OBSM: Objektorientierte Modellierung von Benutzungsschnittstellen

Birgit Kneer
Universität - GH Paderborn
Heinz Nixdorf Institut
D-33098 Paderborn
kne@hni.uni-paderborn.de

Gerd Szwillus
Universität - GH Paderborn
Fachbereich Mathematik/Informatik
D-33098 Paderborn
szwillus@uni-paderborn.de

Zusammenfassung
Die Entwicklung von Benutzungsschnittstellen ist dadurch gekennzeichnet, daß mehrere qualitativ verschiedene Aspekte gleichzeitig oder in aufeinanderfolgenden Entwicklungsphasen Gegenstand der Betrachtung sind. Die Aspekte für sich werden jeweils von dafür entwickelten Spezifikationssprachen und -techniken unterstützt - eine gesamtheitliche Darstellung fehlt jedoch, so daß der Entwickler komplexe Transformationen zwischen verschiedenen Sichten durchführen muß. Wir schlagen eine visuelle Spezifikationstechnik vor, die verschiedene Aspekte des Entwurfs von Benutzungsschnittstellen in einer durchgängigen Notation vereinigt. Sie beruht im wesentlichen auf dem Paradigma der Objektorientierung, Constraints, der Darstellung von Interaktionssequenzen und temporalen Relationen.

1 Einleitung

Die Entwicklung von Benutzungsschnittstellen ist durch die Notwendigkeit gekennzeichnet, eine Vielzahl verschiedener Aspekte betrachten zu müssen. Diese sind bereits für sich genommen komplex, müssen aber auch in ihren gegenseitigen Abhängigkeiten stimmig behandelt werden. Diese Sichten betreffen qualitativ verschiedene Aspekte, die gleichzeitig oder aber in aufeinanderfolgenden Entwicklungsphasen zu behandeln sind (vgl. Abbildung 1).

(1) Zunächst ist hier der komplexe, mehrschichtige Übergang von einer semiformalen Beschreibung der Benutzeraufgaben (Aufgabenmodell, wie etwa in MAD [19]) zu einer Spezifikation des Verhaltens der Benutzungsschnittstelle (Dialogmodell) zu nennen. Diese Transformation beinhaltet den Übergang von der postulierten oder tatsächlichen Handlungsweise des Benutzers zum Verhalten eines Artefakts, der diese Benutzeraktionen ermöglicht, aufgreift und widerspiegelt.

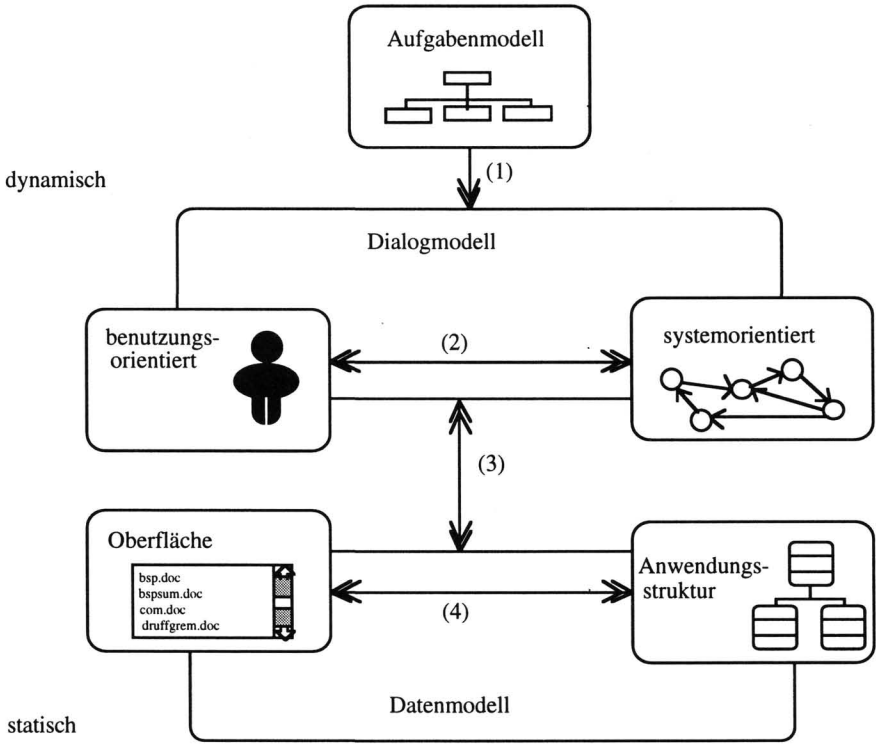

Abb. 1: Entwurfsaspekte von Benutzungsschnittstellen

(2) Innerhalb der Dialogmodellierung kann man eine benutzerorientierte Sichtweise (wie sie etwa mit UAN [7] beschreibbar ist) oder eine systemorientierte Sichtweise (zum Beispiel beschreibbar durch Transitionsnetze [8], DSN [13]) einnehmen. In der Sicht von außen sind die Benutzeraktionen, ihre Reihenfolge und die erfahrbaren Rückmeldungen Entwurfsgegenstand; in der Systemsicht geht es um die Gestaltung der Struktur der Abarbeitung dieser Interaktionen und ihrer Abbildung auf Softwarekomponenten (Objekte).

(3) Die Betrachtung der Softwarekomponenten einer Benutzungsschnittstelle muß einerseits mit Bezug auf die Datenstrukturen (z.B. Objekte, Klassen, Verbünde) geschehen (statische Sicht), andererseits aber auch das dynamische Verhalten dieser Komponenten berücksichtigen. Dieses generelle Grundproblem von Software-Entwicklung bzw. Programmierung wird bei Benutzungsschnittstellen durch den hohen Grad an Dynamik durch die Benutzerinteraktion verschärft.

(4) Schließlich stellt die Kooperation der Benutzungsschnittstelle mit der Applikation ein klassisches Problem des Entwurfs von Benutzungsschnittstellen dar. Die gewünschte Trennung und gleichzeitige enge Kopplung der beiden Anteile motivierte zunächst das Seeheim-Modell [15] und führte mittlerweile zu Ansätzen kooperierender Objektmodelle [3].

Für einzelne dieser Aspekte existieren jeweils Notationen und Spezifikationsmethoden, sowie zum Teil Werkzeuge zum effizienten Arbeiten mit diesen Notationen. Problematisch bleiben jedoch stets die Übergänge zwischen verschiedenen Methoden, da sie ohne Unterstützung etabliert und überprüft werden müssen.

Wir schlagen die visuelle Spezifikationstechnik OBSM (Objektorientierte Benutzungsschnittstellen-Modellierung) vor, die wenige, kompakt notierbare Sprachmittel anbietet, mit denen die angesprochenen Übergänge fließend mit einer durchgängigen Notation behandelt werden können. Sie erlaubt es, Beziehungen herzustellen zwischen

- Benutzeraufgaben (user tasks) und Interaktionsfolgen aus Benutzersicht,
- Interaktionsfolgen und Abarbeitung von Ereignissen,
- Reaktion auf Ereignissen und den Objektstrukturen der Benutzungsschnittstelle sowie
- der Objektwelt der Benutzungsschnittstelle und der Applikation

und diese Beziehungen in einer konsistenten Darstellung zu repräsentieren. Dadurch ermöglicht OBSM die übergreifende Behandlung aller Aspekte in einem gemeinsamen Rahmen, wodurch die gegenseitigen Abhängigkeiten deutlich gemacht und zum expliziten Entwurfsgegenstand werden.

