Preliminary version:
Final version published by Infix Verlag, 1997.

HAMVIS: Generierung von Visualisierungen in einem Rahmensystem zur systematischen Entwicklung von Benutzungsschnittstellen

Eine dem Fachbereich Informatik
der Universität Hamburg
vorgelegte

Dissertation

zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.)

von

Ralf Möller

Hamburg, 1996

Inhaltsverzeichnis

1 Einleitung

Obwohl inzwischen eine Vielzahl von kommerziellen Unterstützungssystemen für die Erstellung von graphischen Oberflächen zur Verfügung steht (z.B. graphische Werkzeugkästen, engl.: toolkits), gehört die Oberflächenentwicklung zu den schwierigsten und teuersten Teilprojekten bei der Entwicklung von Anwendungsprogrammen. Die Verwendung von standardisierten Schnittstellenkomponenten wie z.B. Meßgeräten (gauges), Schaltern (buttons), Schiebereglern (sliders), Listen und Tabellen sowie auch Rollbalken (scrollbars) usw. ist weitverbreitet und wird durch verschiedene Sprachen und Systeme unterstützt. Für viele Anwendungen reicht die Verwendung dieser Art von Standardelementen jedoch nicht aus, um eine adäquate graphische Schnittstelle zu erstellen. Es werden vielfach anwendungsspezifische visuelle Elemente benötigt, die Domänenobjekte auf dem Bildschirm problemnah darstellen. Um eine systematische Generierung von graphischen Schnittstellen mit direktmanipulativen Anwendungsobjekten zu unterstützen, ist es notwendig, nicht nur Werkzeuge zur Erstellung von *datenstrukturorientierten* Visualisierungen zu betrachten (z.B. Graphformatierer für Vererbungshierarchien oder Formulardarstellungen für Klassen- und Objektstrukturen), sondern auch Werkzeuge zur Erstellung von problemnahen und aufgabenorientierten Darstellungen für anwendungsspezifische Objekte und deren Beziehungen zu entwickeln. Die letztgenannte Klasse von Visualisierungen möchte ich als *modellbasierte Visualisierungen* bezeichnen, da zur Generierung eine bestimmte abstrahierende Sicht auf Domänenobjekte (oder ein „Modell“) bestimmt werden muß. In modellbasierten Visualisierungen sollen Domänenobjekte nicht durch Piktogramme auf dem Bildschirm präsentiert werden, sondern durch „Skizzen“, die auf der Basis von geometrischen Informationen unter Berücksichtigung von zusätzlichem Darstellungswissen erzeugt werden. Datenstrukturorientierte Visualisierungen werden hingegen direkt unter Bezugnahme auf den verwendeten Repräsentationsformalismus für Domänenobjekte generiert. Obwohl automatische Graph- und Formulardarstellungen nützliche und notwendige Teilkomponenten in modernen Benutzungsschnittstellen sind, steigt doch in vielen Anwendungen für die Gestaltung der Benutzungsschnittstelle der Bedarf an modellbasierten Visualisierungen. Visualisierungen dieser Art sind problemnah, d.h. sie werden auf vorgegebene Aufgaben ausgerichtet, die mit der Darstellung durchgeführt werden sollen. Visualisierungen dieser Art sind nicht nur zusätzliche Hilfsmittel, sondern definieren die Arbeitsumgebung für den Benutzer. Bei einer Einbettung in eine Benutzungsschnittstelle sind Visualisierungen so zu gestalten, daß die Aktionen zur Erfüllung der Aufgaben direktmanipulativ (z.B. mit einer Maus) durchführbar sind.

1.1 Ein Leitbeispiel für modellbasierte Visualisierungen

Um die Idee der modellbasierten Visualisierungen zu illustrieren und die Teilprobleme aufzuzeigen, die bei der Erstellung und Einbettung in eine Benutzungsschnittstelle entstehen, möchte ich ein Beispiel anführen. Eine Klasse von Anwendungen, bei denen Visualisierungen zur Realisierung von Benutzerhandlungen benötigt werden, sind interaktive Layoutsysteme für Einrichtungsgegenstände.

Beispiele für Benutzerhandlungen in diesen Anwendungen sind „Auswählen zur weiteren Bearbeitung“, „Plazieren“, „Verschieben“ usw. In dieser Arbeit wird als Anwendungs- und Leitdomäne die Konzeption eines Programms aus dieser Anwendungsklasse zur interaktiven Gestaltung der Inneneinrichtung einer Flugzeugkabine betrachtet (genannt XKL). Die Inneneinrichtung einer Flugzeugkabine besteht für diese Anwendung aus räumlichen Objekten wie Passagiersitzen, Flugbegleitersitzen (cabin attendant seats), Küchen (galleys), Waschräumen (lavatories) etc.

Abbildung 1 zeigt eine Oberfläche der XKL-Anwendung zur Plazierung von Einrichtungsgegenständen in der Flugzeugkabine. In dem Beispiel wurde in einem vorhergehenden Konstruktionsschritt schon eine Küche in der Kabine plaziert.

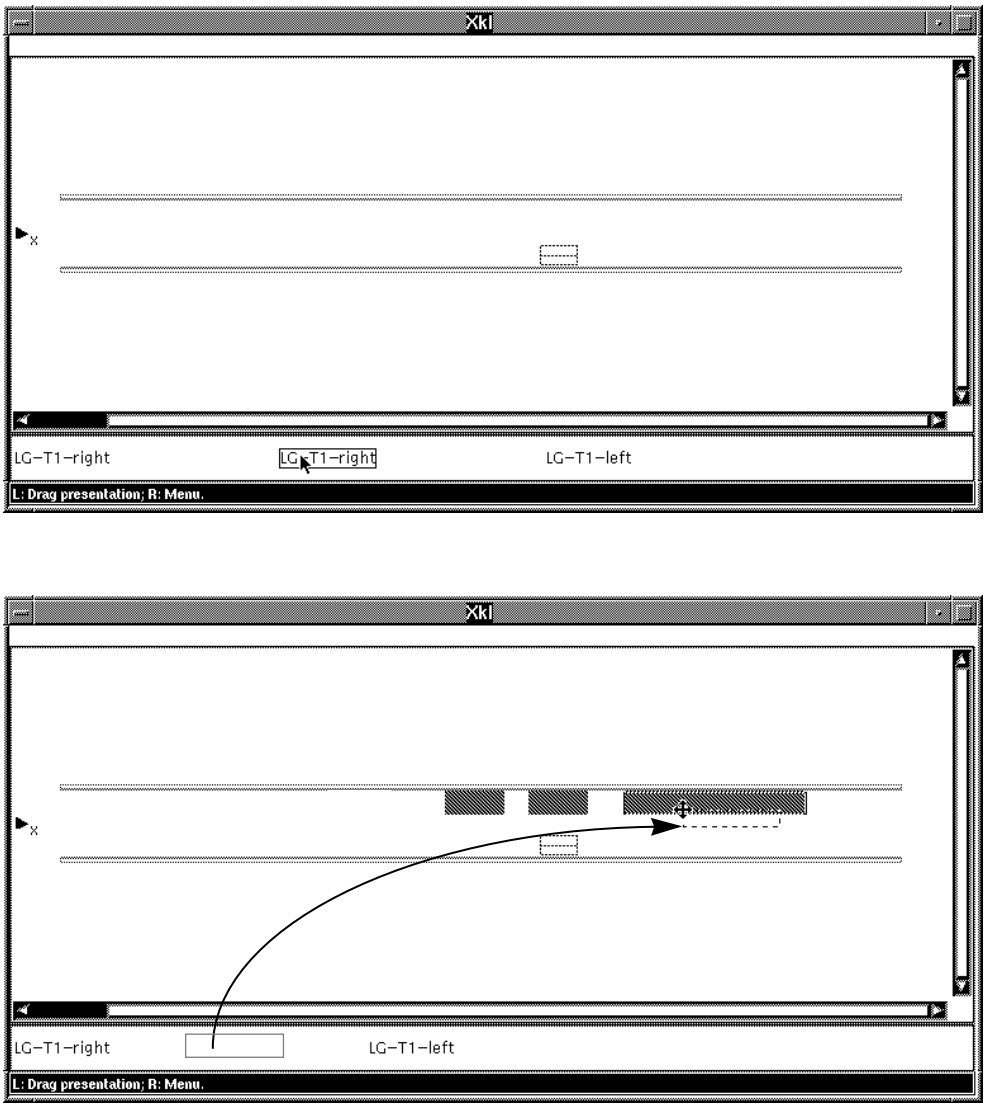


Abbildung 1. Beispiel für eine direktmanipulative Benutzungsschnittstelle für die Gestaltung der Inneneinrichtung einer Flugzeugkabine. In der unteren Objektpalette werden die noch plazierbaren Objekte angedeutet (die Maus zeigt im oberen Teilbild auf ein Kabinenobjekt vom Konzept „Longitudinal-Galley-Typ-1-right“). Bei den gezeigten Graphiken handelt es sich nicht um Zeichnungen, sondern um Bildschirmabzüge des mit Hilfe von HAMVIS erzeugten XKL-Systems (für weitere Beispiele siehe Kapitel 3.5).

Eine Menge von Objekten, die noch in die Kabine eingefügt werden können, wird in einer Objektpalette in dem unteren Teilfenster dargestellt. Diese „prototypischen“ Objekte sind maussensitiv und können mit der Maus in den Flugzeugrumpf gezogen werden. Nachdem ein Objekt aus der Palette selektiert wurde (oberer Teil von Abbildung 1), werden die möglichen Plazierungsbereiche in der Flugzeugkabine automatisch berechnet und anschließend dargestellt (siehe den unteren Teil von Abbildung 1).[1] Das selektierte Objekt kann auf einen der angezeigten Plazierungsbereiche geschoben und in das zugehörige Rechteck „fallengelassen“ werden.

Die skizzierten oberflächennahen Aktionen müssen durch Funktionen „hinter den Kulissen“ unterstützt werden. Die Berechnung der möglichen, noch in der Kabine fehlenden Einrichtungsgegenstände und der jeweils möglichen Plazierungsbereiche erfolgt *vor* bzw. *während* der Interaktionshandlungen durch anwendungsspezifische Berechnungsfunktionen.[2] Eine *anschließende* Speicherfunktion sichert die Lokalisierungsinformationen im Datenmodell. Benutzerhandlungen sind also mit anwendungsspezifischen automatischen Berechnungs- und Speicherfunktionen verwoben. Als *elementare Benutzerhandlungen* lassen sich in dem betrachteten Beispiel z.B. eine Auswahlhandlung und eine Lokalisierungshandlung erkennen. Offensichtlich ist es für die hier vorgesehene Art der Objektlokalisierung „einfacher“, einen Plazierungsbereich innerhalb einer Visualisierung zu selektieren als entsprechende Koordinaten in einem Textfenster einzutippen. Die Sequenz von Auswahl- und Lokalisierungshandlung kann auf einer höheren Ebene als *zusammengesetzte Benutzerhandlung* betrachtet werden. Automatische Berechnungs- und Speicherfunktionen sind aus dieser Perspektive Hilfsfunktionen, die für den Benutzer während der Interaktion transparent sein sollen.

Die Realisierung einer zusammengesetzten Handlung kann in dem betrachteten Beispiel durch eine Ziehen-und-Fallenlassen-Geste (drag-and-drop gesture) erfolgen. Es ergeben sich aus der Wahl der Interaktionsform entsprechende Einschränkungen für die zu verwendenden Interaktionskomponenten und für die Gestaltung von benötigten Visualisierungen. Zu beachten ist, daß in diesem Beispiel die Auswahlhandlung (Klicken auf das zu plazierende Objekt) und die Lokalisierungshandlung (Fallenlassen auf den vorgesehenen Plazierungsbereich) auf der Interaktionsebene mit einer einzigen Mausgeste (drag-and-drop) umgesetzt werden. Die Realisierung einer solchen Handlungssequenz an der Oberfläche kann also mehrere Teilhandlungen verschmelzen, so daß eine angemessene, „flüssige“ Bedienung der Anwendung ermöglicht wird. Die Betrachtung der „tieferen“ Teilhandlungen „Auswahl zur weiteren Bearbeitung“ und „Lokalisierung“ ist jedoch Voraussetzung für die Bestimmung der an der Oberfläche zu präsentierenden Anwendungsobjekte und Interaktionswerkzeuge. Es werden die Objekte angezeigt, die relevant für die Teilhandlungen bzw. -aufgaben sind: eine Menge von mög-

1. Aufgrund von technischen Einschränkungen des Fertigbaus (Einbauraster, Gewicht etc.) können Küchen nur an speziellen Orten (mit unterschiedlichen Freiheitsgraden) installiert werden.

2. Die Berechnung von möglichen Plazierungsbereichen ist eine vergleichsweise teure Operation, da die möglichen Plazierungsbereiche durch andere Kabinenobjekte beeinflußt werden, die vorher in die Kabine integriert wurden. Eine Layoutkonfiguration muß verschiedene gesetzliche Bestimmungen einhalten (Notausgänge, Fluchtwege, minimale Gangbreiten, minimale Abstände der Sitze von Wänden, etc.) und wird weiterhin durch Kostengesichtspunkte beeinflußt. Daher ist es in einer interaktiven Schnittstelle eventuell günstiger, Plazierungsbereiche nicht für alle Objekttypen a priori zu berechnen, sondern nur für das Objekt, das der Benutzer aus der Palette tatsächlich auswählt (siehe Abbildung 1). Ein Ansatz zur Entwicklung eines Expertensystems für das Kabinenlayout wird in [162] und [163] beschrieben. Das Problem, ein Layout bei gegebenen Einschränkungen automatisch zu bestimmen, wird anhand der Domäne des Gebäudedesigns auch von Schwarz et al. in [278] untersucht.

lichen Objekten unterschiedlichen Typs (Küchen, Waschräume, Sitze usw.), schon plazierte Objekte in der Umgebung (für die Auswahlhandlung des „nächsten“ zu plazierenden Objekts) sowie die Flugzeugkabine als globales (statisches) Referenzsystem für die Lokalisierungshandlung. Für die zur Auswahl stehenden Objekte wird ein eigenes Unterfenster[3] des speziellen Standard-Interaktionsbausteins „Auswahlpalette“ benötigt, während für die Lokalisierungshandlung ein Graphikfenster verwendet wird, das unabhängig von der Palette gerollt werden kann. Die Gestaltung der Oberfläche und die Darstellungsform der gezeigten Objekte wird sowohl durch die Art der zugrundeliegenden Handlungen als auch durch deren oberflächennahe Realisierungsform bestimmt.

