört werden. Die Erzeugung einer neuen Ausprägung einer abstrakten Sicht ist immer mit einem expliziten Wechsel in die erzeugte Sicht verbunden.

- Fenster können voneinander abhängig sein, etwa in der Form, daß die Zerstörung eines Fensters die Zerstörung von untergeordneten Fenstern nach sich zieht: Eine Kontrollbeziehung strukturiert abstrakte Sichten baumartig. Die Deaktivierung bzw. Zerstörung einer Sicht ist fest mit der Deaktivierung bzw. Zerstörung aller von ihr kontrollierten Nachfolger verbunden. Das Aktivieren einer Sicht erzwingt die Aktivierung aller Vorgänger innerhalb des Kontrollbaumes.

- Die Bearbeitung verschiedener, sich gegenseitig ausschließender Aufgaben in einem Fenster: Ein wechselseitiger Ausschluß verbietet die gleichzeitige Aktivierung von Gruppen abstrakter Sichten. Die Aktivierung einer Sicht einer derzeit inaktiven Gruppe bewirkt die Deaktivierung aller aktiven Sichten der derzeit aktiven Gruppe. Der gegenseitige Ausschluß betrifft genau definierte Gruppen von Sichten und ist auf einen lokalen Bereich begrenzt.

- Modale Dialoge: Ist die Bearbeitung einer Sicht exklusiv, so sprechen wir von einer modalen Sicht. Diese kann nur durch explizite Wechsel erreicht und wieder verlassen werden. Die Aktivierung einer modalen Sicht wirkt sich global aus, da alle impliziten Navigationsmöglichkeiten gesperrt sind, bis die modale Sicht durch einen expliziten Wechsel zerstört bzw. deaktiviert wird.

Explizite Wechsel sind Operationen, die abstrakte Sichten erzeugen, zerstören, aktivieren oder deaktivieren. Aufgrund ihrer Bedeutung führen wir die expliziten Wechsel zwischen abstrakten Sichten als Operationen auf.

Wir fassen die abstrakten Sichten zusammen mit den wichtigsten, Sicht-übergreifenden Informationen in einer mit State Charts [6] verwandten graphischen Darstellung zusammen. Rechtecke repräsentieren abstrakte Sichten, wobei die Zuordnung über ihre Namen erfolgt. Implizite Wechsel zwischen Sichten werden nicht, explizite Wechsel durch gerichtete Kanten dargestellt, wobei die Richtung der Kante die Navigationsrichtung wiedergibt. Die graphische Darstellung eines expliziten Wechsels wird durch einen Kreis ergänzt, falls dieser Wechsel die Erzeugung einer neuen Ausprägung der abstrakten „Ziel"-Sicht beinhaltet. Die Kontrollbeziehung wird durch eine Kante mit einem Dreieck dargestellt, wobei das Dreieck auf die kontrollierende Sicht weist. Der gegenseitige Ausschluß von Sichten wird durch ein in verschiedene Bereiche geteiltes, gestricheltes Rechteck dargestellt. Modale Sichten werden durch eine doppelte Umrandung gekennzeichnet. Abb. 4 zeigt die aus der Aufgabenanalyse abgeleiteten Beziehungen zwischen abstrakten Sichten unseres Beispiels.

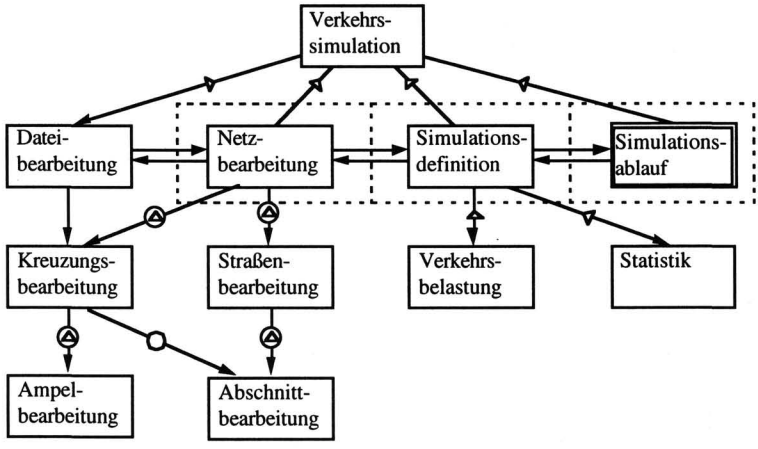


Abb.4: Die graphische Darstellung der abstrakten Sichten des Verkehrssimulationssystems.