OBSM wurde in einer Diplomarbeit [10] aufbauend auf Ideen der Modellierung objektorientierter Strukturen in Benutzungsschnittstellen [20] entwickelt. Wesentlich beruht es auf dem Konzept der Objektorientierung, angereichert um constraint-basierte Konstrukte, die zur deklarativen Beschreibung von Zusammenhängen verschiedenster Art eingesetzt werden. Zusätzlich werden unter anderem Symbole für Aufgaben, ihre zeitlichen Beziehungen (temporale Relationen) und Interaktion verwendet.

Das Papier behandelt im folgenden Kapitel genauer die oben skizzierten Probleme der Übergänge zwischen verschiedenen Entwurfsaspekten, beschreibt im dritten

Kapitel die Grundzüge der OBSM-Notation und behandelt in einem kurzen Ausblick die Arbeiten an der Werkzeugunterstützung und geplanten Erweiterungen.

2 Entwurfsaspekte von Benutzungsschnittstellen

2.1 Vom Aufgabenmodell zum Dialogmodell

Das in der Aufgabenanalyse entstehende Aufgabenmodell ist eine Beschreibung dessen, wie der Anwender seine Tätigkeiten ausführt bzw. wie die einzelnen Aufgaben gemeinsam von ihm und dem System zu erfüllen sind. Die hierbei zwischen Mensch und Maschine auszutauschenden Informationseinheiten und die Struktur der wechselseitigen Kommunikation wird im Dialogmodell festgehalten.

Beide Modelle werden durch unterschiedliche Methoden unterstützt. Techniken zur Aufgabenmodellierung (wie etwa das GOMS-Modell [2]) sind explizit für diese und nicht im Hinblick auf die Dialogmodellierung entwickelt worden. Vielfach sind die durch eine Benutzeraktion ausgelöste Systemreaktion nicht darstellbar und es wird von fehlerfreien, nicht-unterbrechbaren Abläufen ausgegangen. In der Phase des Dialogentwurfs werden daher zwangsläufig andere Modelle eingesetzt, was einen Wechsel in der Vorgehens- und Darstellungsweise bedingt. Typisch ist hier der Einsatz von Transitionsnetzen [8], [22], ereignisorientierten Spezifikationssprachen [6], Grammatiken [4] oder regelbasierten Notationen [13].

Der Übergang vom Aufgaben- zum Dialogmodell wird methodisch nur geringfügig unterstützt; vielfach beschränken sich die entsprechenden Techniken auf Kriterien, anhand derer, ausgehend vom Aufgabenmodell, erste Hinweise für die Gestaltung der Benutzungsschnittstelle ableitbar sind (MAD, [19]). Auch beim Aufgaben-Kontext-Modell [12] läßt sich trotz spezieller Ausrichtung auf die Dialogerstellung dessen systemorientierte Spezifikation, die mit Produktionsregeln erfolgt, nicht unmittelbar aus dem Aufgabenmodell ableiten.

Objektorientierten Methoden (OOM) mangelt es allgemein an Konzepten zur aufgaben- und benutzungsorientierten Entwicklung. Diese Gesichtspunkte gehen in einigen Ansätzen (OOSE [9], OBA [18]) nur indirekt über die Modellierung von Szenarien mit ein, in denen die Verwendung der einzelnen Objekte dargestellt wird. Durch das Prinzip der Objektorientierung in Analyse und Design wird zwar der Übergang zwischen diesen Phasen vereinfacht, jedoch müssen OOM für eine benutzungsorientierte Entwicklung durch entsprechende Techniken ergänzt werden.

2.2 Dialogmodellierung: Benutzer- und Systemsicht

Da ursprünglich die Benutzungsschnittstellen von den Entwicklern der Anwendungs-Software mit entworfen wurden, sind die etablierten Techniken zur Dialogmodellierung systemorientiert, d.h. sie beschreiben die Interaktionen aus Systemsicht. Zu den klassischen Modellen zählen hier die schon erwähnten kontextfreien Grammatiken, Zustands-Übergangs-Diagramme sowie ereignisbasierte Systeme. Diese Techniken unterstützen das Design und die Implementierung der Benutzungsschnittstellen-Software, nicht aber die adäquate Modellierung von Interaktionsfolgen des Benutzers mit der Schnittstelle, dessen Sichtweise durch die Durchführung einer Aufgabe geprägt ist.

Demgegenüber ist die UAN-Notation [7] eigens für ein aufgaben- und benutzerorientiertes Design entwickelt worden. Eine UAN-Spezifikation enthält jedoch keine Aussagen darüber, welche Komponenten der Software für die Verarbeitung der einzelnen Interaktionen verantwortlich sind. Sie muß durch ein vom Entwickler durchzuführendes systemorientiertes Design ergänzt werden, da dieses nicht direkt ableitbar ist.

2.3 Dynamische und statische Sicht

Bei der Entwicklung der Software von Benutzungsschnittstellen sind neben der Beschreibung des Verhaltens (Dynamik) die Datenstrukturen (statische Sicht) zu definieren, die durch die Interaktionen erzeugt, zerstört bzw. modifiziert werden. Es wird allgemein als ein schwieriges Problem angesehen, dynamische und statische Aspekte in einer einzigen konsistenten Darstellung (z.B. einem Diagramm) unter der Vorgabe zu repräsentieren, daß beide Sichtweisen eindeutig und klar daraus hervorgehen und korrekt in Beziehung gesetzt sind*.

Für die Darstellung der statischen bzw. dynamischen Aspekte werden oft unterschiedliche Notationen benutzt. Zur Beschreibung der Dynamik werden etwa in objektorientierten Modellen systemorientierte Verhaltensbeschreibungen eingesetzt, die in das eigentliche Objektverhalten zu übersetzen sind ([1], [17], [21]). Der Einsatz der Diagramme als Kommunikationsmittel ist erschwert, da die

---

*) Beispielsweise wurden in SADT (Structured Analysis and Design Technique, [16]) getrennte Modelle zur Beschreibung von Aktivitäten (dynamisches Modell) und Daten (statisches Modell) aufgestellt. Es ist weder den Erfindern noch anderen gelungen, klar zu definieren, wann die beiden SADT-Modelle zu einem Problem konsistent zu nennen sind; letztlich wurde durch die für die Zeit typische Dominanz der funktionalen Sicht das Datenmodell meist weggelassen.

Zusammenhänge zwischen Ereignissen und Datenänderungen oft Diagrammen entnommen werden müssen, die sich keiner einheitlichen Darstellungsart bedienen und daher ineinander transformiert werden müssen. Dieser Übersetzungsprozeß ist von den Benutzern der Notationen zu leisten.

Ziel einer Notation sollte es jedoch sein, den kognitiv-mentalen Aufwand des Benutzers einer Beschreibungstechnik möglichst gering zu halten, damit dieser sich auf die eigentlichen Informationen konzentrieren kann. In diesem Sinne kann auch von der Forderung nach "Direktheit" für Notationen gesprochen werden.