Bei der in dieser Arbeit betrachteten Klasse von direktmanipulativen Anwendungen ist ein Werkbank-orientierter Ansatz (im Sinne eines allgemeinen CAD-Systems) nicht geeignet, da nicht nur die direkt manipulierten Objekte gezeigt werden, sondern noch ein gewisser „Kontext“ benötigt wird (in der hier betrachteten Anwendungsklasse z.B. Referenzrahmen, Landmarken etc.), der nicht durch den Benutzer selbst gezeichnet werden soll. Die für die Interaktion notwendigen Objekte (z.B. Plazierungsbereiche) werden automatisch unter Berücksichtigung von Domänenwissen berechnet und dargestellt. Weiterhin kann der Benutzer der Anwendung durch Zusatzdarstellungen in die Lage versetzt werden, Entscheidungen angemessen zu treffen. Für eine Verschiebehandlung können z.B. Detaildarstellungen zur Präsentation von Zusatzinformationen bzgl. spezieller räumlicher Konstellationen verwendet werden (siehe Abbildung 2). Für die Positionierung einer Küche ist z.B. in der XKL-Anwendung die Position des Wasseranschlusses wichtig. In der Konfiguration im linken Teil der Abbildung 2 ist der Wasseranschluß günstiger plaziert, da weniger Raum verschenkt wird. Weiterhin muß bei dieser Einbauweise nur an einer Seite die Einbauform der Stauschränke angepaßt werden (Kostenersparnis). Im Gegensatz zu der Darstellung in Abbildung 1 wird für die Zusatzdarstellung eine andere Perspektive gewählt, und die Flugzeugkabine enthält weitere Objekte wie z.B. Stauschränke und Wasseranschlüsse.

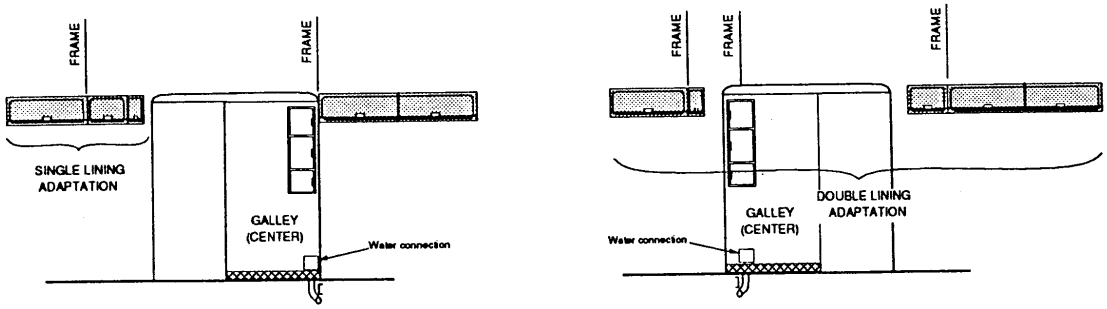


Abbildung 2. Detaildarstellung für die Entscheidungsfindung bei einer Lokalisierungs- oder Verschiebehandlung (siehe Text, entnommen aus einem Handbuch eines Flugzeugherstellers zur Konfiguration der Inneneinrichtung: „Cabin Configuration Guide“).

Für die Präsentation der Informationen wird ein zusätzliches Fenster benötigt. Es wäre möglich, für Zusatzdarstellungen bei Bedarf ein neues, separates Fenster zu erzeugen. Das „plötzliche“ Aufklappen eines Fensters erzeugt aber eine unruhige Interaktionsform. Eventuell müssen Fenster von Hand verschoben werden usw. Es ist daher günstiger, für diese Zusatzinformation zur Entwurfszeit ein

3. Der Begriff „Unterfenster“ (synonym auch „Teilfenster“) bezeichnet einen rechteckigen Bereich innerhalb eines Hauptfensters. Nur Hauptfenster sind auf dem Bildschirm direkt verschiebbar.

geeignetes Teilfenster innerhalb des für die Konstruktionsaktivität verwendeten Hauptfensters vorzusehen. Falls allerdings schon ein verwendbares Teilfenster eingeplant wurde, kann dieses zur Darstellung der Küchenkonfiguration eventuell mitgenutzt werden.

Das Ziel eines Schnittstellendesigners ist es, die Gesamtstruktur einer Anwendung so in Teilaktivitäten („Teildialoge“) zu zerlegen, daß jede Aktivität eine Menge von zusammengehörigen Benutzeraufgaben unterstützt. Eine Aktivität ist mit einem Ziel verbunden, das durch Ausführung von alternativen Handlungen erreicht werden kann. Ist das Ziel erreicht, kann die Aktivität beendet werden.[4] Da aber mehrere Handlungsmöglichkeiten zur Verfügung stehen können (z.B. Erzeugen und Lokalisieren eines neuen Objekts, Verschieben eines Objekts, Ausrichten von Objekten, Löschen von Objekten), muß ggf. der Benutzer entscheiden, ob die Aktivität beendet werden kann.

Durch die Beschränkung des Bildschirmplatzes ist es meist nicht möglich, für jede (zusammengesetzte) Benutzerhandlung eine separate Visualisierung vorzusehen. Auch wenn genügend Platz zur Verfügung steht, ergibt sich durch Herstellung eines inhaltlichen Zusammenhangs zwischen Teildarstellungen in einer Gesamtdarstellung eine vorteilhaftere Interaktionsform. Innerhalb der Gesamtdarstellung für eine Aktivität sind die vorgesehenen Visualisierungen so zu konzipieren, daß sie mehrere Benutzerhandlungen gleichzeitig unterstützen (z.B. eine Plazierung von Objekten sowie auch deren Verschiebung). Auch die oberflächenorientierten Realisierungsmöglichkeiten von Handlungen (z.B. durch Mausgesten) müssen auf mehrere (zusammengesetzte) Handlungen abgestimmt werden.

In Anwendungen wie XKL, in denen dreidimensionale geometrische Objekte manipuliert werden, besteht weiterhin das Problem, adäquate Sichten zu bestimmen, so daß die manipulierten Objekte in der Graphik auch sichtbar sind. Jedes Graphikteilfenster zeigt geometrische Objekte aus einer speziellen Perspektive[5] und mit einem angepaßten Detaillierungsgrad.

4. Da die Organisation der Speicherung von Zwischenresultaten zusammen mit Wiederaufsetzmöglichkeiten (auch bei Fehlern) nichts Wesentliches zur Visualisierungsgestaltung beiträgt, werden diese Aspekte hier nicht näher betrachtet.

5. Eine künstlerische „Verschmelzung“ von Darstellungen mit verschiedenen Perspektiven möchte ich hier nicht betrachten.

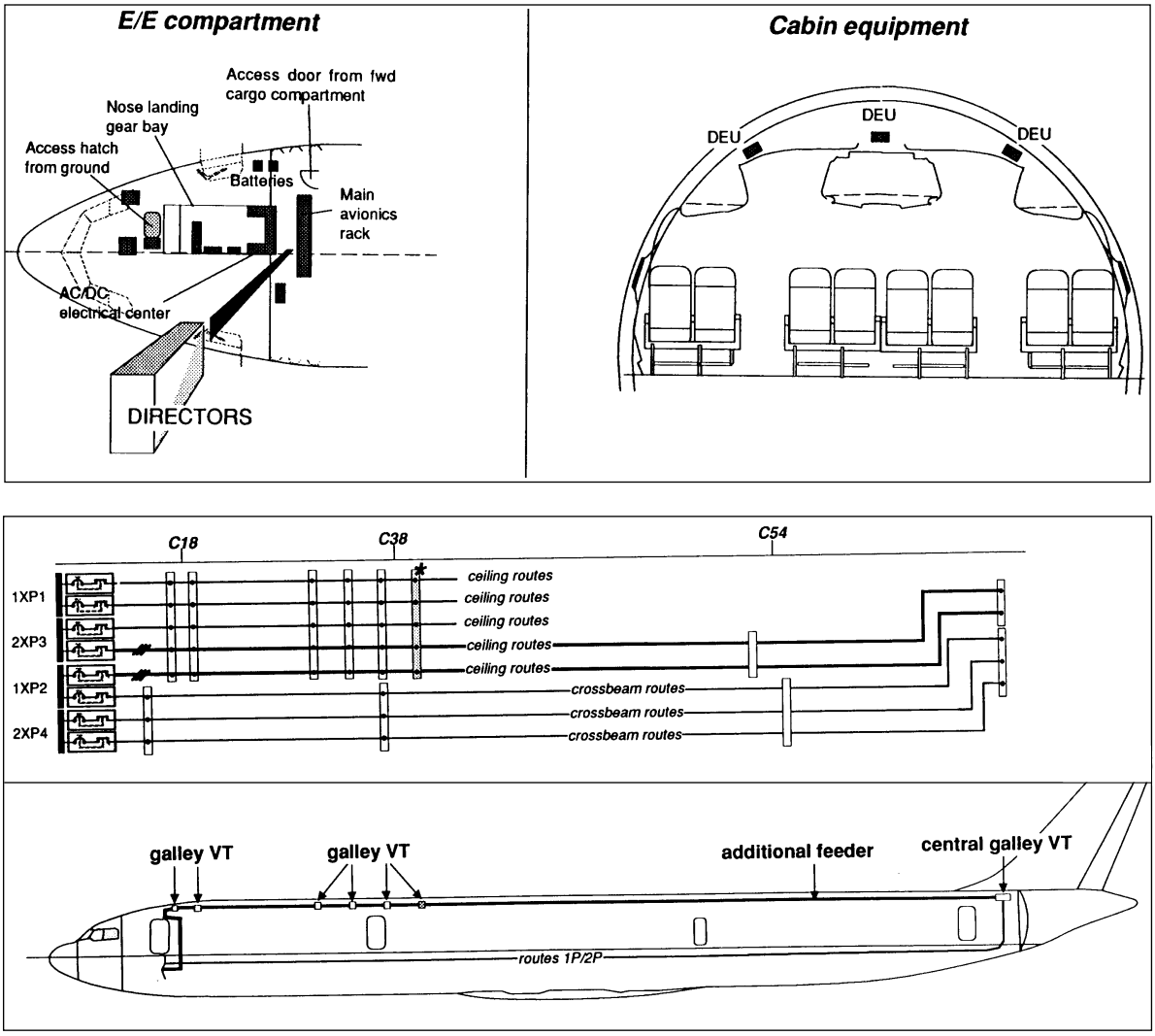


Abbildung 3. Visualisierungen in technischen Zeichnungen (siehe Text, entnommen aus einem Handbuch eines Flugzeugherstellers zur Konfiguration der Inneneinrichtung: „Cabin Configuration Guide“).

Nicht nur für Benutzungsschnittstellen ist es wichtig, Teilvisualisierungen aufeinander abzustimmen und zueinander in Beziehung zu setzen. Die hier angesprochenen Aspekte sind grundlegend für visuelle Kommunikationsformen. Abbildung 3 zeigt zwei handgezeichnete technische Zeichnungen, die jeweils aus zwei Teilen bestehen. In der oberen Darstellung werden zwei Teilzeichnungen dazu verwendet, Anwendungsobjekte aus verschiedenen Sichten darzustellen. Die Teilzeichnungen für Geräte der elektronischen Ausstattung sind nebeneinander plaziert, hätten aber auch übereinander angeordnet werden können. Die Wahl wurde vermutlich durch das verwendete Papierformat bestimmt (DIN-A4 Querformat).

In einem Fenster innerhalb einer Benutzungsschnittstelle müssen Teildarstellungen in ähnlicher Weise angeordnet und zueinander in Beziehung gesetzt werden wie in einer technischen Zeichnung. Die relative Anordnung hängt dabei vom „Status“ des in den Teilfenstern dargestellten Inhalts ab. Man kann z.B. zwischen Hauptdarstellungen (mit den zu manipulierenden Objekten), Zusatzdarstellungen

(z.B. Detailausschnitten, Darstellungen von speziellen räumlichen Konstellationen etc.) sowie Standard-Interaktionsbausteinen wie Paletten für Anwendungsobjekte oder für Interaktionswerkzeuge (toolboxes) unterscheiden. Sowohl die Anordnung von Teildarstellungen als auch deren Inhalt ist abhängig von der Bedeutung bzw. der kommunikativen Aufgabe und der rhetorischen Funktion der Einzeldarstellungen in der Gesamtoberfläche. Die untere Zeichnung in Abbildung 3 belegt dieses an einem Beispiel. Die Koordinatensysteme der Teildarstellungen im unteren Bild sind in der x-Dimension zueinander in Beziehung gesetzt. Das xy-Referenzsystem „Flugzeug“ der unteren Zeichnung (im Längsschnitt) wird in der oberen, etwas abstrakteren Skizze für ein Referenzsystem entlang der x-Achse mitbenutzt. Der Flugzeugrumpf braucht also in dieser Teildarstellung nicht noch einmal gezeichnet zu werden (Redundanzvermeidung). Bedingt durch diese *inhaltliche Beziehung* wurde ein vertikal ausgerichtetes Anordnungsschema für die Teildarstellungen vorgesehen.

Nach der Bestimmung von notwendigen Standard-Interaktionsbausteinen und des Inhaltes von graphischen Darstellungen müssen zur Entwicklungszeit der Anwendung noch Darstellungsattribute wie Farben, Füllmuster, Liniendicken usw. ausgewählt werden, so daß graphische Objekte zur Benutzungszeit (Laufzeit) der Anwendung durch ein UIMS (user interface management system) adäquat gezeichnet werden können.[6] Wenn wir davon ausgehen, daß die Objekte in den hier betrachteten Anwendungen nicht realitätsnah, d.h. mit den Farben der Realweltobjekte dargestellt werden, ist die Wahl der Zeichenattribute primär von der rhetorischen Funktion der Objekte beeinflußt. Weiterhin spielen Konventionen (schwarzer oder weißer Hintergrund?) sowie Aspekte der menschlichen Perzeption eine Rolle. Um eine leichte Wiedererkennung von Objekten in verschiedenen Teildarstellungen zu gewährleisten, dürfen die Darstellungsattribute für gleiche Objekte z.B. nicht unmotiviert geändert werden. Allerdings sollten in einer speziellen Visualisierung die für die Benutzerhandlung weniger wichtigen Objekte abgeschwächt dargestellt werden, um eine Überfrachtung der Darstellung zu vermeiden (siehe z.B. den linken Teil der oberen Zeichnung in Abbildung 3). Wichtige Objekte sind jedoch hervorzuheben. Zu beachten ist, daß sich die rhetorische Stellung von Einzelobjekten im Laufe der Benutzung einer Anwendung ändern kann.

Die vorgestellten Beispiele belegen, daß das Design und die Komposition von visuellen Präsentationen, die in einer interaktiven Benutzungsschnittstelle eingebettet werden, eine nicht-triviale Aufgabe ist. Es sind mehrere, unter Umständen widersprüchliche Einschränkungen zu erfüllen, um zu einer ansprechenden, kohärenten Gesamtdarstellung zu gelangen. Zusammenfassend läßt sich sagen, daß bei der Schnittstellengestaltung verschiedene Einflüsse und Einschränkungen koordiniert werden müssen.

1.2 Ziel der Arbeit und Einordnung in die Forschungslandschaft

Diese Arbeit beschreibt einen neuen computergestützten Ansatz zur systematischen Entwicklung von aufgabenorientierten modellbasierten Visualisierungen für Benutzungsschnittstellen. Das vorgeschlagene methodische Vorgehen für die Visualisierungsgenerierung ist eingebettet in einen wissensbasierten Gesamtrahmen zur Entwicklung von Anwendungsprogrammen. Der Begriff „methodisches Vorgehen“ bezieht sich hierbei auf eine integrierte Behandlung der Schnittstellenentwicklung und der Spezifikation von Anwendungsfunktionen mit formalen Modellen. In dieser Arbeit wird insbesondere

6. Man könnte argumentieren, daß eine Wahl von Farben usw. auch direkt vom Benutzer der Anwendung zur Laufzeit erfolgen könnte. Obwohl dieses durchaus sinnvoll sein kann, müssen doch in jedem Fall zur Entwicklungszeit aufeinander abgestimmte Standardwerte (defaults) für die Zeichenparameter festgelegt werden.

untersucht, wie beschreibungslogische Repräsentationstechniken ([183], [237]) eingesetzt werden können, um deklarative Modelle zur Repräsentation des für die Schnittstellenerzeugung relevanten Wissens zu formulieren. Deklarativ heißt in diesem Zusammenhang nicht nur, daß explizite (oder reifizierte) Repräsentationsformen verwendet werden, die dann mit bestimmten „Werkzeugen“ transformiert werden können, sondern bedeutet, daß sich die Herleitung der Oberflächeneinheiten in wesentlichen Teilen als logische Ableitung formulieren läßt bzw. aus dieser Perspektive gedeutet werden kann.