7 Diskussion und Ausblick

Das beschriebene Vorgehensmodell erweitert und präzisiert verschiedene Vorversionen. Es stellt ein Gerüst dar, das nicht immer in allen Einzelheiten durchlaufen wird. Je nach Art des Problems sind Vereinfachungen bzw. Spezialisierungen angebracht. Vereinfachungen sind z.B. bei Oberflächen angezeigt, die nur einen geringen Graphikanteil besitzen und sich hauptsächlich aus Eingabemasken zusammensetzen. Spezialisierungen ergeben sich insbesondere für GIS- und CASE-Anwendungen.

Die in Objektsichten enthaltenen Skizzen sind als Mittel zur Veranschaulichung nur begrenzt tauglich. Deshalb werden – wie in Abb. 1 angedeutet – begleitend Prototypen erstellt, wobei es sich in der Anforderungsanalyse um schnell zu erstellende Wegwerf-Prototypen handelt. Wir halten ein abgestuftes Vorgehen für angebracht, wie es z.B. in [1] vorgeschlagen wird. Ein erster Schritt kann die Verwendung eines DTP- oder Zeichenprogramms mit Hypertext-Möglichkeiten sein. Nach der Definition von Objektsichten und Operationen können Prototypen mit detaillierterem Layout und Verhalten erstellt werden. Wir haben an dieser Stellen durchaus gute Erfahrungen mit einer Kombination aus DTP-Programm und User-Interface-Builder gemacht.

Die Anwendbarkeit einer Methode, die die Erstellung umfangreicher und detaillierter Dokumente mit sich bringt, hängt wesentlich von unterstützenden Werkzeugen ab. Die Entwicklung solcher Werkzeuge ist deshalb eines unserer nächsten Ziele. Längerfristig verfolgen wir die Weiterführung der methodischen Überlegungen in Richtung Softwareentwurf. Vergleichende Entwicklungen in verschiedenen Umgebungen zeigen, daß zahlreiche Regeln zur systematischen Umsetzung von Problemweltmodell, benannten Objekten und abstrakten Sichten in einen Softwareentwurf existieren. Der Schwerpunkt bei unseren weiteren Überlegungen wird auf objektorientierte Umgebungen mit integriertem UI-Builder oder UIMS, insbesondere dem von uns entwickelten DIWA-System [8], liegen.

8 Literatur

[1] Bösze J. und Aschacher, D. *Prototyping mit "Hyper-Tools"*. Proceedings Software Ergonomie '91, Teubner, Stuttgart, 1991, 119-129

[2] Balzert, H. *Der Janus-Dialogexperte: Vom Fachkonzept zur Dialogstruktur.* Proceedings Softwaretechnik 93, 62-72

[3] Bass, L. und Coutaz, J. *Developing Software for the User Interface.* Addison Wesley, Reading Massachusetts, 1991

[4] Coad, P. und Yourdon, E. *Object-Oriented Analysis.* Prentice Hall, Englewood Cliffs, NJ, 1990.

[5] Foley, J., Kim, W., Kovacevic, S. und Murray K. *Defining Interfaces at a High Level of Abstraction.* IEEE Software 6,1 (1989), 25-32.

[6] Harel, D. *On Visual Formalisms.* Communcations of the ACM 31,5 (1988), 514-530.

[7] Janssen, C., Weisbecker, A. und Ziegler, J. *Generierung graphischer Benutzerschnittstellen aus Datenmodellen und Dialognetzspezifikationen.* in: Züllighoven, et. al. (Hrsg.) Requirements Engineering '93: Prototyping. Teubner, Stuttgart, 1993, 335-349.

[8] Six, H.W. und Voss, J. *A Software Engineering Perspective to the Design of a User Interface Framework*, in: Proceedings CompSAC 92, IEEE Computer Society Press, Los Alamitos, 1992, 128-134

[9] Ziegler, J. *Aufgabenanalyse und Funktionsentwurf.* in: Balzert, H., et. al. (Hrsg.) Einführung in die Software Ergonomie. de Gruyter, Berlin, 1988, 231-252.

Georg Kösters, Josef Voss
Lehrgebiet Praktische Informatik III
FernUniversität Hagen
58084 Hagen
e-mail: georg.koesters@fernuni-hagen.de
josef.voss@fernuni-hagen.de