2.4 Objektmodell der Anwendung und der Oberfläche

Graphische Benutzungsschnittstellen (*graphical user interfaces, GUIs*) werden technisch als Hierarchie graphischer Objekte implementiert, die von sog. Toolkits wie OSF/Motif [14], XView [5] und InterViews [11] angeboten werden. Die Applikation (AP) einer Anwendungs-Software weist in der Regel eine Struktur auf, die sich wesentlich von der Struktur des GUI unterscheidet. Die Kopplung beider Komponenten ist somit durch eine i.a. nicht-triviale Abbildung charakterisiert. Vielfach sind nicht nur die Daten oder Objekte der AP selbst, sondern auch deren Beziehungen zueinander durch Objekte des GUI zu repräsentieren. Beispielsweise können die Vorgänger- bzw. Nachfolgerbeziehungen in einer Baumstruktur innerhalb der AP mit Zeigern realisiert und in der Oberfläche als Pfeile dargestellt werden.

3 Modellierung mit der OBSM-Notation

Mit der OBSM-Notation werden die im vorangegangenen Kapitel erläuterten Aspekte einschließlich der Übergänge zwischen diesen unterstützt. OBSM ist eine graphische Notation, die für den praktischen Einsatz werkzeugunterstützt sein muß. Dies erlaubt dem Notations-Benutzer, den Informationsgehalt der Darstellungen durch "Ein- und Ausblenden" selbst zu bestimmen, d.h. er kann sich gezielt Diagrammkomponenten anzeigen lassen. In diesem Sinne werden in den folgenden Beispieldiagrammen immer nur die in einem bestimmten Kontext relevanten Informationen dargestellt.

3.1 Aufgabenmodellierung

Die graphische Darstellung eines Aufgabenmodells umfaßt im allgemeinen die Aufgaben, deren hierarchische Struktur sowie die Reihenfolge ihrer Bearbeitung.

Die OBSM-Notation kann zur Visualisierung dieser Gesichtspunkte eingesetzt werden und bietet darüber hinaus Sprachelemente, mittels derer die zeitlichen Beziehungen der Ausführungssequenzen sowie Ausnahme- und Fehlersituationen ausgedrückt werden können. Daneben besteht die Möglichkeit, Systemaufgaben zu integrieren sowie zwischen benutzer- und systembedingten Verzweigungen zu unterscheiden. Bezüge zum Objektmodell erlauben es, bereits auf dieser Ebene im Aufgabenkontext relevante Objekte einzubinden. Durch die Vielschichtigkeit ihrer Ausdrucksmöglichkeiten, von denen einige im folgenden anhand eines Beispiels vorgestellt werden sollen, kann die OBSM-Notation als Darstellungstechnik für vorhandene Methoden der Aufgabenanalyse eingesetzt werden.

Als Beispiel soll die Verwaltung einer Telefonliste dienen. Abbildung 2 zeigt die Aufgabe Telefonliste verwalten und ihre Verfeinerung in Eintrag verwalten mit den Unteraufgaben Eintrag lesen, Eintrag ändern sowie Eintrag löschen. Die hierarchische Aufgabengliederung wird durch Ineinanderschachtelung der entsprechenden Rechtecksymbole repräsentiert. Diese Darstellungsweise ist ähnlich der für Klassen- und Objekten, bei denen ebenfalls verfeinernde Komponenten im Klassen- bzw. Objektrechteck enthalten sind und die Spezifikationsgenauigkeit mit der Schachtelungtiefe zunimmt (s. Abschnitt 3.2).

Die zwischen den einzelnen Aufgaben bestehenden zeitlichen Beziehungen werden mittels temporaler Relationen modelliert. In der Telefonliste sollen beliebig oft einzelne Einträge verwaltet werden können (Wiederholungspfeil). Die Aufgabe Eintrag verwalten selbst besteht aus den sich zeitlich ausschließenden Unteraufgaben zum Lesen, Ändern oder Löschen eines Eintrags (Auswahlsymbol). Der Benutzer soll also genau eine dieser Teilaufgabe bearbeiten und sich anschließend für die Aufgabe Eintrag verwalten erneut entscheiden können. Die Unteraufgabe Eintrag löschen ist hier im Gegensatz zu den anderen nicht unterbrechbar (durchgezogene Rechtecklinie des Aufgabensymbols).

Generell werden mit unterbrochenen Linien die Stellen markiert, an denen andere Aufgaben eingeschoben werden dürfen. Zur Bestimmung derjenigen Aufgaben, die dann als Unterbrechung auftreten dürfen, werden neben den Unterbrechungssymbolen die Positionen in der Aufgabenhierarchie ausgewertet [10]. Auf diesen Beschreibungsmechanismus selbst soll hier nicht eingegangen werden.

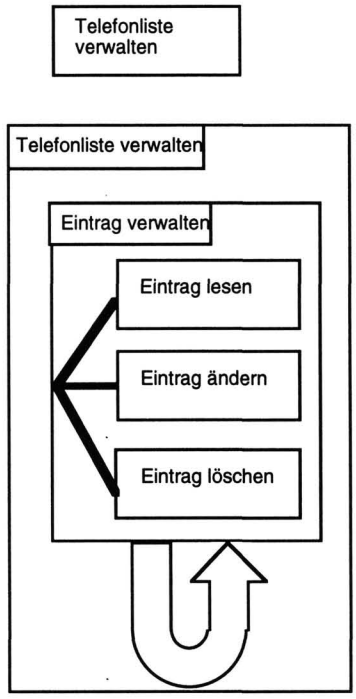

Abb. 2: Aufgabenmodell

Objektintegration

Aufgaben stehen in engem Zusammenhang mit den Objekten, auf die sie sich beziehen. Insbesondere beziehen sich die Aufgaben, die in Kooperation mit einer Anwendung erledigt werden, auf Objekte des Softwaresystems. Daher ist es naheliegend, diesen Zusammenhang auch im Modell explizit darzustellen. Dabei bleibt es dem Anwender der OBSM-Notation überlassen, bis zu welchem Grad er seinem Aufgabenmodell Objekte hinzufügen möchte - dies erlaubt einen fließenden Übergang einer benutzungsorientierten zu einer systemorientierten Sicht.

Abbildung 3 zeigt die Verfeinerung der Aufgabe Eintrag löschen. In ihrer ersten Unteraufgabe soll zunächst ein Eintrag ausgewählt und anschließend entfernt werden* . In der Anwendung ist ein Objekt notwendig, das diese Funktionalität durch eine geeignete Methode unterstützt. Derartige Zusammenhänge, d.h. die Bearbeitung einer Aufgabe und der damit verbundene Aufruf einer Software-Prozedur, werden durch einen "Blitz" notiert. Dieser dient als generelles Gestaltungsmittel zur visuellen Hervorhebung der Benutzeraktivitäten, die quasi von außen auf das Softwaresystems "treffen". Dementsprechend zeigt im Diagramm ein Blitz von der Aufgabe entfernen auf die durch das Dreieck

*) Zur Festlegung dieser zeitlichen Beziehung zeigt ein Sequenzpfeil von der ersten zur nachgelagerten Aufgabe.

symbolisiert Methode löschen des Anwendungsobjektes Telefonliste und stellt so deren Aufruf durch eine Benutzeraktion dar. Die Parameter, die hierbei übergeben werden (im Beispiel Name), legen fest, welche Eingaben vom Benutzer erwartet werden und somit Bestandteil seiner Aufgaben sind.