Die wesentlichen Teilprobleme, die bei der Entwicklung eines Unterstützungssystems zur Visualisierungsgenerierung gelöst werden müssen, sind:

1. Repräsentation von Wissen über grundlegende Konzepte und Relationen für eine Anwendungsklasse in einer Form, die es erlaubt, die Erzeugung und Ausgestaltung von Visualisierungen als wissensbasierte Inferenzschritte deklarativ zu beschreiben,
2. Bereitstellung von Möglichkeiten zur Strukturierung einer Applikation in Teilphasen oder „Aktivitäten“, die jeweils zur Ausführung von mehreren zusammengehörenden Benutzeraktionen gedacht sind und an der Benutzungsoberfläche des Rechners jeweils durch ein Fenster repräsentiert werden.
3. Schaffung von Repräsentationsformen zur Definition von Beziehungen und Abhängigkeiten zwischen Benutzeraktionen und automatischen Berechnungs- und Speicherfunktionen,
4. Formalisierung von Schlußfolgerungsprozessen, so daß Visualisierungen zur Entwicklungszeit einer Anwendung hergeleitet und vorgeschlagen werden können, mit denen Benutzeraktionen interaktiv zur Laufzeit durchführbar sind (notwendige Perspektive, adäquater Detailreichtum, usw.),
5. Bereitstellung von Repräsentationsformen zur Bestimmung von Zeichenattributen und Darstellungsformen für graphische Objekte (z.B. Techniken zur Hervorhebung),
6. Analyse der Möglichkeiten zur Komposition von Visualisierungen mit Standardinteraktionsbausteinen zur Einbettung in ein Fenster der Benutzungsschnittstelle.

Für die einzelnen Teilaspekte wurden in der Literatur verschiedene Lösungsvorschläge vorgestellt. Ich möchte in den folgenden Abschnitten eine Grobeinordnung dieser Arbeit in bestehende Forschungsaktivitäten geben (siehe Kapitel 2 für eine detaillierte Übersicht).

Grundannahme des vorgestellten Ansatzes ist, daß Schnittstellen und automatische Berechnungs- und Speicherfunktionen aufeinander abgestimmt sein müssen und gleichberechtigt mit formalen Methoden entworfen und realisiert werden; eine inzwischen weitverbreitet vertretene Forderung im Bereich der Mensch-Computer-Interaktion (MCI oder HCI: Human-Computer Interaction) [245]. Der Schwerpunkt dieser Arbeit liegt auf der Unterstützung der Generierung von Visualisierungen, die geeignet sein sollen, direktmanipulative Interaktionshandlungen zur Manipulation von Objekten einer Anwendungsdomäne zu ermöglichen. Für die Ausführung von Handlungen werden im Gegensatz zu CAD-orientierten Zeichenprogrammen (Werkbank-orientierter Ansatz) nicht nur die manipulierten Objekte dargestellt (z.B. Plazierungsbereiche, s.o.), sondern nicht-manipulierbare, aber für die Interaktionen notwendige Anwendungsobjekte wie z.B. auch Referenzsysteme (eine Flugzeugkabine) oder Landmarken (Türen, Cockpitfenster). Ähnlich wie beim Werkbank-orientierten Gestaltungsansatz hat der Endbenutzer ein Repertoire von Handlungsmöglichkeiten, aus denen er eigenständig auswählen

kann. Der Benutzer soll möglichst nicht explizit aufgefordert werden, dieses oder jenes zu tun, d.h. eine modale Interaktionsform sollte vermieden werden.

Verschiedene Forschungsarbeiten aus dem Bereich Mensch-Computer-Interaktion betonen die Notwendigkeit von expliziten Modellen für die Wissenquellen, die für den Entwurf von Gebrauchssoftware und deren Benutzungsschnittstellen benötigt werden, und arbeiten Schwächen von Prototypenansätzen (z.B. „Rapid Prototyping“) heraus [130]. Für die Anwendungsentwicklung wurden verschiedene Modellierungsformen auch für die Design- bzw. Entwurfsphase vorgeschlagen ([233], siehe auch Kapitel 2). Einer der wesentlichen Modellierungsansätze im HCI-Bereich besteht darin, als Ausgangspunkt der Entwicklung einer Anwendung die Aktionen (oder Aufgaben, engl. tasks) des Endbenutzers zu analysieren und zu modellieren. Es ist dabei zwischen praktischen Arbeiten zur modellbasierten Generierung von Oberflächen (z.B. [309], [230], [184], [298]) und Arbeiten zur theoretischen Untersuchung von Spezifikationstechniken zu unterscheiden (z.B. [1], [65], [74], [77]).

Eine häufig vertretene Forderung ist es, Aktionen- bzw. Aufgabenmodelle so zu gestalten, daß sie direkt ausführbar sind (Executable Task Analysis [57]) bzw. als Basis für die weitere formale Modellierung dienen können, so daß sich ein kontinuierlicher Übergang zu Spezifikationen auf der Ebene eines Ziel-UIMS ergibt [23]. Das Ziel dieser Arbeit ist es, Entwurfsmethoden und Repräsentationsformen nicht nur für die Arbeit mit „Bleistift und Papier“ (einschließlich computergestützter Zeichenmedien) zu konzipieren. Aus vorgegebenen Modellen für Darstellungswissen und Wissen über Aktionen und Interaktionstechniken sowie den zur Entwurfszeit aufgestellten Modellen für Domänenwissen soll das für die Einbettung der Visualisierungen in eine Benutzungsschnittstelle notwendige Laufzeitsystem automatisch generiert werden können. Von anderen Autoren vorgeschlagene Entwurfsmethoden, deren Ergebnisse nicht automatisch in ein Laufzeitsystem umsetzbar sind, können allerdings durchaus als Grundlage bzw. Ergänzung dienen.

Die praktischen MCI-Arbeiten zur Aufgaben- und Aktionenmodellierung belegen u.a., daß die explizite Betrachtung von Aktionen auf „tiefer“ Ebene (also nicht nur auf der Ebene der direkten Interaktionsgesten wie z.B. Mausklicks) sehr vielversprechend ist, um die enorme Komplexität der Entwicklung von graphischen Oberflächen beherrschen zu können. Die Granularität der Aktionenmodellierung und die generelle Ausrichtung der Modelle variiert jedoch in verschiedenen Ansätzen beträchtlich (siehe Kapitel 2). Zu beachten ist, daß in dieser Arbeit modellbasierte Visualisierungen betrachtet werden (s.o.) und nicht datenstrukturorientierte Visualisierungen aus Aktionen- und Domänenmodellen hergeleitet werden sollen (siehe z.B. [252]).

Für modellbasierte Visualisierungen sind neuere Forschungsarbeiten aus der KI-Forschung relevant. Auf der theoretischen Seite sind z.B. Ableitungen über Teil-Ganzes-Relationen in Bezug auf geometrische und begriffliche Informationen bei der Visualisierungsgestaltung zu berücksichtigen. Auch die Integration von anwendungsspezifischen Domänenmodellen und Aufgabenzerlegungen mit vordefinierten, generischen Konzepten für Domänenobjekte und Aktionen wurden in KI-Ansätzen in vielfältiger Form untersucht. Zu nennen sind hier insbesondere die Ansätze zur Inhaltsplanung in Sprachgenerierungssystemen (siehe z.B. [18], [134]) sowie Arbeiten zu Intelligenten Multimedia-Präsentationssystemen (IMMP-Systeme, siehe [331]).

Obwohl es in dieser Arbeit nicht darum geht, daß ein Planungsagent[7] dynamisch zur Benutzungszeit situationsangepaßte Präsentationen generieren soll, möchte ich aufzeigen, daß die Arbeiten aus dem

IMMP-Bereich auch für die Unterstützung von Schnittstellenentwicklern (HCI-Ansatz) relevant sind. Gerade für modellbasierte Visualisierungen ist es notwendig, auch die kommunikative Funktion von Teildarstellungen explizit zu repräsentieren, um z.B. entsprechende Zeichenattribute vorschlagen zu können oder die Gesamtgestaltung der Darstellungen abstimmen zu können (siehe die oben aufgeführten Punkte). Auch die von IMMP-Systemen vorgenommene Inhaltsplanung kann in diesem Kontext betrachtet werden. Wenn für die Gestaltung einer kohärenten Darstellung bestimmte Objekte notwendig sind, so sollte dieses *zur Entwicklungszeit* der Anwendung *hergeleitet* werden. Falls die Informationen nicht aus den deklarierten Domänenmodellen ableitbar sind, so kann dieses dem Schnittstellen- bzw. Anwendungsentwickler mitgeteilt und Domänenmodelle können entsprechend angepaßt werden. Allerdings ist zu beachten, daß die Ableitungen aufgrund von konzeptuellen Informationen erfolgen müssen, da konkrete Objekte zur Entwicklungszeit nicht unbedingt bekannt sind. Das formale Grundgerüst der Beschreibungslogik ermöglicht die Modellierung des notwendigen Wissens in adäquater Form (siehe z.B. die Ausführungen in [25], [32], [31]).

Neben den formalen Modellen sind aber auch praktische Aspekte nicht zu vernachlässigen. Gerade bei Verwendung von formalen Repräsentationstechniken muß aufgezeigt werden, in welcher Form Oberflächenkomponenten gestaltet werden können, damit zur Entwicklungszeit einer Anwendung die Spezifikation von Modellen einfach möglich ist und die Präsentation der Ergebnisse von internen Ableitungen durch einen Schnittstellenentwickler nachvollzogen werden kann (siehe z.B. [93], [176]). Durch graphische Interaktionstechniken und speziell entwickelte visuelle Sprachen lassen sich theoretische Aspekte durchaus auf sehr einfache Weise vermitteln, so daß eine formale Modellbildung für die Schnittstellenentwickler nicht unbedingt zu einer Erhöhung der Komplexität führen muß. Zur Erzielung einer Vereinfachung bei der Entwicklung von Oberflächen mit modellbasierten Visualisierungen ist es m.E. wichtig, eine Aktionenmodellierung auf einer Granularitätsebene durchzuführen, so daß eine Beziehung zwischen den Aktionenmodellen (auf verschiedenen Abstraktionsebenen) zu Elementen der fertigen Oberfläche erkennbar bleibt.

Auf der softwaretechnischen Seite sind Aspekte der Gestaltung von User Interface Management Systemen (UIMS) relevant, da ein UIMS als Zielsystem (backend) verwendet werden muß, um eine Integration von modellbasierten Visualisierungen in ein Rechnersystem zu erreichen. Moderne UIMS bieten Werkzeuge und Programmierkonzepte auf einem hohen Abstraktionsniveau, mit dem die notwendigen Basisinteraktionswerkzeuge für die in dieser Arbeit betrachteten Anwendungsklasse realisiert werden können. Eine objektorientierte UIMS-Bibliothek für Interaktionswerkzeuge (für eine spezielle Anwendungsklasse) bietet jedoch eine recht komplexe „Infrastruktur“, die erst nach längerer Einarbeitung nutzbar wird. Es ergibt sich auch hier die Frage, wie eine Vereinfachung erzielt werden kann.

Die Basisidee dieser Arbeit besteht darin, daß der Entwickler die gewünschten „Interaktionsdienste“ während der Entwicklungsphase einer Anwendung zunächst grob beschreibt. Die Beschreibungsperspektive bzw. das Beschreibungsvokabular liefert hierzu die aus der Mensch-Computer-Interaktion bekannte tiefe Modellierung von Benutzeraktionen (siehe hierzu Kapitel 2). Allerdings muß die Beschreibung der Endbenutzeraktionen auf der Ebene des Umgangs mit Domänenobjekten erfolgen und nicht auf der Ebene der Manipulation von Graphikobjekten (mit graphischen Gesten). Bei der

7. Auch in HCI-Ansätzen wird der Begriff eines Agenten verwendet [1], jedoch sind hier die Einheiten auf kleinerer Granularitätsstufe modelliert. Es liegt hier eine andere Sicht vor.

Aktionenmodellierung ist zu unterscheiden zwischen der *konzeptuellen Beschreibung einer Aktion* (mit entsprechenden Kasusrollen für die manipulierten Objekte bzw. die erzeugten Objekte) und der *Beschreibung der Zerlegung einer Aktion*. Die Aktionenzerlegung wird *anwendungsspezifisch* durch den Entwickler definiert. Durch die konzeptuelle Beschreibung der mit den Einzelaktionen in Zusammenhang stehenden Domänenobjekte lassen sich durch formale Schlußfolgerungen die „passenden Aktionskonzepte“ aus dem *vorgegebenen Modell von Aktionenkonzepten* bestimmen. Durch das vordefinierte generische Aktionenmodell ist also der mögliche Raum der Beschreibungen strukturiert. Aktionen werden durch den Entwickler immer feiner beschrieben, so daß auf unterer Ebene eine *automatische Zuordnung von Diensten des UIMS* bzw. der anwendungsklassenspezifischen UIMS-Bibliothek erfolgen kann.

Die Aktionenzerlegung definiert die grobe *Dialogstruktur* der Anwendung. Über die Aktionenmodellierung wird dabei auch der *„Basisinhalt“ von Visualisierungen* festgelegt. Durch die Konzepte der Aktionen werden *Einschränkungen für die konkrete Ausgestaltung der Visualisierungen* festgelegt. Die konzeptuelle Beschreibung von Interaktionsdiensten auf tiefer semantischer Ebene ermöglicht es, den Basisinhalt adäquat zu „vervollständigen“ (siehe die Komponenten zur Inhaltsplanung in IMMP-Systemen). Die *Vervollständigung des Basisinhalts* und die weiteren Schritte zur Strukturierung der Gesamtanwendung bis hin zur konkreten *Präsentationsstruktur*, in der alle Visualisierungskomponenten mit ihren Darstellungsfenstern, Zeichenattributen usw. festlegt sind, müssen *auf der Basis von konzeptuellen Informationen* erfolgen, da konkrete Anwendungsobjekte zur Entwicklungszeit nicht notwendigerweise bekannt sind. Auch hier ist also eine besondere Modellbildung und eine Formalisierung durch beschreibungslogische Schlüsse notwendig.

Es wird deutlich, daß sich durch die Verknüpfung und Integration verschiedener Forschungsgebiete interessante Perspektiven für die Oberflächenerstellung und auch für die Entwicklung von Anwendungen ergeben. Es ist jedoch notwendig, bestehende Arbeiten zu integrieren und ein abgestimmtes Rahmenwerk zu konzipieren, das dem Schnittstellenentwickler formale Repräsentationstechniken und interaktive Werkzeuge an die Hand gibt, mit denen die interagierenden Teilprobleme bearbeitet werden können.