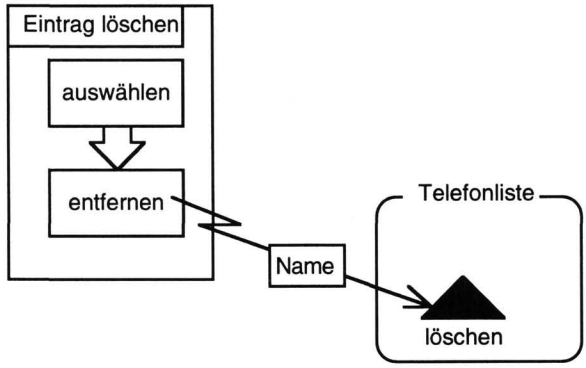

Abb. 3: Objektanbindung

Mit fortlaufender Verfeinerung werden zunehmend Objekte integriert. Es kristallisieren sich auf diese Weise diejenigen heraus, die von Benutzer und Anwendung gemeinsam benutzt werden und es entsteht parallel zum Aufgabenmodell der sogenannte Objektkern ([20]). Dieser enthält die in der Oberfläche sichtbar zu machenden Objekte sowie die Methoden, die dem Benutzer zugänglich sein müssen. Er bildet die Basis für die anschließende Software-Entwicklung, in der die Benutzungsschnittstelle und die Applikation weitgehend unabhängig voneinander erstellt werden können.

Statische und dynamische Sicht

Implizit wird in der obigen Darstellung auf zwei (konzeptionellen) Zeichenflächen agiert: auf einer "statischen" und einer "dynamischen" Zeichenfläche* . Bei der Modellierung der Dynamik steht die Fläche der Symbole für die zeitliche Dimension, d.h. für die Zeit, die eine Aufgabe benötigt oder die durch eine temporale Relation beschrieben wird. So legt der Pfeil für die Sequenz nicht nur die Reihenfolge der Vorgänge fest, sondern repräsentiert auch die Zeit zwischen den Vorgängen. Das Symbol nimmt Platz ein, "verbraucht" also Zeit. Auf der anderen Zeichenfläche steht der Raum, den ein Objekt belegt, für den Speicherplatz, den es zur Laufzeit einnimmt. Die flächige Darstellung von Klassen und Objekten erlaubt es, die jeweils zugehörigen Attribute, Methoden und Constraints graphisch innerhalb der Symbole darzustellen (s. Abschnitt 3.2).

---

*) Zur übersichtlicheren Darstellung werden hier in den Diagrammen das Modell der Dynamik auf der linken und statische Aspekte auf der rechten Seite gezeichnet.

3.2 Anwendungs- und Oberflächenstruktur

3.2.1 Objekte des Kerns und der Oberfläche

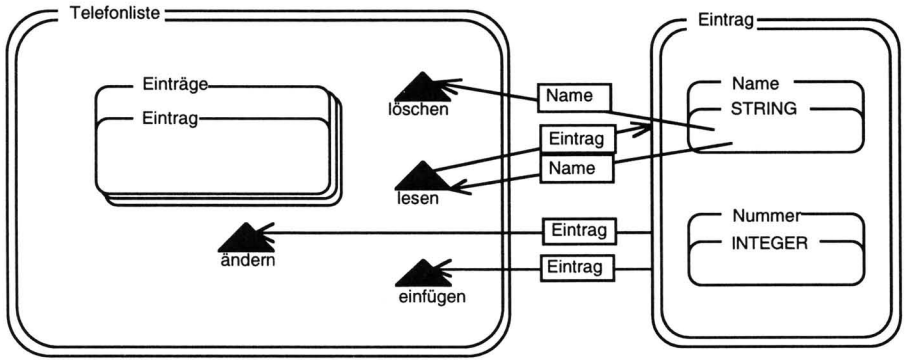

Abb. 4: Klassen des Objektkerns

Den Objektkern für das obige Beispiel zeigt Abbildung 4. Im Diagramm der Klasse Telefonliste - zur Unterscheidung von Objekten wird das Klassensymbol mit einer Doppellinie gezeichnet - werden die einzelnen Listeneinträge in der Objektmenge Einträge zusammengefaßt. Ihre Elemente sind Objekte der Klasse Eintrag und besitzen alle ein Attribut Name vom Typ STRING und ein INTEGER-Attribut Nummer. Zum Zugriff auf Einträge stehen die Methoden lesen, einfügen, ändern und löschen zur Verfügung. Die zugehörigen Parameter (Eintrag, Name) werden durch Pfeile mit den Methodendreiecken verbunden. Hierbei legt die Pfeilrichtung den jeweiligen Inhalt als Ein- oder Ausgabewert fest, die Verlängerung des Pfeiles zum Klassensymbol Eintrag bzw. Name definiert hingegen den Parametertyp.

Abb. 5: BO-Darstellung

Mit der Bestimmung der Kernobjekte ist noch nicht festgelegt, wie diese dem Benutzer dargestellt werden sollen. Dies ist Gegenstand der Entwicklung der Benutzungsschnittstelle, d.h. mit ihrer Gestaltung sind u. a. die Objekte zur graphischen Repräsentation der Kernobjekte in der Oberfläche festzulegen (BO-Objekte). Im Beispiel soll jeder Eintrag der Telefonliste gemäß Abbildung 5 angezeigt werden. Die notwendigen Vereinbarungen enthält die Klasse BO-Eintrag (Abbildung 6).

Die Klasse enthält Objekte, die den am Bildschirm dargestellten Text (Name und Nummer) und das Telefonsymbol (Telefon) repräsentieren. TEXT ist ein graphisches Element, das - im Gegensatz zum STRING - Textattribute zur

graphischen Darstellung festlegt, also zum Beispiel den Font, die Größe und die *Bounding Box* (BBox). ICON ist ebenfalls vordefiniert und repräsentiert Bitmuster. Das Gleichheitsconstraint ⟨=⟩ bestimmt, daß beide Texte übereinander positioniert werden sollen; das LINKS-VON-Constraint definiert, daß das Telefonsymbol links von beiden Texten stehen soll. Die Constraints werden in dieser und den folgenden Abbildungen nur in abstrakter Form verwendet. Analog zur Vorgehensweise bei der Beschreibung von Aufgaben und Klassen bzw. Objekten werden sie innerhalb ihres Symbols spezifiziert. Hierbei werden sie mit Durchgriff auf die Elementarkomponenten der verknüpften Objekte durch Verbindung einfacher Constraints hierarchisch aufgebaut. Letztlich werden atomare Constraints durch mathematische Beziehungen zwischen Zahlen, z.B. x- und y-Koordinaten, festgelegt. Auf diesen Mechanismus soll hier ebenfalls nicht näher eingegangen werden.

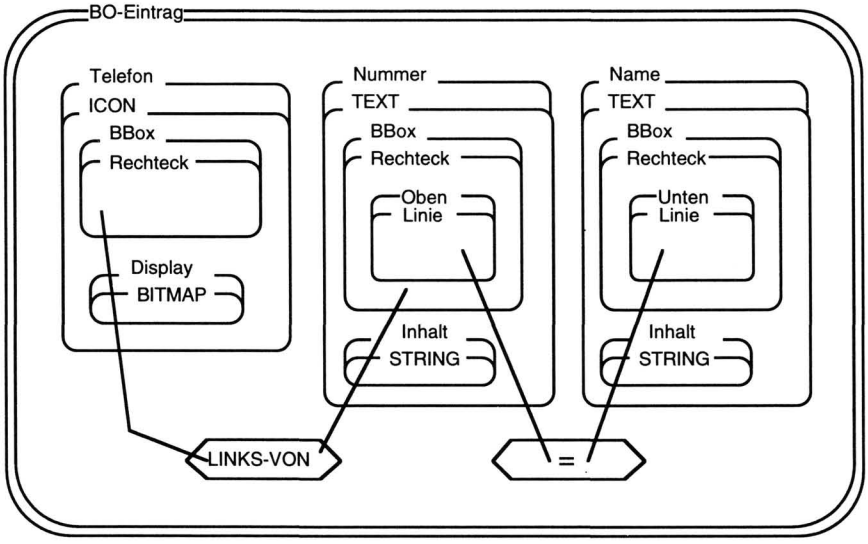

Abb. 6: Objekt zur graphischen Repräsentation des Kernobjektes Eintrag

3.2.2 *Kopplung der Kern- und Oberflächenobjekte*

Objekte des Kerns und der Oberfläche können auf vielfältige Weise miteinander gekoppelt sein. In unserem Beispiel soll zur Laufzeit der Anwendung für jedes Kernobjekt Eintrag ein Oberflächenobjekt BO-Eintrag existieren. Diese Forderung wird durch ein sog. Existenz-Constraint (in Abbildung 7 durch EX gekennzeichnet) realisiert.

Der Objektrand wird in diesem Zusammenhang als boolescher Wahrheitswert interpretiert (true: Objekt existiert, false: Objekt existiert nicht), was durch die andersartige Pfeilspitze zusätzlich verdeutlicht wird. Das Instanziieren und Löschen der Einträge im Kern und in der BS erfolgt somit in wechselseitiger Abhängigkeit (Instanzkopplung), wobei dann jeweils Name und Nummer in beiden Objektstrukturen übereinstimmen müssen.

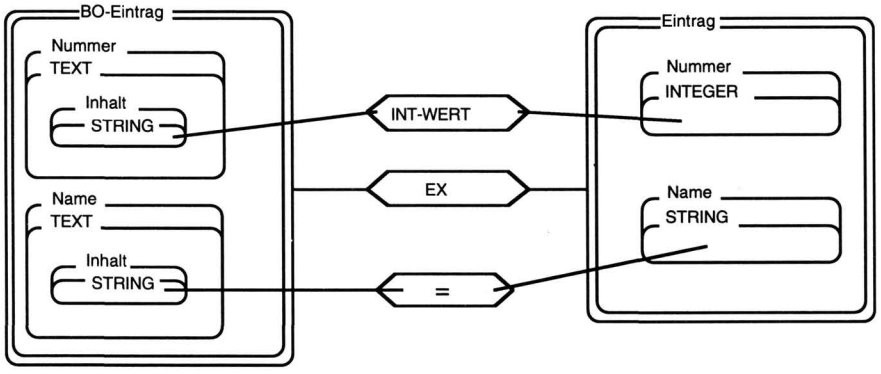

Abb. 7: Instanzkopplung: Eintrag und BO-Eintrag

In Abbildung 7 sind diese Bedingungen für jeweils ein beliebiges Objekt aus Eintrag und BO-Eintrag spezifiziert. Da dies für alle Objekte der Klassen gelten soll, sind die entsprechenden Vereinbarungen auf Klassenebene zu treffen. Hierbei sind die Abhängigkeiten der Gleichheit-Constraints von der Instanzkopplung zu berücksichtigen, denn zur Laufzeit sollen sich immer nur die Werte der gekoppelten Objekte entsprechen.

Insgesamt entsteht eine stark verflechtete Struktur aus Kern- und Schnittstellenobjekten, die über Constraints miteinander in Beziehung gesetzt werden.

3.3 Interaktionssequenzen

Bisher wurden das Aufgabenmodell und der Objektkern mit dessen Repräsentationen in der Oberfläche festgelegt. Die Querbezüge zwischen den Aufgaben und den Kernobjekten markieren hierbei die Stellen, an denen das Softwaresystem die einzelnen Tätigkeiten des Benutzers unterstützen soll. Es werden so die Aufgabenkontexte bestimmt, in denen Kernmethoden aufrufbar sein sollen; offen ist hingegen, wie der Benutzer deren Aktivierung veranlassen kann.

Dies ist Gegenstand der Benutzerinteraktionsfolgen, in denen das Interaktionsverhalten zwischen Benutzer und System modelliert wird.

Das folgende Diagramm (Abbildung 8) zeigt ein Beispiel für die Interaktion innerhalb der Aufgabe auswählen (vgl. Abbildung 3): Der Benutzer soll durch das Drücken einer Maustaste (MDown) über dem zugehörigen Oberflächenobjekt (BO-Eintrag) einen Eintrag selektieren können.

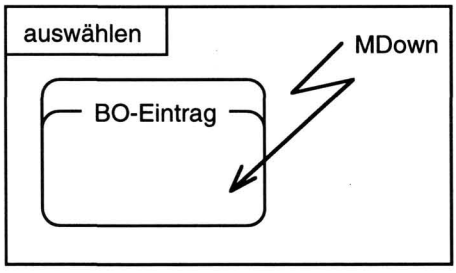

Abb. 8: Verfeinerung der Aufgabe auswählen

Der Blitz symbolisiert in den Interaktionssequenzen Benutzeraktionen, die außerhalb des Softwaresystems ausgelöste Ereignisse darstellen und immer einem Objekt zugeordnet werden. Dieser Darstellungsweise liegt die Idee zugrunde, daß jedes (gültige) Ereignis von einem oder mehreren Objekten verarbeitet wird. Gerade in graphischen Oberflächen bezieht sich eine Benutzeraktion fast ausschließlich auf ein (auf dem Bildschirm sichtbares) Objekt. So können bei der Modellierung des Benutzerverhaltens in vielen Fällen die Aktionen zusammen mit dem jeweils betroffenen BO-Objekt dargestellt werden. Ist dies nicht möglich, wie bei manchen Tastaturkommandos, so ist auf jeden Fall ein Objekt in der Benutzungsschnittstelle, das nicht BO-Objekt ist, für das Ereignis zuständig. Mit zunehmendem Detaillierungsgrad des Modells wird festgehalten, welche Objekte die verschiedenen Ereignisse verarbeiten.

Abbildung 9 zeigt eine mögliche Verfeinerung der kompletten Aufgabe Eintrag löschen: Nach der Auswahl des BO-Repräsentanten eines Telefoneintrags soll durch dessen Verschieben in einen Papierkorb die Löschmethode aktiviert werden. Dies wird durch ein wiederholtes MDMove (Mausbewegung mit gedrückter Taste) "auf" BO-Eintrag modelliert. Die Übereinstimmung zwischen dem zuvor ausgewählten und jetzt zu verschiebenden Eintrags wird durch den Parameter E an der Stelle des Objektnamens hergestellt. Für die Dauer dieser Aktionen wird das Constraint AN aktiviert, wodurch das Piktogramm den Mausbewegungen folgt. Da der bisherige Mauszeiger nun nicht mehr angezeigt werden soll, wird seinem Sichtbarkeitsattribut der Wert *false* zugewiesen. Wird unter diesen Vorbedingungen die Maustaste über dem Papierkorb losgelassen, soll der Eintrag aus der Objektmenge Einträge von Telefonliste entfernt werden.