Es ist das Ziel dieser Arbeit, ein Rahmensystem für die methodische Visualisierungsgenerierung auf der *Basis von konzeptuellen Informationen* zu erstellen, bei dem eine *Trennung zwischen Entwurfszeit (bzw. Entwicklungszeit) und Laufzeit (oder Benutzungszeit)* der Anwendung vorgenommen wird. Wesentliche Gestaltungsmerkmale einer Oberfläche und der darin präsentierten Visualisierungen sollen durch die beteiligten Personengruppen zur Entwicklungszeit bewertbar sein.

Ich möchte mit dieser Arbeit eine Basis schaffen für die Definition von wiederverwendbaren formalen „Grundbausteinen“, die der Design- bzw. Entwurfsebene zuzuordnen sind und nicht nur auf der Ebene der oberflächennahen Interaktionselemente anzusiedeln sind. Die in dieser Arbeit entwickelten Modellierungsmethoden und Repräsentationstechniken zur modellbasierten, aufgabenorientierten Konstruktion von Visualisierungen werden anhand des interaktiven Rahmensystems HAMVIS (HAMburger VIsualisierungs-System) demonstriert, das prototypisch implementiert wurde.

1.3 HAMVIS: Ein Rahmensystem zur methodischen Visualisierungsgenerierung

In diesem Abschnitt wird die Grobkonzeption des HAMVIS-Rahmensystems anhand des XKL-Anwendungsszenarios vorgestellt. In Ergänzung zu den im „Software-Engineering“ üblichen Phasenmodellen möchte ich die notwendigen Teilschritte zur Generierung von Visualisierungen hervorheben. In Abbildung 4 sind die im Kontext von HAMVIS besonders betrachteten Phasen als Arbeitsschritte der Entwickler illustriert. Es wird davon ausgegangen, daß Teilphasen mehrfach durchlaufen werden müssen.

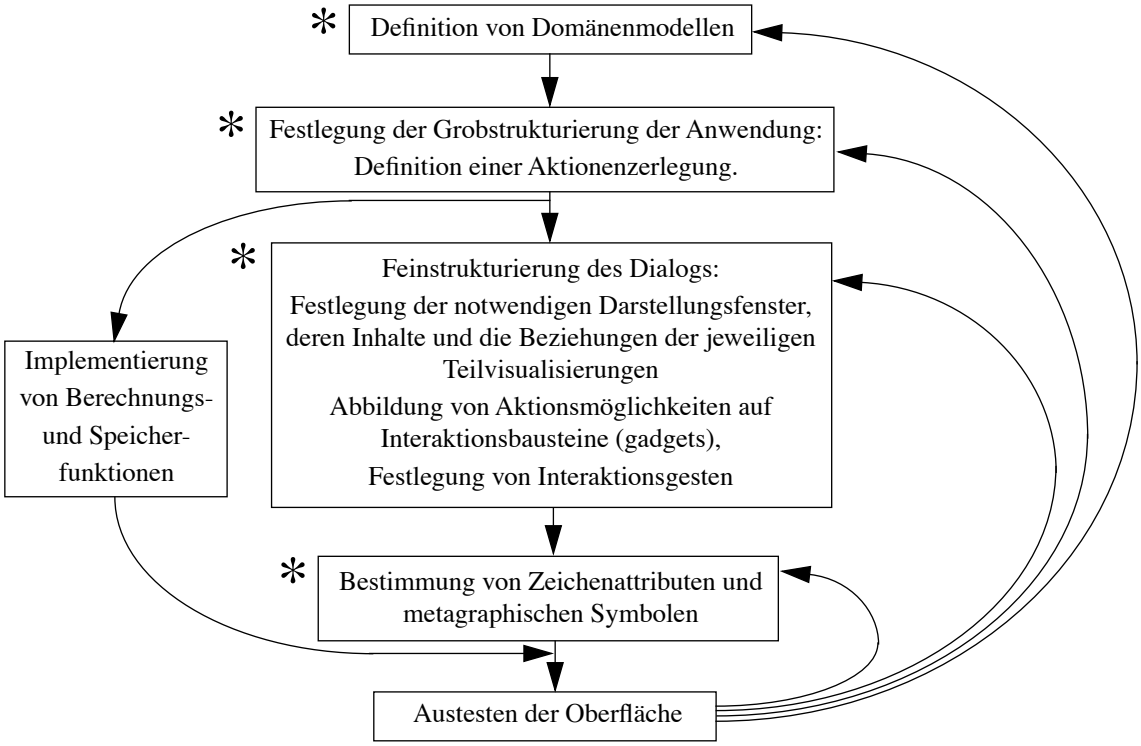


Abbildung 4. Grobüberblick über die Arbeitsschritte der Entwickler zur Generierung von aufgabenorientierten visuellen Oberflächenkomponenten. Phasen, die mit dem HAMVIS-Rahmensystem unterstützt werden, sind mit einem Stern gekennzeichnet.

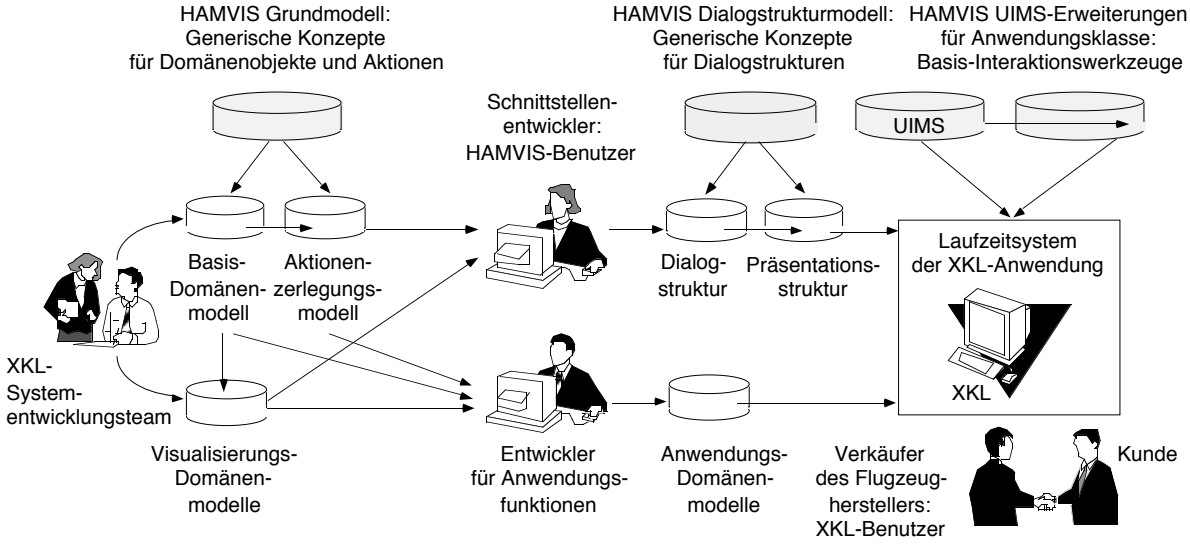


Abbildung 5. Szenario für die Entwicklung der XKL-Anwendung mit HAMVIS.

HAMVIS unterstützt die Entwicklung von Anwendungen in dem Szenario aus Abbildung 5. An der Entwicklung einer Anwendung sind mehrere zu unterscheidende Personengruppen beteiligt. Ein Anwendungssystem – in unserem Fall XKL – wird für einen bestimmten *Anwender* erstellt (auch Benutzer oder Endbenutzer genannt). In diesem Fall handelt es sich um einen Kundenberater eines Flugzeugherstellers, der wiederum der Auftraggeber für XKL ist. XKL soll bei einer Fluggesellschaft, dem Kunden eines Flugzeugherstellers, vom Kundenberater zur interaktiven Konfiguration eines Verkehrsflugzeugs eingesetzt werden. Unter Einbeziehung des Benutzers und des Auftraggebers wird die Anwendung von einem *Entwicklungsteam* eines Softwarehauses erstellt. Hierzu werden die *von HAMVIS bereitgestellten Modelle* und Bibliotheken eingesetzt.

Die Untersuchungen wurden für die Anwendungsklasse der Layoutprobleme durchgeführt. Für diese Anwendungsklasse stellt HAMVIS ein *Grundmodell* bereit, mit dem die relevanten Benutzerhandlungen formal beschrieben sind. Vordefinierte Bausteine sind im oberen Teil der Abbildung in dunklem Grau dargestellt. Sowohl das Grundmodell als auch die Aktionenmodelle sind durch beschreibungslogische Repräsentationsstrukturen beschrieben. Im HAMVIS-Ansatz müssen *anwendungsspezifische Domänenmodelle* als Erweiterung des HAMVIS-Grundmodells durch beschreibungslogische Repräsentationsformen definiert werden (siehe den mittleren Teil der Abbildung). Domänenmodelle werden durch das Systementwicklungsteam erstellt. Im Entwicklungsteam wird die Modellierung der Domäne und die Strukturierung der Anwendung festgelegt. Aktionen, die durch den Benutzer durchgeführt werden sollen, müssen durch automatische Berechnungs- und Speicherfunktionen unterstützt werden, die parallel zur Oberflächenentwicklung durch den HAMVIS-Benutzer von einem *Anwendungsentwickler* auf der Basis der Vorgaben aus dem Aktionenmodell implementiert werden (siehe auch Abbildung 4). Für die Definition dieser Funktionen und für die Gestaltung der Oberfläche werden Datenmodelle entworfen und Daten bereitgestellt (z.B. CAD-Daten für die Geometrie des Flugzeugs und der Einrichtungsgegenstände). Für die Implementation von Berechnungs- und Speicherfunktionen können komplexe Werkzeuge eingesetzt werden. Für HAMVIS sind jedoch nur die Typen der Parameter und Werte von Anwendungs- und Berechnungsfunktionen relevant, ansonsten werden sie in dieser Arbeit als Blackbox betrachtet.[8] Das Endprodukt (Laufzeitsystem) wird auf

der Basis der anwendungsspezifischen Modelle und einiger anwendungsübergreifender UIMS-Erweiterungen (Bibliothek von Interaktionswerkzeugen für eine Anwendungsklasse) automatisch generiert.

Zu beachten ist, daß in diesem Entwicklungsszenario wesentliche Gestaltungsmerkmale der Oberfläche und der gezeigten visuellen Darstellungen festgelegt werden, bevor konkrete Domänenobjekte durch Berechnungsfunktionen erzeugt werden. Für die Realisierung der hier betrachteten Lokalisierungshandlung ist es z.B. wichtig, daß die präsentierten Plazierungsbereiche nicht übereinander liegen, da sie in diesem Fall nur sehr schwer ausgewählt werden können. Wenn nun die Planung der Perspektive der Darstellung erst zur Laufzeit durchgeführt wird, so liegt vielleicht zunächst eine Menge von Bereichen vor, die sich auch in der Seitenansicht nicht überlappen. Falls keine anderen Einschränkungen die Seitenansicht ausschließen, so könnte sie gewählt werden. Wenn sich später doch eine Menge von Plazierungsbereichen ergibt, die sich in der Seitenansicht überlagern, müßte die Darstellungsperspektive geändert werden. Ein Perspektivenwechsel ist jedoch eine entscheidende Änderung der Darstellung und sollte vermieden werden. Es muß zur *Entwicklungszeit* eine Sicht festgelegt werden, die auch für den ungünstigsten Fall geeignet ist.

Modellbasierte Visualisierungen für eine Benutzungsschnittstelle sind in diesem Entwicklungsszenario für bestimmte Benutzeraufgaben gedacht, d.h. sie müssen so konzipiert werden, daß der Endbenutzer interaktiv Aktionen durchführen kann, durch welche die ihm zugedachten Aufgaben erfüllt werden. Im Entwicklungsszenario aus Abbildung 5 ist der Endbenutzer von XKL ein Kundenberater eines Flugzeugherstellers. Aktionen dieses Benutzers werden durch das Entwicklungsteam mit Hilfe der durch HAMVIS bereitgestellten Aktionenkonzepte formal definiert. Genauer gesagt, es werden die zur Laufzeit möglichen Aktionen des Benutzers modelliert. Wenn also im HAMVIS-Kontext zur Entwicklungszeit von einer Benutzeraktion gesprochen wird, ist immer eine *Handlungsmöglichkeit* bzw. ein *Aktionenschema* gemeint.

Damit Ableitungen über den Inhalt von Visualisierungen möglich werden, sind die durch HAMVIS bereitgestellten Aktionenkonzepte auf einer „tiefen“ Ebene modelliert, z.B.: „Auswahl zur weiteren Bearbeitung“, „Lokalisierung eines räumlichen Objekts“ usw. Die Umsetzung auf eine oberflächennahe Ebene („Anklicken“, „Ziehen-und-Fallenlassen“) erfolgt automatisch. Die Aktionenkonzepte von HAMVIS modellieren *elementare Benutzeraktionen*.

Die Aufgabe des Entwicklungsteams besteht darin, zur Entwicklungszeit der Anwendung durch Komposition von elementaren Benutzeraktionen und Berechnungs- und Speicherfunktionen Schemata für *zusammengesetzte Handlungen* zu modellieren. In dem Beispiel aus Abbildung 1 wird die zur Laufzeit verwendete Oberfläche für eine zusammengesetzte Handlung zur Erzeugung eines neuen Einrichtungsgegenstandes gezeigt. Der XKL-Benutzer wählt ein Prototypobjekt aus. Die möglichen Plazierungsbereiche für ein Prototypobjekt werden daraufhin automatisch berechnet. Der Benutzer wiederum lokalisiert anschließend das neue Kabinenobjekt durch Plazierung in einem der möglichen Plazierungsbereiche. Die Plazierungsinformationen müssen dann im Datenmodell abgespeichert werden. Die Typen der Parameter und Werte der Berechnungs- und Speicherfunktionen werden durch die vorgesehenen elementaren Benutzeraktionen eingeschränkt. Für zusammengesetzte Handlungen dieser Art versucht HAMVIS, aufgrund der Konzepte der Benutzerhandlungen eine oberflächennahe Realisierung zu bestimmen. In dem Beispiel aus Abbildung 1 ist eine Ziehen-und-Fallenlassen-Geste

8. Für die am Entwurf beteiligten Personengruppen wurde in Abbildung 5 jeweils ein Repräsentant als Stellvertreter eingezeichnet. Die Personengruppen müssen nicht disjunkt sein.

adäquat. Durch eine solche Realisierung einer zusammengesetzten Handlung mit einer einzigen Mausgeste sind für den Benutzer die Berechnungsfunktionen transparent. Man kann also auf höherer Ebene auch bei einer zusammengesetzten Handlung von einer *Benutzer*aktion sprechen. Zusammengesetzte Handlungen stellen Handlungsalternativen dar. Im XKL-Beispiel könnte z.B. eine andere zusammengesetzte Handlung die Möglichkeit zur Verschiebung eines Kabinenobjekts durch den XKL-Benutzer modellieren.