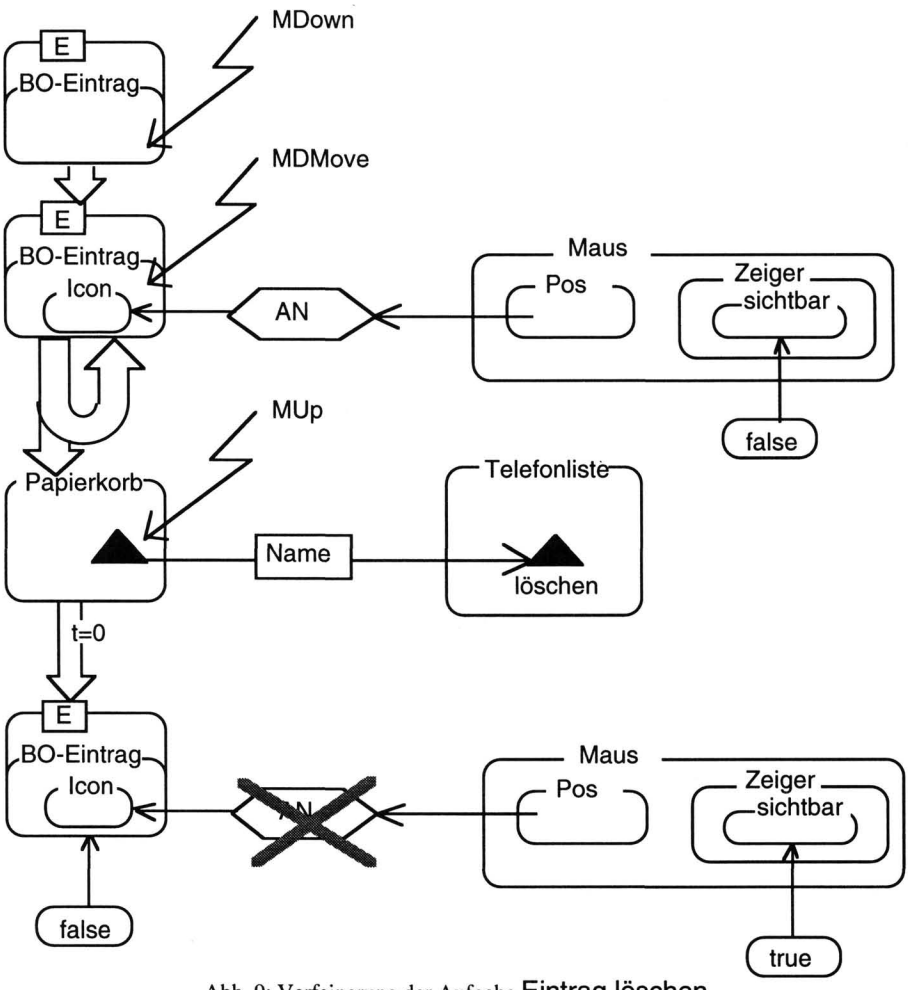

Abb. 9: Verfeinerung der Aufgabe **Eintrag löschen**

An dieser Stelle zeigt sich exemplarisch der Übergang von der abstrakten Darstellung eines Methodenaufrufs innerhalb einer Aufgabe (Abbildung 3) zur Spezifikation eines programmtechnischen Ereignisses. Die Aktivierung von **löschen** folgt also nicht direkt aus einem Benutzerereignis, sondern durch ein in einer anderen Methode generiertes (System-) Ereignis. Zur klaren Unterscheidung der Systemereignisse von den Benutzeraktionen werden erstere mit einem einfachen Pfeil dargestellt.

Direkt anschließend (Zusatzspezifikation t=0 am Sequenzpfeil) soll das Oberflächenobjekt entfernt werden. Zur Modellierung wird dem hier ebenfalls als Wahrheitswert interpretierten Objektrand der Wert *false* zugewiesen. Die Instanzkopplung bewirkt an dieser Stelle auch das Löschen des zugehörigen Kernobjektes. Die Aufgabe löschen ist mit dem Deaktivieren des Constraints und dem Sichtbarmachen des ursprünglichen Mauszeigers beendet.

Die Modellierung der Aufgaben und somit auch die der Interaktionsfolgen befindet sich immer auf der dynamischen Zeichenfläche, während bezüglich der angebundenen Objekte die statische und die dynamische Sicht möglich sind. Werden sie zur dynamischen Zeichenfläche gezählt, so wird neben deren aktueller Verwendung gezeigt, in welchem Zustand sie sich befinden müssen, d.h. auf welche Werte innerhalb eines Kontextes einzelne Attribute gesetzt sein müssen. Dabei werden jeweils nur die relevanten Attribute gezeichnet. Die in einer Situation nicht dargestellten Attribute dürfen beliebige Werte haben.

Mit der Interpretation als statische Zeichenfläche werden die Auswirkungen der Benutzeraktionen auf das Objektmodell betrachtet. Mit dieser Sichtweise stellen die Querbezüge zwischen Aufgabenmodell und Objektkern gleichzeitig die Wechselwirkungen zwischen dynamischer und statischer Zeichenfläche her.

In den Interaktionsfolgen bedingt der zunehmende Einsatz der Objektsymbole auf der dynamischen Zeichenfläche eine noch stärkere Vermischung mit der statischen Sicht. Auf beiden Zeichenflächen werden für Objekte, Ereignisse, Methoden und Constraints jeweils die selben Symbole verwendet. Hierdurch soll der Übergang zwischen beiden Sichtweisen erleichtert werden, da die Darstellungen nicht zwischen den verschiedenen Zeichenflächen übersetzt werden müssen.

3.4 Aspekte des systemorientierten Modells

Für eine vollständige Modellierung müssen mit den Benutzerinteraktionsfolgen alle möglichen Zweige der Benutzungsschnittstelle erfaßt werden. Aus den verschiedenen Interaktionssequenzen können wesentliche Aussagen zum Objektmodell abgeleitet werden: So legen die Spezifikationen zu einer Objektart an verschiedenen Stellen dar, welche Komponenten die Klasse enthalten und welche Ereignisse sie verarbeiten muß, wobei letzteres Hinweise auf die in ihr zu implementierenden Methoden gibt. Die Gesamtheit aller Benutzerinteraktionsfolgen beschreibt das Verhalten der Benutzungsschnittstelle. Für eine Überprüfung des Modells und insbesondere für eine Simulation der verschiedenen Abläufe ist es erforderlich, nach jedem Dialogschritt den Standpunkt des Benutzers (Zustand des Dialogs) und die möglichen Folgeaktionen vollständig darzustellen.

Für diesen Aspekt bieten Zustands-Überführungs-Diagramme eine vorteilhafte Darstellung. Für jeden Zustand zeigt der ihn repräsentierende Knoten anhand der ausgehenden Kanten alle in ihm möglichen Folgeaktionen, ankommende Kanten stellen die Aktionen dar, mit denen der Zustand erreicht wurde. Hiermit verglichen bilden die verschiedenen Benutzerinteraktionsfolgen einzelne Pfade durch einen derartige Graphen und zeigen meist nur eine Teilmenge der Alternativen.

Zur Gewinnung der Zustandsinformationen aus den Benutzerinteraktionsfolgen wurde ein System dynamischer Kellern entwickelt, auf dessen Grundlage Zustands-Überführungs-Diagramme abgeleitet werden können [10]. Das Kellerverhalten ist direkt aus den Interaktionsfolgen generierbar. Eine wesentliche Aufgabe ist hierbei die Umsetzung der temporalen Relationen, da sie zusammen mit der hierarchischen Aufgabengliederung ausschlaggebend für die zulässigen Folgeaktionen sind.

Jede mögliche Aufgabe bzw. Aktion wird durch ein Kellersymbol repräsentiert. Die Gesamtheit der jeweils obersten Elemente gibt die aktuellen Handlungsalternativen des Benutzers wieder. Abbildung 10 zeigt dies für sechs aufeinanderfolgende Kellerzustände der Telefonlistenverwaltung und eines Hilfesystem (in der Abbildung mit H gekennzeichnet).

Die Entscheidung des Benutzers für die Aufgabe Telefonliste verwalten führt zur Ersetzung des zugehörigen Zeichens gemäß ihrer Unterstruktur. Die unterste Kellerzeile steht hier für die Möglichkeit, die Aufgabe erneut auszuführen, wobei der Leerkeller die Alternative keiner erneuten Wiederholung symbolisiert.

Die Verwaltung eines Eintrags besteht aus dem Lesen, Ändern oder Löschen. Die Auswahl einer dieser Aufgaben, im Beispiel Eintrag löschen, führt zu einer erneuten Ersetzung und zum Entfernen der Lese- und Ändern-Kellerzeichen, da die zugehörigen Aufgaben nicht mehr als Alternativen zur Verfügung stehen. Ebenso kann der Benutzer an dieser Stelle das Hilfesystem nicht mehr aufrufen, da Eintrag löschen in Abbildung 2 als nicht-unterbrechbar vereinbart wurde. Der Keller des Hilfesystems wird daher durch den Querstrich als oberstes Symbol als gesperrt gekennzeichnet. Dies gilt solange, bis die als nicht unterbrechbar gekennzeichnete Aufgabe beendet ist und damit der Keller wieder "geöffnet" wird.

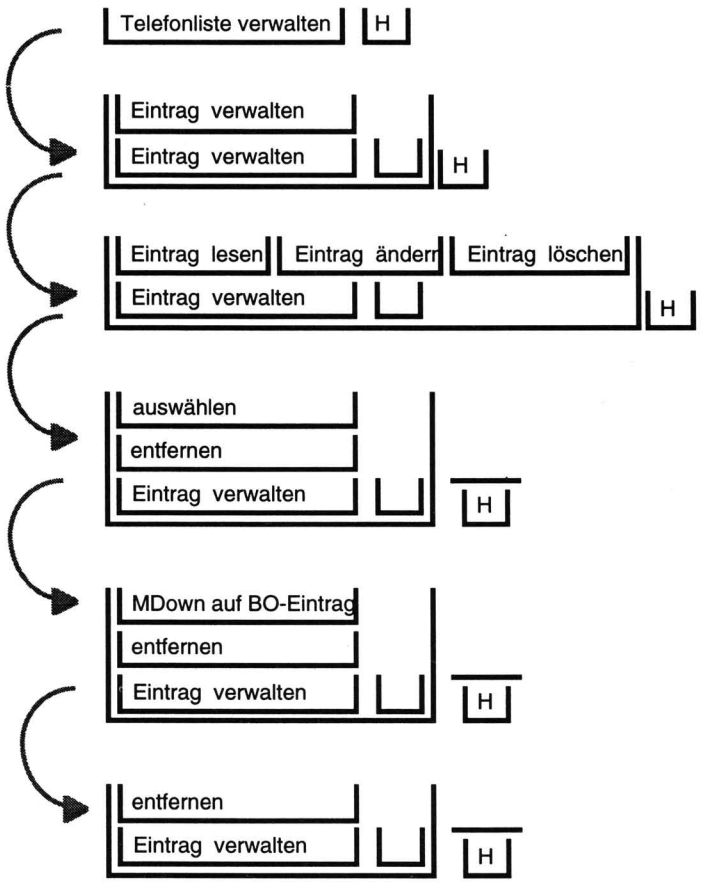

Abb. 10: Beispiel eines Kellersystems

Mit der Bearbeitung der Aufgabe **auswählen** bzw. der Aktion **MDown** wird der Keller, da er keine weitere Unteraufgabe enthält, wieder entfernt und als nächstes kann die Löschoperation durchgeführt werden.

Diese am Beispiel gezeigte Darstellung läßt sich aus beliebigen Interaktionssequenzen erzeugen, womit eine OBSM-Spezifikation simulierbar wird.

3.5 Zusammenfassung

Anhand eines Beispiels wurde demonstriert, wie die OBSM-Notation die verschiedenen Übergänge zwischen den Entwurfsaspekten (vgl. Kapitel 2) ermöglicht:

(1) Die Benutzerinteraktionsfolgen ergeben sich hier als Verfeinerung des Aufgabenmodells und ermöglichen so eine weitgehend kontinuierliche Modellierung von globalen Aufgaben des Benutzers bis zur gewünschten Detaillierung, ggf. sogar bis hin zu den einzelnen Tastendrücken. Das Ergebnis der Aufgabenanalyse muß somit nicht für die Dialogbeschreibung in eine andere Darstellungsform transformiert werden. Dadurch ist es möglich, innerhalb einer Notation die Konsistenz zwischen den modellierten Aufgaben und den einzelnen Dialogschritten sicherzustellen und zu dokumentieren. Darüberhinaus kann mit dieser Vorgehensweise eine leichte Änderbarkeit erzielt werde.

(2) Bei der Modellierung der Aufgaben und Interaktionsfolgen werden die im jeweiligen Kontext relevanten Objekte miteinbezogen, d.h. die einzelnen Aktionen können bereits im benutzungsorientierten Design den verantwortlichen Komponenten zugeordnet werden. Diese Vorgehensweise und das formale System zur Gewinnung der Zustandsinformationen erlauben es, wesentliche Aspekte der systemorientierten Beschreibung zu folgern, bzw. im Zusammenhang mit der benutzungsorientierten Sicht zu entwerfen. Daneben sind aus den vorhandenen Informationen andere "klassische" Darstellungsweisen (z.B. Zustands-Überführungs-Diagrammme) ableitbar.

(3) Die OBSM-Notation versucht, innerhalb einer Darstellung durch Unterscheidung zweier "virtueller Zeichenflächen" eine Brücke zwischen dynamischer und statischer Sicht zu bauen. Für die darstellungsrelevanten Komponenten werden auf den sich einander ergänzenden Zeichenflächen jeweils gleiche Symbole mit gleicher Semantik eingesetzt, die in ihrer Gesamtheit ein logisches Diagramm der Benutzungsschnittstelle bilden. Die hierbei bewußt in Kauf genommene Vermischung der ohnehin nicht voneinander unabhängigen statischen und dynamischen Aspekte erlaubt es dem Notations-Benutzer, dieselben Diagramme aus verschiedenen Blickwinkeln zu betrachten. Zusätzlich bewirken die Querbezüge zwischen Aufgabenmodell und Objektkern sowie zwischen Benutzerinteraktionsfolgen und Objekten eine enge Kopplung dieser Perspektiven. Aufgaben und Ereignisse stehen also wie in der realen Welt in direktem Zusammenhang mit den beteiligten Objekten, so daß man von einer frühen, bereits auf Aufgabenebene erfolgenden, Objektanbindung sprechen kann. Gleichzeitig

wird durch die graphischen Bezüge dokumentiert, für welche Aufgaben die jeweiligen Komponenten des Objektkerns entwickelt wurden.