Mehrere Handlungsalternativen können durch das Entwicklungsteam zu einer sogenannten *Aktivität* zusammengefaßt werden, wobei für jede Aktivität durch HAMVIS ein separates Interaktionsfenster vorgesehen wird. Mehrere Aktivitäten bilden eine *Applikation*. Durch Definition einer Aktionenzerlegung legt das Entwicklungsteam die *Struktur der Anwendung* fest. HAMVIS stellt eine graphische Oberfläche bereit, mit der eine Aktionenmodellierung und -zerlegung durchgeführt werden kann. Abbildung 6 vermittelt ein Beispiel für eine Aktionenzerlegung der hier als Leitbeispiel betrachteten XKL-Anwendung.

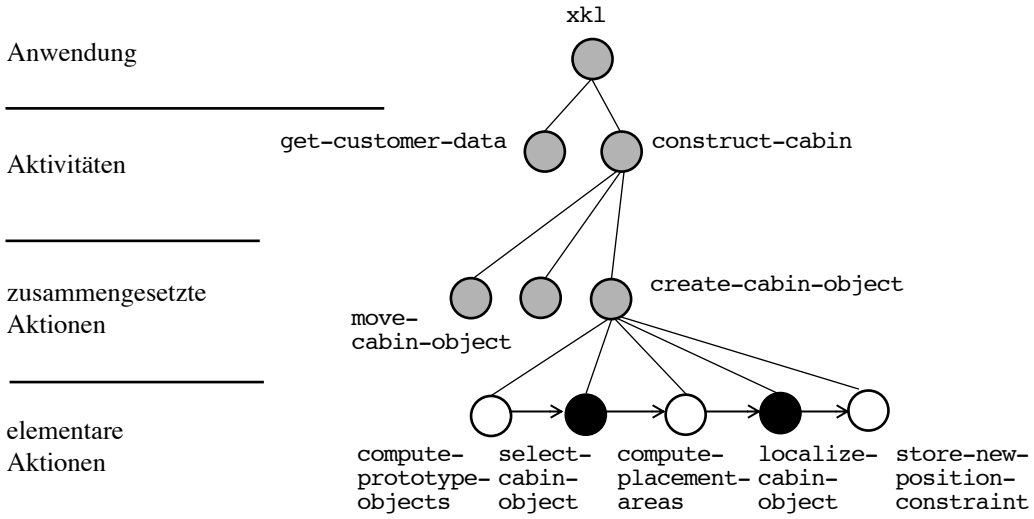


Abbildung 6. Skizze der Aktionenzerlegung zur Modellierung der Struktur der XKL-Anwendung. Benutzeraktionen sind mit schwarzen Kreisen gekennzeichnet. Weiße Kreise markieren automatische Berechnungsfunktionen. Die Abhängigkeiten zwischen Benutzeraktionen und Berechnungsfunktionen sind symbolisch mit Pfeilen angedeutet.

Durch eine anwendungsspezifische Aktionenzerlegung wird durch den Entwickler festgelegt, wann welche Berechnungs- oder Speicherfunktion evaluiert werden muß und wie das, was zur Laufzeit darzustellen ist, ermittelt wird. Aus einer Aktionenmodellierung werden durch formale Verfahren Einschränkungen für die Darstellung von Objekten abgeleitet. Die dem Entwicklungsteam von HAMVIS zur Verfügung gestellten Werkzeuge sind so gestaltet, daß durch HAMVIS aus einer Aktionendekomposition unter Zuhilfenahme eines UIMS automatisch ein Laufzeitsystem erstellt werden kann, wobei jedoch die Modellierung von Aktionen auf einem hohen Abstraktionsniveau erfolgen kann. Durch das Laufzeitsystem wird zur Benutzungszeit der Anwendung die Verwaltung und Darstellung der Daten übernommen.

Eine zentrale Idee von HAMVIS besteht darin, eine Menge von vordefinierten Konzepten zur Modellierung von elementaren Benutzeraktionen (schwarze Kreise in Abbildung 6) bereitzustellen. Die Konzepte der elementaren Benutzeraktionen, die in einer anwendungsspezifischen Aktionendekomposition verwendet werden, bestimmen den *Basisinhalt* der zur Laufzeit zu präsentierenden Visualisierungen. Die in einer Aktionenzerlegung verwendeten elementaren Benutzeraktionen legen fest, *was* in der Oberfläche erscheinen muß. Durch die Aktionenhierarchie und die Konzepte der elementaren Benutzeraktionen werden Einschränkungen dafür definiert, *wie* etwas zu präsentieren ist, damit der Benutzer die Aktionen zur Laufzeit auch tatsächlich durchführen kann. Das Wissen zur Abbildung auf UIMS-Dienste, so daß Benutzeraktionen zur Laufzeit durchführbar sind, wird durch HAMVIS über die Konzepte der elementaren Benutzeraktionen repräsentiert. Mögliche Interaktionsformen zur Umsetzung einer Aktion legen weitere Einschränkungen für die Gestaltung der hierfür benötigten Visualisierungen fest. HAMVIS verwaltet die Informationen über die zu präsentierenden Objekte und die jeweiligen Einschränkungen. Ich möchte noch einmal betonen, daß zur Entwicklungszeit i.a. nur konzeptuelle Informationen über die darzustellenden Objekte bekannt sind, da sie erst zur Laufzeit als Instanzen erzeugt werden. HAMVIS stellt Konstrukte bereit, mit denen diese Informationen repräsentiert werden können.

Aktionen und deren Konzepte definieren, was für die *Ausführung* der Aktionen in einer Visualisierung *mindestens* dargestellt werden muß. Damit für den Anwendungsbenutzer zur Laufzeit eine adäquate Interaktionsform erzielt werden kann, müssen ggf. noch weitere Objekte kommuniziert werden. HAMVIS stellt Modelle für die in der Anwendungsklasse benötigten „Kommunikationsschemata“ und Dialogstrukturen bereit. In dem betrachteten Beispiel aus Abbildung 1 wird z.B. für die Lokalisierungshandlung als räumliches Referenzsystem der Flugzeugrumpf mit Landmarken wie Türen, Cockpitfenster usw. benötigt. Die Aufgabe von HAMVIS besteht auch darin, zur Entwicklungszeit durch formale Schlußfolgerungsprozesse sicherzustellen, daß die erforderlichen Informationen zur Laufzeit aus den Domänenmodellen, die durch das Systementwicklungsteam bereitgestellt werden, abgeleitet werden können.

HAMVIS gestattet dem Entwickler, die Oberflächengestaltung schon zur Entwicklungszeit aus der Kommunikationsperspektive zu betrachten. Damit der Anwendungsbenutzer zur Laufzeit agieren kann, werden ihm die hierzu notwendigen Informationen präsentiert. Die Planung der Gestaltung dieser Kommunikation erfolgt aber zur Entwicklungszeit. Wissen über kohärente Kommunikationsformen in der hier betrachteten Anwendungsklasse wird durch HAMVIS repräsentiert, so daß eine adäquate *Dialogstruktur* aufgebaut werden kann. In einer Dialogstruktur werden die Komponenten der Oberfläche, die Beziehungen zwischen Komponenten sowie die für die Erstellung eines Laufzeitsystems notwendigen Informationen repräsentiert und verwaltet (vgl. Abbildung 5).

Zum Aufbau eines Dialogmodells stellt HAMVIS für den Designer eine weitere interaktive Oberfläche bereit. Mit dieser Oberfläche werden dem Oberflächenentwickler die zur Laufzeit darzustellenden „Informationseinheiten“ (Mengen von Objekten) der Anwendung präsentiert, die aus der vorher erstellten Aktionendekomposition abgeleitet wurden. Nach und nach kann er die Einheiten in das Dialogmodell integrieren. Ein Beispiel für eine solche Einheit ist die Menge der Plazierungsbereiche, die zur Lokalisierung eines Kabinenobjektes möglich sind. Bei der Integration einer Informationseinheit bestimmt HAMVIS, welche Zusatzinformationen (Referenzsysteme, Landmarken) für die Darstellung benötigt werden bzw. wünschenswert sind und welche Präsentationseinschränkungen hierfür „in Kauf genommen“ werden müssen. Für jede Einheit ist bekannt, für welche Aktion sie benötigt wird.

Wenn für die eine Aktion eine Seitenansicht angemessen ist und für eine andere Aktion eine Sicht von oben benötigt wird, so werden hierfür zur Entwicklungszeit zwei verschiedene Teilvisualisierungen im Dialogmodell „eingeplant“. Zur Laufzeit werden in getrennten Unterfenstern jeweils Darstellungen mit verschiedener Perspektive gezeigt. Es kann allerdings auch der Fall sein, daß eine einzige Visualisierung geeignet ist, mehrere Aktionen zu unterstützen (z.B. die Lokalisierung von Kabinenobjekten und auch deren Verschiebung). Es ergeben sich eventuell mehrere Varianten zur Dialogstrukturierung, die von HAMVIS bestimmt und verwaltet werden. Die von HAMVIS bereitgestellte Entwicklungsoberfläche für die Organisation der Dialogstruktur dient dem Oberflächenentwickler dazu, mögliche Varianten zu bewerten und ggf. zu verwerfen. Es wird durch das Dialogmodell weiterhin festgelegt, welches Domänenwissen zu Darstellungszwecken vom Entwicklungsteam zu repräsentieren ist und wie Domänenobjekte strukturiert werden müssen, damit die für die Durchführung der Aktionen benötigten Visualisierungen generiert werden können.

Die in der Dialogstruktur vermerkten Einheiten haben jeweils eine bestimmte kommunikative Funktion und stehen in bestimmten Beziehungen zueinander (Hauptobjekte, Zusatzobjekte, manipulierbare Objekte). Durch diese *Charakterisierungen* ergeben sich (weitere) Einschränkungen für die Darstellungsform und die zur Darstellung verwendeten Zeichenattribute wie z.B. Farbe, Strichdicke usw. Verschiedene Informationseinheiten in der Dialogstruktur, die durch gleiche Darstellungscharakterisierungen ausgezeichnet sind, sollten mit gleichen Zeichenattributen dargestellt werden. HAMVIS überträgt hier die Idee der Markierungen (markups), die im Bereich der Textsatzsysteme für die Beschreibung bzw. die Charakterisierung von Textteilen entwickelt wurde, auf den Graphikbereich. Ähnlich wie in Textverarbeitungssystemen eine Markierung z.B. „Section“ vergeben wird, sieht HAMVIS eine Markierung „Referenzrahmen“ für bestimmte Objekte vor. Zu beachten ist, daß der Markierungstyp für eine Informationseinheit nicht vom Konzept der beschriebenen Objekte abhängt, sondern sich aus der kommunikativen Stellung in der Dialogstruktur ergibt.

Für jeden Markierungstyp müssen geeignete Zeichenattribute bestimmt werden. Die Zeichenattribute für verschiedene Markierungen sind jedoch nicht voneinander unabhängig. Wenn der Hintergrund weiß ist, so können zur Darstellung von wichtigen Objekten dunkle Farben, von Zusatzobjekten hellere Farben gewählt werden. Ist der Hintergrund jedoch schwarz, lassen sich dunkle Farben kaum noch verwenden usw. Aus den im Dialogmodell eingetragenen Informationen über Darstellungseinschränkungen und kommunikative Funktionen von Dialogstruktureinheiten berechnet HAMVIS geeignete Standard-Zeichenattribute. Die Standardwerte können durch den HAMVIS-Benutzer aber noch nach seinen Wünschen angepaßt werden. Durch den Mechanismus der Markierung braucht aber die Änderung nur an einer Stelle vorgenommen zu werden. Über das Konzept der Markierung wird also – ähnlich wie bei Textsatzsystemen – auch in Graphiken eine konsistente Verwendung von Zeichenattributen in der gesamten Anwendung gefördert.

Eine Dialogstruktur mit entsprechenden Markierungen für die Darstellungsattribute bezeichne ich als Präsentationsstruktur. Zusammen mit den auf der Basis der Spezifikation in der Aktionenzerlegung implementierten Berechnungs- und Speicherfunktionen und einem UIMS kann aus der Präsentation ein Laufzeitsystem generiert werden, so daß sich der Oberflächenentwickler nicht mit den Details der Programmierung von Fensterschnittstellen beschäftigen muß.

1.4 Der wissenschaftliche Beitrag dieser Arbeit

Die Erstellung von interaktiven Oberflächen mit problemnahen, aufgabenorientierten Visualisierungen ist – wie schon eingangs erläutert – vielfach mit hohen Kosten verbunden. Nicht selten basiert die Gestaltung von graphischen Oberflächen auf Ad-hoc-Entscheidungen, die weitaus häufiger durch Berücksichtigung der Erstellungskosten geprägt sind als auf eine intensive Betrachtung der für den Benutzer und dessen Aufgaben notwendigen Inhalten zurückgeht. Der vielfach beschrittene Weg, eine Bibliothek von wiederverwendbaren Moduln einzurichten, hat sich als nicht unproblematisch erwiesen, da vielfach Moduln nicht kompatibel sind und nicht ohne Schwierigkeiten kombiniert werden können. Obwohl es möglich ist, Interaktionsbausteine für verschiedene Anwendungsfälle in einer Art Werkzeugkasten (toolbox) zugängig zu machen, ist jedoch auch das Finden von geeigneten Bausteinen und deren Integration ein großes Problem. Es ist vorteilhaft, verschiedene „Dienste“ in ein Rahmensystem zu integrieren, so daß bei der Entwicklung einer Anwendung mit einer interaktiven Oberfläche vielfältige Entwurfsentscheidungen koordiniert werden können.

Diese Arbeit definiert mit HAMVIS die Konzeption eines Rahmensystems zur methodischen Entwicklung von aufgabenorientierten, interaktiven Visualisierungen für Benutzungsoberflächen. Das Problem bei der Definition eines Rahmensystems dieser Art besteht darin, geeignete Teilprobleme zu definieren und so zu lösen, daß am Ende die Teillösungen und die jeweils zur Problemlösung verwendeten Wissensbasen nahtlos ineinandergreifen. Es wird gezeigt (siehe auch [213]), wie generisches Wissen über Handlungen (z.B. zur Manipulation von räumlichen Objekten) so repräsentiert werden kann, daß bezüglich einer Aktionendekomposition, die durch das Systementwicklungsteam für eine spezielle Anwendung erstellt wird, Visualisierungsinhalte bestimmt werden können. Es wird weiterhin aufgezeigt, daß wesentliche Gestaltungsmerkmale der Anwendung *schon zur Entwicklungszeit* einer Anwendung systematisch festgelegt werden können, so daß ein effizientes Laufzeitsystem generiert werden kann. Da zur Entwicklungszeit konkrete Objekte nicht in jedem Fall verfügbar sind, müssen konzeptuelle Informationen verarbeitet werden. Die Arbeit zeigt, daß mit Hilfe von Beschreibungslogiken die notwendigen Ableitungen zur Visualisierungsgestaltung formalisiert werden können.

Informationen über die Art der Darstellung von Domänenobjekten werden als *Einschränkungen* in einer Dialogstruktur gesammelt. Wie schon bei der Aktionendekomposition muß auch hier ein *Wechselspiel zwischen generischem Wissen und anwendungsspezifischen Modellen* behandelt werden. HAMVIS verwendet beschreibungslogische Repräsentationsformen auch für die Definition von *generischen Dialogschemata für graphische Anwendungen*. Notwendige und wünschenswerte Bestandteile der Dialogstruktur werden aus den Vorgaben formal abgeleitet. Durch Anforderungen der Präsentation wird festgelegt, wie *domänenspezifische Modelle zur Repräsentation von Anwendungswissen* (Datenmodelle und Funktionen) gestaltet werden müssen.