(4) Das Objektmodell der Anwendung und der Oberfläche ergeben sich durch sukzessive Integration der Objekt bei der Aufgaben- und Interaktionsmodellierung wobei die Kopplung zwischen GUI und AP mittels Constraints definiert wird. Einen speziellen Einsatz von Constraints bildet die Instanzkopplung, die es erlaubt, mittels Constraints auch die Existenz eines Objektes von einem Datum oder der Existenz eines anderen Objektes abhängig zu machen.

Insgesamt hat der Benutzer von OBSM einen großen Freiraum bei der Modellierung, da die Notation von sich aus nahezu keine Abgrenzungen vornimmt. Prinzipiell stehen ihm in allen Arbeitsphasen sämtliche Darstellungstechniken zur Verfügung. So können Informationen zu den Zeitpunkten des dynamischen Entwicklungsablaufs festgehalten werden, an denen sie anfallen und relevant sind, und müssen nicht aufgrund einer Trennung von Phasen oder Modellen in ihrer Erfassung verlagert werden.

4 Ausblick

Für den praktischen Einsatz der OBSM-Notation ist eine Werkzeuglandschaft zur Unterstützung der Modellierungsprinzipien notwendig. Eine wesentliche Aufgabe ist die Bereitstellung eines graphischen Editors, zu dem bereits ein Prototyp existiert. In einer Folgeversion soll es u. a. möglich sein, reale Oberflächenobjekte, d.h. deren graphischen Darstellungen, in die Modellierung miteinzubeziehen.

Weitgehende Hilfe kann zudem bei der Hervorhebung und Generierung verschiedener Sichten geleistet werden. Bei der Beschreibung einer BS entsteht konzeptionell ein großes Diagramm, dessen Informationen je nach Betrachtungsstandpunkt mehr oder weniger relevant sind. Dem Werkzeug-Benutzer müssen geeignete Mechanismen zum Ein- und Ausblenden verschiedener Modellkomponenten geboten werden, so daß er die Bildschirmausgabe weitgehend seinem eigenen Informationsbedürfnis anpassen kann. Hierbei können Abstraktionen wie z.B. Piktogramme und vereinfachte Darstellungen diese Vorgehensweise wesentlich unterstützen.

Neben der Bereitstellung verschiedener Sichten auf das Gesamtdiagramm können Simulationen einen wesentlichen Beitrag zum korrekten Verständnis des Gesamtsystems und der dynamischen Abläufe leisten. Für das zu entwickelnde

Werkzeug bietet sich eine visuelle Simu-lation an, die auf der OBSM-Notation basiert und so die Abläufe anhand der erstellten Diagramme zeigt. Für Beschreibung und Simulation existiert dann ein einziges Modell, wodurch kein Mehraufwand für die Erstellung eines separaten Simulationsmodells zu leisten ist. Zudem wird auf diese Weise erneut ein mentaler Übersetzungsaufwand bei den Benutzern vermieden, da die Diagramme selbst animiert werden.

Die Simulation kann eine bereits während der Modellierung stattfindende Verifikation des zukünftigen Systemverhaltens bieten und zur zusätzlichen Konsistenzprüfung und Absicherung der vorliegenden Ergebnisse führen, so daß sich ein iteratives Vorgehen beim Entwurf bis hin zum Prototypen ergibt.

Daneben soll eine möglichst automatisiert Generierung des systemorientierten Modells aus der benutzerorientierten Darstellung bis hin zum Programmgenerator realisiert werden. Denkbar ist hier, die Komponenten der statischen und der systemorientierten Sicht möglichst weitgehend aus den Benutzerinteraktionsfolgen abzuleiten.

5 Literatur

[1] Booch, G.: *Object Oriented Design with Applications,* Benjamin/Cummings, Redwood City, Ca, 1991

[2] Card, S.; Moran, T.; Newell, A.: *The Psychology of Human-Computer Interaction,* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1983

[3] Dance, J.; Granor, T.; Hill, R.; Hudson, S.; Meads, J.; Myers, B.; Schulert, A.: *The Run-time Structure of UIMS-Supported Applications,* Computer Graphics, Vol. 21, No. 2, 1987

[4] Green, M.: *A Survey of Three Dialogue Models,* ACM Transaction on Graphics, July 1986

[5] Heller, D.: *XView Programming Manual - The Definitive Guide to the X Window System* (Vol. VII), O'Reilly & Associates, Sebastopol (CA), 1989

[6] Hill, R.: *Event-Response Systems - A Technique for Specifying Multi-Threaded Dialogues,* Proceedings of CHI'87, Toronto, April 1987

[7] Hix, D.; Hartson, R.: *Developing User Interfaces: Ensuring Usability through Product & Process,* John Wiley & Sons, New York, 1993

[8] Jacob, R.: *A State Transition Diagram Language for Visual Prgramming,* IEEE Computer, August 1985

[9] Jacobson, I.; Christerson, M.; Jonsson, P.; Övergaard, G.: *Object-Oriented Software Engineering: A Use Case Driven Approach,* Addison-Wesley, 1992

[10] Kneer, B.: *Visuelle Modellierung von Bnutzungsschnittstellen objektorientierter Applikationen,* Diplomarbeit, Universität - GH Paderborn, Fachbereich Mathematik/Informatik, März 1994

[11] Linton, M.; Vlissides, J.; Calder, P.: *Composing User Interfaces with InterViews,* Stanford University, 1990

[12] Monk, A.; Curry, M.; Maidment, B.: *Task-Based Interface Specification,* University of York, Technical Report, 1991

[13] Monk, A.; Curry, M.: *Dialog Specification as a Link between Task Analysis and Implementation,* University of York, Technical Report, 1992

[14] Open Software Foundation:, *OSF/Motif Programmers Guide,* Prentice-Hall, Englewood Cliffs (NJ), 1990

[15] Pfaff, G.: *User Interface Management Systems,* Springer Verlag, 1985

[16] Ross, D.: *Structured Analysis (SA): A Language for Communication Ideas,* IEEE Trans-actions on Software Engineering, SE-3, Januar 1977

[17] Rumbaugh, J.; Blaha, M.; Premerlani, W.; Eddy, F.; Lorensen, W.: *Object-Oriented Modeling and Design,* Prentice Hall, Englewood Cliffs, New Jersey, 1991

[18] Rubin, K.; Goldberg, A.: *Object Behavior Analysis,* Communications of the ACM, 35(9), 1992

[19] Sebillotte, S.: *Task Analysis and Formalisation according to MAD: Hierarchical Task Analysis, Method of Data Gathering and Examples of Task Descriptions,* Technischer Report, Amsterdam, 1992

[20] Szwillus, G.: *Visuelle Spezifikation des Kerns objektorientierter Applikationen,* GI Informatik aktuell: Informatik - Wirtschaft - Gesellschaft, GI-Jahrestagung '93, Springer-Verlag, Berlin, 1993

[21] Walters, N.: *Using Harel Statecharts to Model Object-Oriented Behavior,* ACM SIG-SOFT, Software Engineering Notes, 17(4), 1992

[22] Wasserman, A.: *Extending State Transition Diagram for the Specification of Human-Computer Interaction,* IEEE Transactions on Software Engineering, SE-11, 8, August 1985