Einschränkungen für die Darstellung definieren einen Variantenraum für die Oberflächenstrukturierung. Die von HAMVIS bereitgestellten Wissensquellen (siehe auch [214]) strukturieren diesen Variantenraum. Der Oberflächenentwickler kann Varianten für die Oberflächengestaltung aus der Perspektive der Kommunikationstheorie bewerten und mögliche Varianten schon im Vorwege verwerfen. HAMVIS stellt für den Oberflächenentwickler interaktive Oberflächen zum *inkrementellen Aufbau und zur Inspektion von Dialogmodellen bzw. Dialogmodellvarianten* bereit. Durch die Explizierung der verschiedenen Einflüsse in Form von Modellen können Entwurfsalternativen gete-

stet werden, ohne (in doch recht aufwendiger Weise) Codeteile in einer Programmiersprache anpassen zu müssen.

Gegenüber bestehenden Arbeiten zur modellbasierten Generierung von Oberflächen setzt HAMVIS einen anderen Schwerpunkt. Zum einen werden Visualisierungen auf der Basis von geometrischen Modellen erzeugt und nicht durch Piktogramme. Zum anderen werden nicht nur explizite Modelle verwendet, die dann von Werkzeugen (tools) manipuliert werden, sondern die zur Visualisierungsgestaltung nötigen Schlüsse werden über beschreibungslogische Repräsentationskonstrukte als logische Ableitungen formuliert. Durch die durchgängige Verwendung einer Beschreibungslogik kann zudem erreicht werden, daß wesentliche Aspekte bei der Realisierung von Teilschritten als wissensbasierte Ableitungen formuliert werden können, die das Verhalten des Rahmensystems erläutern. Für die Realisierung von automatischen Teilschritten bei der Visualisierungsentwicklung können ausgetestete beschreibungslogische Deduktionsalgorithmen zum Einsatz kommen, so daß nicht jeder Teilschritt als separates Werkzeug neu implementiert werden muß. Es sind auch Aussagen über Berechenbarkeits- und Komplexitätsaspekte möglich. Mit der entwickelten Prototypimplementation für HAMVIS wurde das als Grundlage verwendete beschreibungslogische Repräsentationssystem CLASSIC [25] um objektorientierte Mechanismen (generische Funktionen und Methoden) erweitert (siehe [215]). Durch die Erweiterungen wurde die softwaretechnische Grundlage für die nahtlose Integration von logischen Modellen in die Anforderungen einer bestehenden objektorientierten UIMS-Umgebung geschaffen.

Inspiriert durch IMMP-Systeme wird die Gestaltung von Visualisierungen auch zur Entwicklungszeit aus einer kommunikationsorientierten Sicht betrachtet. Dadurch wird erreicht, daß die kommunikative Funktion von Teilkomponenten explizit gemacht wird, so daß die Idee einer Markierung (markup) im Sinne von Textverarbeitungssystemen auch auf die Definition von Visualisierungen und ihre Teilkomponenten angewendet werden kann. Das bei der Textverarbeitung bewährte Prinzip der Trennung von Inhalt und Form kann durch die explizite Modellierung von Benutzeraktionen, die Modellierung der Dialogstruktur und der Markierungen in der Präsentationsstruktur auch bei der Generierung von visuellen Oberflächen zum Einsatz kommen.

HAMVIS verbindet Arbeiten aus den Bereichen Wissensrepräsentation/KI, KI-Softwaretechnik, Mensch-Computer-Interaktion, Intelligente Multimedia-Präsentationssysteme sowie Benutzungsschnittstellen-Entwicklungsumgebungen und zeigt die Querbezüge auf.

1.5 Der weitere Aufbau dieser Arbeit

Nach einer Motivation der Konzeption des HAMVIS-Rahmensystems erfolgt in Kapitel 2 eine Diskussion der relevanten Arbeiten aus folgenden Forschungsbereichen: Werkzeuge und Unterstützungssysteme zur Entwicklung von Benutzungsschnittstellen, Mensch-Computer-Interaktion, Graphische Kommunikation und Intelligente Multimedia-Präsentationsplanung. Es wird dargelegt, welche Forschungsergebnisse für die Gestaltung von HAMVIS maßgeblich waren und wie die bisherigen Arbeiten integriert und erweitert werden.

Kapitel 3 beschreibt die innerhalb des HAMVIS-Szenarios zu verwendenden objektorientierten und beschreibungslogischen Repräsentationssysteme für die Modellierung von Aktionenkonzepten. Insbesondere werden die in dieser Arbeit entwickelten Erweiterungen zur Unterstützung der Visualisierungsgenerierung und Anwendungsentwicklung aufgezeigt. Dabei werden Kenntnisse in

objektorientierter Programmierung (aus der CLOS-Perspektive) sowie Kenntnisse der Theorie der Beschreibungslogiken vorausgesetzt.

Der Aufbau einer Aktionenmodellierung und der interaktive Aufbau von Dialogstrukturvarianten wird am Beispiel von XKL vorgestellt. Anhand der XKL-Anwendung wird beschrieben, wie anwendungsspezifische Modelle in das HAMVIS-Rahmenwerk eingegliedert werden. Zur Illustration der Visualisierungskomposition werden Teilkomponenten der Benutzeroberfläche des HAMVIS-Rahmensystems skizziert. Weiterhin werden die zur Visualisierungsgenerierung von HAMVIS verwendeten internen Repräsentationsformen für räumliche Objekte, Benutzerhandlungen und Dialog- bzw. Diskursstrukturen sowie die zur Verwaltung von Abhängigkeiten zwischen Präsentationsparametern (z.B. Zeichenattributen) vorgesehenen Modelle erläutert. Inferenzen von HAMVIS schränken den möglichen Gestaltungsraum für Darstellungen ein. Das Kapitel zeigt die internen Ableitungen und erläutert die Präsentation der Ergebnisse innerhalb der HAMVIS-Benutzungsoberfläche.

In Kapitel 4 werden die in dieser Arbeit vorgestellten Methoden zur aufgabenorientierten Visualisierungsgenerierung für Benutzungsschnittstellen zusammengefaßt und bewertet. Die Arbeit schließt mit einem Ausblick für weitergehende Forschungsarbeiten.

2
Relevante Forschungsarbeiten

Aus der Einleitung wurde deutlich, daß die Entwicklung von interaktiven Visualisierungen nicht losgelöst von der Entwicklung der gesamten Benutzungsschnittstelle eines Programmes betrachtet werden kann. Benutzungsschnittstellen wiederum sind in ein Gesamtsoftwaresystem einzubetten, so daß auch Software-Engineering-Gesichtspunkte zu berücksichtigen sind. Einerseits muß der Code von Anwendung und Schnittstelle getrennt werden, um z.B. verschiedene Schnittstellenbibliotheken oder ein UIMS (user interface management system) zu unterstützen.[9] Andererseits ist inzwischen die Einsicht weitverbreitet, daß es unmöglich ist, qualitativ hochwertige Schnittstellen (und damit auch Anwendungssysteme) zu entwickeln, wenn Schnittstellen nachträglich auf bestehende, aus technikzentrierter Sicht entworfene Anwendungen aufgesetzt werden. Die klassische Dekomposition eines Anwendungsprogramms (bekannt als Seeheim Modell [247]) in Benutzungsschnittstelle, Dialogkontrolle und automatischen Programmkomponenten („eigentliche Anwendung“) und die Idee einer strengen Trennung der Entwicklung von Benutzungsschnittstelle und Anwendung wurde von vielen Autoren kritisiert (siehe z.B. [314] und [121]). Es wird heutzutage mehr als ein logisches Modell für das Systemdesign angesehen und nicht als Architekturkonzept für die Entwicklung von interaktiven Systemen wie etwa XKL betrachtet. Das Problem dieses Ansatzes ist, daß die für direktmanipulative Interaktionsformen notwendige Rückkopplung auf Anwendungsinformationen nicht adäquat unterstützt wird. Im Gegensatz zu formularorientierten (bzw. maskenbasierten) Schnittstellen rufen graphische Benutzungsschnittstellen bei der Interaktion des Benutzers Anwendungsfunktionen auf, um z.B. den Zustand der Anwendung widerzuspiegeln. Diese Art der direkten graphischen Rückkopplung der Wirkung einer Interaktionshandlung (semantic feedback) muß durch Anwendungsfunktionen und durch entsprechende Datenstrukturen unterstützt werden. Es kann der Fall sein, daß die Datenmodelle der Anwendung in besonderer Weise strukturiert werden müssen, damit die geforderten Werte auch zum rechten Zeitpunkt in Form von adäquaten Objekten geliefert werden können.

Bei der in Abbildung 1 betrachteten Visualisierung muß beispielsweise während der Interaktion eine Berechnungsfunktion gestartet werden, um die möglichen Plazierungsbereiche zu bestimmen. Sie werden benötigt, um die Zielflächen für die Ziehen-und-Fallenlassen-Geste (drag-and-drop gesture) zu bestimmen. Die Zielflächen werden entsprechend hervorgehoben, sobald die Maus auf eine der Flächen positioniert wird.

Nicht zu vernachlässigen ist auch der Einfluß von Oberflächenaspekten auf die Gestaltung von Anwendungsfunktionen und -datenstrukturen. Ein Beispiel hierfür ist die Verschiebung eines Objekts. Bei einer Objektverschiebung muß der mögliche Verschiebungsbereich bekannt sein. Es ist nicht sinnvoll, dem Anwendungsbenutzer zu gestatten, ein Objekt z.B. aus der Flugzeugkabine herauszubewe-

9. In der Literatur werden verschiedene technische Lösungen für ein Kopplungsprotokoll vorgestellt: Smalltalks Model-View-Controller-Architektur [165], Dämonen [111], Aktive Werte (engl. active values in LOOPS [22] oder KEE [136]), Beschränkungen (constraints) [28] und verschiedene andere Ansätze [346]. Zu diesem Thema sind auch die Arbeiten von Myers et al. [222] [223] [225] [226] und Szekely [308] relevant.

gen. Jede gewählte Position im Verschiebungsbereich sollte zulässig sein, d.h. eine interaktive Verschiebung innerhalb des möglichen Bereichs sollte bei der Abspeicherung im Datenmodell nicht zu einem Fehler führen.

Ein möglicher Verschiebungsbereich kann entweder statisch dargestellt oder dynamisch eingeblendet werden, wenn der Benutzer ein Objekt anklickt (siehe Abbildung 14).

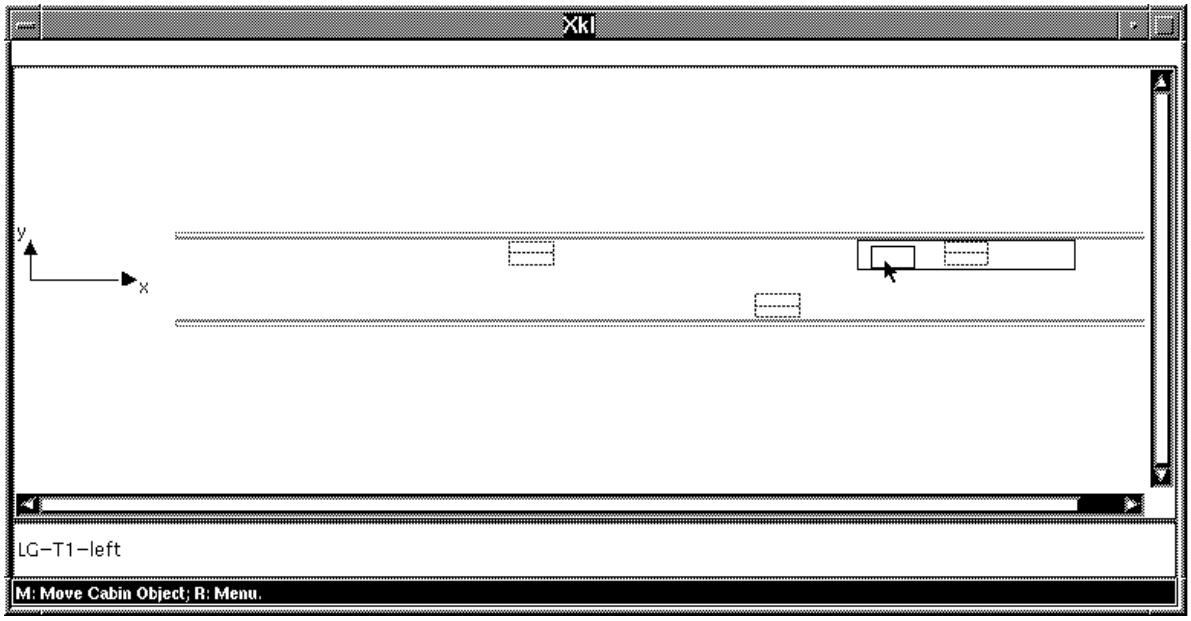


Abbildung 7. Verschiebung des rechten oberen Kabinenobjekts (hier in der Schwarz-Weiß-Darstellung gestrichelt dargestellt). Das Objekt kann nur innerhalb des angezeigten Positionierungsrechtecks bewegt werden. Der Mauszeiger kennzeichnet zusammen mit dem zugehörigen schwarzen „Verschiebungsrechteck“ die neue Position.

Während der Verschiebeaktion (dragging) bewegt sich das zu verschiebende Objekt (grau) mit. Es kann allerdings nicht über den Verschiebungsbereich (schwarze, dicke Linie) hinausbewegt werden. Eine willkürliche Bewegungsmöglichkeit des Objekts bei eventueller anschließender Fehlermeldung ergibt eine inadäquate Interaktionsform. Der mögliche Verschiebebereich muß also spätestens beim Anklicken des Kabinenobjekts als explizites Objekt zur Darstellung des „Anwendungszustandes“ bereitgestellt werden. Werden beim Systementwurf die Anforderungen einer graphischen Interaktion nicht berücksichtigt, so sind z.B. Informationen, die notwendig sind, um eine Rückkopplung des Anwendungszustandes zu geben, im ungünstigen Fall auf verschiedene, nicht zusammenhängende Datenstrukturen verteilt. Die Definition der Dialogsteuerung beeinflußt also die Anwendungsschnittstelle (application interface), die wiederum Auswirkungen auf den Entwurf der Algorithmen und Datenstrukturen der Anwendung hat. Ein unabhängiger Entwurf der beiden Teile ist kaum möglich.

Wird die Dialogsteuerung primär durch Algorithmen der Anwendung bestimmt, ist die Gefahr groß, daß für den Benutzer eine undurchsichtige Interaktionsform entsteht, insbesondere wenn der Benutzer nicht „agieren“ kann, sondern nur „Daten eingibt“. In KI-Anwendungen wie z.B. Expertensystemen trat genau dieses Problem auf. Unmotivierte „Fragen“, die durch die Abarbeitungsreihenfolge der im Expertensystem verwendeten Regeln für den Benutzer „zufällig“ aneinandergereiht erschienen, sind ein Beispiel für einen solchen Effekt. Mit gewissen Einschränkungen müssen Daten im Vorwege

erfragt werden, um strukturierte Interaktionsformen zu realisieren (außer bei komplexen Abhängigkeiten). Entwicklungssysteme für Expertensystemoberflächen enthalten entsprechende Werkzeuge, mit denen dieses realisiert werden kann. Ein Beispiel ist das von Löwgren vorgestellte anwendungsübergreifende System „Ignatius“ [178]. Die Architektur von Ignatius sieht als Teilkomponenten Modelle des Glaubens und Wissens (auch Überzeugungsmodelle genannt) vor, die dazu verwendet werden, sicherzustellen, daß ein Datum, das einmal durch den Benutzer mit Hilfe der Oberfläche eingegeben wurde, nicht erneut erfragt wird. Um bestimmte Informationen von Benutzer zu erhalten, gestattet Ignatius dem Schnittstellenentwickler, bestimmte „Teilfragen“ a priori zusammenzufassen. Als Ergänzung zu einem Benutzermodell stellt Ignatius ein Systemmodell bereit.

Wenn nun durch den internen Kontrollfluß des Expertensystems eine der Teilfragen beantwortet werden muß, so werden die jeweils zusammengefaßten Fragen gleich mit gestellt. Ein Dialog wird durch Eingabeschemata bestimmt (input domain plans). Für Eingabeschemata werden durch den Schnittstellenentwickler Eingabemasken mit einem interaktiven Schnittstellenbaukasten (interface builder) erstellt. „Fragen“ werden also durch Präsentation von Standard-Interaktionsbausteinen wie Ankreuzlisten, Schalter usw. gestellt. Dadurch wird dem Benutzer auch die interaktive „Antwortmöglichkeit“ gegeben. Ignatius ist dafür ausgelegt, konsultative Dialogformen für diagnostische Expertensysteme mit Standard-Interaktionsbausteinen (insbesondere Textfeldern, Schaltflächen, Ankreuzlisten usw.) zu unterstützen. Eine unabhängige Entwicklung von Oberfläche und Wissensbasis ist in dieser Anwendungsklasse ebenfalls kaum möglich. Forschungsarbeiten im Bereich der Erklärungsgenerierung für Expertensysteme mit Hilfe von natürlichsprachlichen Schnittstellen haben zudem gezeigt, daß es beinahe unmöglich ist, ein System zur Generierung natürlicher Sprache an eine Wissensbasis anzukoppeln, die dafür nicht speziell konstruiert worden ist.[10]

Die Quintessenz ist, daß (auch) bei einem wissenbasierten System nicht nur das statische Modell (die Wissensbasis) im Zentrum des Systementwurfs stehen darf. Für ein System, das erfolgreich einsetzbar sein soll, muß auch die Problemlösungsstrategie (z.B. eines Expertensystems) auf die Belange einer interaktiven Schnittstelle abgestimmt werden. Im allgemeinen ist es nachteilig, wenn der Entwurf der Schnittstelle durch implementationsnahe, kontrollflußorientierte Gesichtspunkte von automatischen

---

10. Siehe hierzu die Arbeiten zu XPLAIN [303] [304] und dem Nachfolgesystem EES [217] [229]. Weiterhin sind hier die Forschungsarbeiten zu XTRA [2] und WISBER [343] zu nennen. Eine tiefere Diskussion dieses Themas findet man in [160]. Ansätze jüngerer Zeit untersuchen spezielle Methoden zur Entwicklung von KI-Systemen wie z.B. KADS auf eine Eignung zur Unterstützung einer Erklärungsgenerierung [218]. Sogenannte Expertensysteme der zweiten Generation (Second Generation Expert Systems) zeichnen sich dadurch aus, daß die Erklärungsgenerierung ein integraler Bestandteil der Gesamtarchitektur ist [305].

Eine große Nachfrage nach einer Erklärungskomponente deutet allerdings auf eine starke Intransparenz des Anwendungssystem hin, die vielleicht durch ein anderes Anwendungsdesign vermieden werden könnte (siehe die Argumentation von Holz in [130]). Zu beachten ist auch das Wechselspiel zwischen Anforderungen von Problemlösungsprozessen und Erklärungsgenerierungsaspekten. Zur Unterstützung einer Erklärungsgenerierung könnten kompliziertere Modelle notwendig sein, was sich durch eine erhöhten Laufzeit der Berechnungsprozesse ungünstig bemerkbar machen kann. Für eine interaktive Schnittstelle können gewisse Laufzeiteinbußen inakzeptabel sein.

Ein weiterer wichtiger Punkt ist, daß einfache und verständliche Erklärungen nicht in jedem Fall ohne großen Aufwand gegeben werden können. Betrachtet man zum Beispiel ein Simulationssystem, das mit Hilfe von Finite-Elemente-Modellen das Verformungsverhalten einer Automobilkarosserie simuliert, so wäre es müßig, nach einer Erklärung der Position eines bestimmten kleinen Elementes am Ende eines Crash-Simulationsexperiments zu fragen.

Berechnungsfunktionen beeinflußt wird. Im Gegenteil, wie sich aus der obigen Diskussion ergibt, müssen Berechnungs- und Speicherfunktionen neben der Pflege der für die internen Algorithmen benötigten Repräsentationsformen auch Datenstrukturen und -objekte bereitstellen können, die für die Präsentation der Berechnungsergebnisse an der Oberfläche benötigt werden. Wichtig ist, daß dieses durch die zugrundeliegenden Entwicklungswerkzeuge und eine angepaßte Entwurfsmethodik unterstützt wird.

Die Einflußnahme von Oberflächenaspekten auf die Definition von Datenstrukturen und Algorithmen heißt jedoch nicht, daß Dialoge nicht durch Berechnungsprozesse initiiert werden können (mixed initiative dialogs). Dieses sollte jedoch auf der grobkörnigeren Ebene der abgrenzbaren Teilaktivitäten erfolgen und nicht auf der Ebene der Regelverkettung (wenn z.B. bei einer Rückwärtsverkettung nicht herleitbare Vorbedingungen auftreten).

Zusammenfassend läßt sich sagen, daß aus Sicht der Gesamtanwendung interne Modelle für Speicherungs- und Berechnungsprozesse auf die Belange der Schnittstelle abgestimmt werden müssen. Andererseits muß bei der Entwicklung der Schnittstelle berücksichtigt werden, daß bestimmte Berechnungsprozesse eine hohe Zeit- und Speicherkomplexität haben. Insbesondere Rückkopplungen des Anwendungszustandes in Schnittstellen mit visuellen Darstellungen sind nicht in jedem Fall einfach zu realisieren. Eine Trennung von Anwendung und Schnittstelle ist jedoch aus Gründen der Arbeitsteilung zu unterstützen. Die Architektur und Systementwicklungsmethodik von HAMVIS versucht, diesen Einsichten durch explizite Modellierung von Benutzeraktionen und deren Kopplung mit Berechnungs- und Speicherfunktionen gerecht zu werden, indem Visualisierungen nicht a posteriori auf vorgefertigte Anwendungsfunktionen aufgesetzt werden, sondern die Modellbildung der Anwendung auf die Erfordernisse der Visualisierungsgenerierung abgestimmt wird und umgekehrt.

Die relevanten Forschungsarbeiten aus verschiedenen Bereichen können in folgende Hauptkategorien eingeteilt werden:

- *manuelle* Schnittstellenkonstruktion (mit Aspekten der Softwaretechnologie, der Mensch-Computer-Interaktion und der kognitiven Psychologie).
- *automatische* und *dynamische* Präsentationsgenerierung (mit Schwerpunkt auf multimodaler Kommunikationstheorie, Wissensrepräsentation und -verarbeitung)

Bei den letztgenannten Ansätzen ist die Kommunikationsperspektive vorherrschend. Es werden von einem *(Software-)Agenten* „Äußerungen“ generiert, die dazu dienen, seine kommunikativen Ziele zu erfüllen, wobei auch die Historie der Kommunikationsakte des menschlichen Kommunikationsteilnehmers herangezogen wird.

Der manuelle Ansatz wird durch Schnittstellen-Entwicklungssysteme (UIMS) unterstützt. Aus Sicht des Schnittstellendesigners ist ein UIMS eine Menge von Werkzeugen und Bibliotheken, die ihn bei der Erstellung eines interaktiven Systems zur Entwicklungszeit unterstützen. Ein UIMS stellt außerdem ein Laufzeitsystem bereit, das zur Benutzungszeit der Anwendung Dienste bereitstellt zur Bestimmung des Layouts von Teilfenstern, zum Zeichnen und Neuzeichnen von graphischen Objekten, zum Rollen von Fenstern, zur Behandlung von Eingabegeräten (z.B. Maus), zur Eingabe von Werten, zur Organisation einer interaktiven Hilfe usw. UIMS-Konzepte sind inzwischen sehr mächtig geworden; es ist ein eigenes Forschungsgebiet entstanden. Mit der Betrachtung eines konkreten UIMS wird ein Überblick über den Stand der Kunst in diesem Bereich und über die wesentlichen Konzepte

gegeben, die zum Verständnis der technischen Realisierung des von HAMVIS zu generierenden Laufzeitsystems notwendig sind.

Obwohl die Verwendung eines bestimmten UIMS die Gestaltung einer Schnittstelle schon entscheidend prägen kann, sind zur Entwicklungszeit der Oberfläche insbesondere zur Gestaltung der anwendungsspezifischen Visualisierungen noch sehr viele verschiedene Entscheidungen zu fällen und diverse Gestaltungsmöglichkeiten für den Oberflächenentwickler offen. Das Feld der Mensch-Computer-Interaktion, das in diesem Kapitel kurz skizziert wird, liefert den theoretischen Rahmen für die Arbeit des Oberflächendesigners.

Eine etwas andere Sicht auf die Konzeption der Mensch-Computer-Interaktion wird von intelligenten Multimedia-Präsentationssystemen (IMMPS) vertreten. Ein IMMPS kommuniziert als Agent direkt mit dem Benutzer. Ausgaben werden auf der Basis von vorgegebenen Modellen über den Informationsbedarf und den Wissensstand des Benutzers generiert. Die Benutzermodelle können ggf. während des Dialoges aktualisiert werden. Ausgaben werden als Handlungen im Sinne der Sprechakttheorie (Cohen und Levesque [54], Cohen und Perrault [55]) betrachtet, die zur Erfüllung bestimmter kommunikativer Ziele geeignet sind. Kommunikative Ziele leiten sich, der Maxime des rationalen Handelns folgend, aus generellen Zielen des Agenten her. Die Planung von Ausgaben (Äußerungen) erfolgt durch Schlußfolgerungsprozesse über den explizit repräsentierten kommunikativen Zielen. „Konstruktionspläne“ für eine Präsentation enthalten explizite Angaben über den kommunikativen Zweck von Präsentationskonstituenten in Hinblick auf die Erreichung von kommunikativen Zielen. In diesen Systemen spielt ein menschlicher Designer einer generierten Präsentation keine Rolle, d.h. eine Entwurfs- oder Nachbearbeitungsphase ist vom Prinzip her nicht vorgesehen.

Die Konzeption von HAMVIS geht von der These aus, daß es vorteilhaft ist, auch im Rahmen eines *Entwurfsphasen*-Unterstützungssystems für die Komposition von Visualisierungen die explizite Repräsentation der kommunikativen Funktion von Teildarstellungen innerhalb eines Dialogmodells zu betrachten.

Folgende Anforderungen an die zu generierenden Visualisierungen, die durch die menschliche Informationsaufnahme bedingt sind, unterstützen diese These:

- Ein wichtiger Punkt der Darstellungsqualität ist *visuelle Stabilität*. Drastische Veränderungen der Darstellung (z.B. Wechsel des Layouts der Teilfenster, häufiges Erscheinen von neuen Fenstern) sind möglichst zu vermeiden. Eine vollständige Änderung der Erscheinung der Darstellung sollte nur bei einem Wechsel der Arbeitsaktivität vorgenommen werden. Weiterhin sollte die Anzahl der Teilfenster minimiert werden. Der rhetorische Status eines Teilfensters (z.B. Hauptaktionsfenster, Fenster für wechselnde Zusatzinformationen, Werkzeugkiste, Palette) sollte innerhalb einer Aktivität beibehalten werden (rhetorische Stabilität). Zeichenattribute von gleichen graphischen Objekten, die innerhalb verschiedener Fenster einer Anwendung dargestellt werden, sollten bei jeweils gleicher rhetorischer Funktion identisch sein.
- Sobald ein Objekt dargestellt wurde, sollte es nicht ohne Grund wieder gelöscht werden, erneut erscheinen (ebenfalls ohne erkennbaren Grund), wieder verschwinden usw. (*Prinzip der Inhaltsstabilität*). Wenn Objekte erscheinen oder verschwinden sollen, bieten sich Techniken zur Konditionierung an. Bevor ein Objekt erscheint, wird ein geeignetes Anzeichen dargestellt. Ein Beispiel wären sich vergrößernde Rechtecke, die von einer Quelle zu dem Zielort des neu zu präsentieren-

den Objekts wandern (zooming rectangles, angewendet z.B. beim Finder™ des Macintosh™). Anzeichentechniken für das Verschwinden von Objekten sind ebenfalls gebräuchlich (zooming rectangles oder „Auflösen in Atome“).

- Der Grad der Modellierungsabstraktion einer Visualisierung sollte für ein Teilfenster während der Arbeitsaktivität gleich bleiben (*Prinzip der Abstraktionsstabilität*). Details können jedoch z.B. auf expliziten Benutzerwunsch eingeblendet werden.

Die Betrachtung dieser Prinzipien macht deutlich, daß sich graphische Darstellungen nicht unmotiviert ändern dürfen. Änderungen der Darstellungen zur Laufzeit sind jedoch nicht grundsätzlich zu vermeiden. Ziel der Konstruktion in der XKL-Anwendung ist es gerade, die „Welt“ so zu verändern, daß ein Zielzustand erreicht wird. Unproblematisch ist es noch, wenn sich eine Darstellungsänderung direkt als Folge einer Aktion des Benutzers ergibt (exekutive Interaktionsform z.B. Löschen oder Verschieben eines Objekts). Änderungen, die sich aus Berechnungsfunktionen ergeben, sind jedoch schwieriger zu vermitteln. Die Berechnungsfunktionen sollten zumindest explizit durch den Benutzer angestoßen werden (deskriptive Interaktionsform). Wenn aus inhaltlichen Gründen eine Änderung der Darstellung vorzunehmen ist, so gibt es vielfach auf der Präsentationsebene geeignete kausale Anzeichen, um eine Änderung kommunikativ umzusetzen.

Voraussetzung für die Realisierung von Stabilität ist die Betrachtung der Gesamtanwendung und aller durch die Gesamtanwendung zu unterstützenden Benutzeraktionen während der *Entwurfsphase* der Anwendung. Es gibt gewisse Inhalte, die kommuniziert werden müssen, wobei gewisse graphische Techniken zur Verfügung stehen. Überlegungen dieser Art machen deutlich, daß es sinnvoll ist, die Generierung von Visualisierungen im Rahmen von Benutzungsschnittstellen *zur Entwicklungszeit* aus der Perspektive der Kommunikationstheorie zu betrachten. Dem Benutzer der Anwendung *zur Laufzeit* hingegegen braucht die Deutung der Oberfläche aus der Kommunikationsperspektive gar nicht bewußt zu sein. Der Übergang zu einer kommunikationsorientierten Perspektive zur Laufzeit kann bei graphischen Oberflächen als Übergang auf eine Metaebene verstanden werden, die jedoch für die „normale“ Arbeit nicht relevant ist. Zur Entwicklungszeit ist es m.E. jedoch notwendig, auch auf der tiefen Ebene formale Strukturen sowohl für die Deutung der Benutzeraktionen als auch für die Repräsentation des Dialogmodells zu definieren.

In der Konzeption von HAMVIS werden die Entwurfszeit-Modelle zur Dialoggestaltung (automatisch) in UIMS-Konstrukte übersetzt. Zur Laufzeit wird eine Anwendung also durch konventionelle UIMS-Abstraktionen „betrieben“. Das heißt jedoch nicht, daß ich die Anwendung von agentenorientierten Ansätzen, die eine „Kommunikationsplanung“ direkt zur Laufzeit durchführen, nicht für sinnvoll halte. Ich möchte in dieser Arbeit jedoch versuchen, Entwurfszeit und Laufzeit von Visualisierungen strikt zu trennen. Ob eine Anwendung als Agent angesehen werden kann oder nicht, hängt u.a. von der Konzeption des UIMS ab, welches bei einem agentenorientierten Ansatz immer mächtiger werden kann. Entwicklungszeit-Entscheidungen über die Systemgestaltung sind in jedem Fall zu fällen. Agentenorientierte Laufzeitsysteme befinden sich zur Zeit noch im Forschungsstadium, HAMVIS greift daher auf klassische UIMS-Techniken zurück.

2.1 Entwicklungssysteme für Benutzungsschnittstellen

UIMS im engeren Sinne sind mächtige Programmbibliotheken zur Bereitstellung von Abstraktionen zur Implementierung des Laufzeitsystems von Benutzungsschnittstellen. Unter Betrachtung des

gesamten Spektrums von Entwicklungsaktivitäten (siehe Abbildung 8) spricht Myers von Schnittstellenentwicklungssystemen (user interface development systems [222]). Sie bestehen nicht nur aus Programmbibliotheken (für das Laufzeitsystem), sondern aus verschieden Teilprogrammen zur schnellen Entwicklung von Oberflächen. Foley et al. verwenden den Terminus Schnittstellen-Designumgebung (user interface design environment [92]). Im weiteren Sinne werden auch die in Abbildung 8 aufgeführten (Hilfs-)Programme unter dem Begriff UIMS zusammengefaßt.

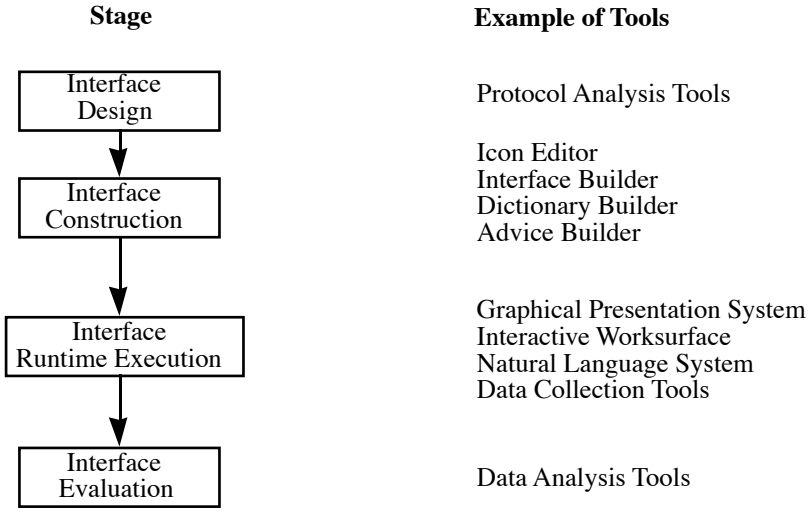


Abbildung 8. Verschiedene Phasen der Oberflächenentwicklung (nach [128]).

Im UIMS-Bereich gibt es sehr viele verschiedene Ansätze und Systeme. In [91] geben Foley et al. einen umfangreichen Literaturüberblick, der hier nicht wiederholt werden soll (siehe auch Myers, [222]). In diesem Abschnitt werden die aktuellen Haupttrends, Modelle und Abstraktionen aus dem Bereich UIMS skizziert, die zur Motivation der HAMVIS-Konzeption und zum Verständnis der Anforderungen der Realisierung des HAMVIS-Laufzeitsystems für eine Anwendung auf der technischen Ebene notwendig sind.

2.1.1 Systeme zur Unterstützung von Designern in der Entwurfsphase

Für die Entwurfsphase einer Anwendung wurden verschiedene Werkzeuge zum Zeichnen der Oberfläche vorgestellt (siehe z.B. [169]). Die Idee besteht darin, Designern ohne Erfahrung mit formalen Repräsentationsformalismen eine Arbeitsumgebung zur Verfügung zu stellen, mit der Bestandteile der Oberfläche für eine Anwendung gezeichnet bzw. skizziert werden. Auch dynamische Aspekte der Interaktion können durch Einblendung von neuen Bildern z.B. als Reaktion auf Mausklicks erfaßt werden. Die Implementierung einer Oberfläche wird allerdings durch diesen Ansatz kaum vereinfacht. Es entsteht das Problem, daß Änderungswünsche des Designers, die nach der Implementierung zu erwarten sind, zwar schnell aufgezeichnet werden können, jedoch einen erheblichen Aufwand bei der Neuprogrammierung verursachen. Der Zeichen- bzw. Skizzierungsansatz ist also nur für die erste Entwurfsphase geeignet.

Üblicherweise bestehen Oberflächen aus graphischen Elementen, die mit Hilfe von speziellen Eingabegeräten manipuliert werden können. Daher scheint es offensichtlich, die gleichen direktmanipulati-

ven Eingabetechniken auch für die Erstellung von Oberflächen zu verwenden. Systeme, die dieses unterstützen, werden als interaktive Schnittstellenbaukästen (interface builder) bezeichnet.

2.1.2 Interaktive Schnittstellenbaukästen

Schnittstellenbaukästen stellen standardisierte Dialogobjekte wie Schaltflächen (buttons), Listen und Tabellen, Textbeschriftungs- und -editierfelder, Armaturen (gauges oder interactors in Myers' Terminologie [223]) etc. bereit, die interaktiv in einem Dialogfenster plaziert werden können. Kommerzielle Produkte sind für verschiedene Sprachen und Programmentwicklungsumgebungen verfügbar (Common Lisp: z.B. Interface Builder und Classworks™ von Harlequin,[11] C++: StarView von StarDivision, Smalltalk: ObjectWorks von ParcPlace).

Einfache, formularbasierte Schnittstellen können mit geringem Aufwand erstellt werden. Wenn jedoch eine erweiterte Flexibilität notwendig ist, stellen sich die Schwächen von Schnittstellenbaukästen heraus. Es ist zwar möglich, anwendungsspezifische Graphiken mit Werkzeugen zu zeichnen, die auch in Zeichenprogrammen zu finden sind,[12] jedoch ist dieses nur für statische Daten sinnvoll. Denkbar ist zum Beispiel, die Position von graphischen Objekten an Zustände von Anwendungsobjekten zu binden[13], wenn in einer Anwendung aber z.B geometrische Daten manipuliert werden, die sich dynamisch ändern können, so ist eine Programmierung auch bei Verwendung von Schnittstellenbaukästen kaum zu umgehen. Wenn die Anwendung schon geometrische Daten verarbeitet, so ist es außerdem ungünstig, die gleichen Daten für die Oberfläche noch einmal zu erfassen bzw. als gezeichnete Graphikobjekte doppelt zu repräsentieren. Schnittstellenbaukästen sind m.E. für die Seeheim-Entwurfsmethodik von Anwendungen ausgelegt.

Graphische Objekte, die in einem Teilfenster gezeigt werden, müssen automatisch maussensitiv gemacht werden, wobei entsprechende Anzeichen (z.B. ein Rechteck um den maussensitiven Bereich) gezeigt werden müssen, wenn der Mauszeiger sich über einem anklickbaren Objekt befindet. Bei vielen Schnittstellenbaukästen wird diese Art der Maussensitivität höchstens für vordefinierte Standard-Interaktionsbausteine bereitgestellt (z.B. Piktogramme, Schaltflächen). Das System ClassWorks (von Harlequin Inc.) unterstützt dieses aber z.B. auch für gezeichnete Graphiken. In ClassWorks können spezielle, vordefinierte „Aktionen" (Ausschneiden, Kopieren, Einsetzen, Bewegung innerhalb vorgegebener Einschränkungen) auf die graphischen Objekte angewendet werden.[14] Allerdings wird zur Zeit noch kein Rückkopplungsmechanismus unterstützt, um anzuzeigen, welche Aktion auf die Objekte anwendbar ist, auf die die Maus zeigt. Es gibt andere Systeme, die ähnliche Mechanismen

---

11. Beispiele für nicht-kommerzielle Lisp-basierte Systeme sind GINA [292] [293] und Gilt [226] (eine Komponente von Garnet [225]).

12. Beispiele für Systeme dieser Art sind KEEpictures [344], Pogo (ein Teilsystem von HITS [128]), ClassWorks von Harlequin Inc. und G2 von Gensym Inc. Zusätzlich zu den objektorientierten Zeichenwerkzeugen werden verschiedene Wege zur Kopplung von graphischen Objekten mit Anwendungsobjekten unterstützt. HITS stellt außerdem einen Gesten-Editor bereit, um quasi Freiform-Mausgesten zu erkennen. Die Semantik von Gesten ist allerdings inhärent mehrdeutig. Siehe hierzu z.B. die Arbeiten von Wahlster [328].

13. Dieses wird durch SimKit unterstützt, einer anderen Komponente des KEE-Systems [61]. Objekte, ihre Eigenschaften, Beziehungen und Beschränkungen werden auf die folgenden graphischen Merkmale abgebildet: Form, Textur, Farbe, Schattierung, Größe, relative Position, Animationsgeschwindigkeit.

14. Aktionen sind als generische Funktionen (im CLOS-Sinne [294]) realisiert. Zusätzliche anwendungsspezifische Aktionen können definiert werden, und Methoden können zu neuen und bestehenden Aktionen hinzugefügt werden.

unterstützen (z.B. Hypercard™ oder Toolbook™), wobei jedoch jedes System seine eigene Nomenklatur für die zugrundeliegenden Konzepte verwendet.

Das automatische Verwalten von maussensitiven Bereichen ist nicht trivial. Schon die Definition der Anwendungsbedingung von Aktionen ist bei einer kompositionellen Hierarchie mit Relationen wie „besteht-aus“, „enthalten-in“ usw. eventuell semantisch mehrdeutig (vgl. z.B. Pars-pro-toto-Deixis). Insbesondere für eine Rückkopplung beim Eintreten der Maus in einen maussensitiven Bereich (z.B. durch Hervorhebungstechniken) sind ausgefeilte Indexstrukturen notwendig, um die für interaktive Anwendungen erforderliche Verarbeitungsgeschwindigkeit zu erzielen.[15] Durch ein UIMS müssen die hierfür benötigten Konzepte und Algorithmen bereitgestellt werden.

Das Layout von Interaktionselementen in einem Dialogfenster muß ggf. adaptiert werden, wenn das Fenster interaktiv in seiner Größe geändert wird (z.B. Listen oder Textfelder, die in einem solchen Fenster plaziert sind, können länger und breiter gemacht werden). Die Größe des gesamten Anwendungsfensters hängt von der Größe und Positionierung der einzelnen Interaktionselemente ab. Um nicht „von Hand“ Positionen mühsam bestimmen zu müssen oder aufwendigen Code zur Adaption des Layouts bei Änderung der Fenstergröße zu schreiben, werden durch UIMS deklarative Layoutbeschreibungsformen bereitgestellt. Komplexere Abhängigkeiten bei der Größe von Dialogelementen lassen sich nicht auf einfache Weise durch graphische Interaktion mit einem Schnittstellenbaukasten beschreiben. Daher wird meist eine textuelle Beschreibungssprache verwendet, um Layouteinschränkungen zu definieren (mit Angaben für Teilungsverhältnisse, Minimal-, Maximal- und Standardangaben für Breiten und Höhenspezifikationen). Zu beachten ist, daß eine graphische Angabe der Höhe in Pixeln z.B. bei Textfenstern wenig sinnvoll ist, da die Höhen vernünftigerweise in Zeilen und nicht in Pixeln angegeben werden sollte. Die Höhe einer Zeile hängt von dem gewählten Zeichensatz ab, so daß die tatsächliche Höhe erst zur Laufzeit bei der Fenstererzeugung bekannt wird (siehe [109] und [212] für eine Diskussion über Layoutspezifikationen). In einigen Fällen ist es sinnvoll, die Größe eines Unterfensters auf den gezeigten Inhalt abzustimmen. Die Bestimmung eines Layouts für die Teilfenster einer Applikation kann also als recht komplexer Einschränkungserfüllungsprozeß angesehen werden.[16] Für ein mit HAMVIS erstelltes Anwendungssystem muß für jede definierte Aktivität ebenfalls eine Beschreibung der Abhängigkeiten bei der Fensteranordnung und -größenbestimmung vorgegeben werden.

Es gibt einige Vorschläge, Layouteinschränkungen für anwendungsspezifische graphische Objekte, die in einem Graphikfenster gezeigt werden, durch direktmanipulative Techniken und Werkzeuge zu definieren (z.B. Lapidary [224], OPUS [135] oder auch ClassWorks).[17] Während hierarchische, boxenorientierte Layoutbeschreibungen auch interaktiv spezifiziert werden können, gibt es Probleme bei der Definition von relativen Positionierungseinschränkungen von graphischen Objekten in einem Teilfenster (z.B ausschließlich horizontale und vertikale Bewegung, beschränkte Bewegung in einem

15. Ob ein Bereich oder ein Objekt maussensitiv ist, kann von den gedrückten Umschalttasten (Shift, Meta, Control, etc.) abhängig sein.

16. Die Arbeit von Graf im WIP-Projekt hat gezeigt, daß es notwendig sein kann, auch den Inhalt einer Darstellung anzupassen, wenn eine akzeptable Layoutgestaltung nicht ermittelt werden kann [102]. Dieses tritt insbesondere auf, wenn die Darstellungen z.B. auf Papier ausgedruckt werden sollen, ist jedoch für rollbare Darstellungen auf einem Computerbildschirm eventuell auch bedeutsam.

17. Arbeiten zu diesem Thema sind von Kurlander [167] und Myers [221] publiziert worden. Ein Überblick über die verschiedenen Ansätze ist in [62] zu finden.

bestimmten rechteckigen Bereich, usw.). In den meisten Fällen ist die Ausdrucksmächtigkeit von graphischen Spezifikationstechniken zu eingeschränkt. Obwohl die Intention dieser Systeme eine Erleichterung der Spezifikation bei gleichzeitiger Erhöhung der Ausdrucksmächtigkeit war, sind die Systeme tatsächlich nicht sehr einfach zu verstehen und schon gar nicht intuitiv zu benutzen. Szekely, einer der Autoren von Lapidary, kommt zu dem gleichen Schluß [309].

Die Entwicklung von Oberflächen erfordert mehr als durch Schnittstellenbaukästen unterstützt wird. Programmiertätigkeiten (auf niedriger Ebene) sind für ernsthafte Anwendungen immer noch notwendig. Viele Informationen zur Abbildung von Anwendungsobjekten auf graphische Objekte werden durch interaktive Schnittstellenbaukästen nicht explizit modelliert. Sie bleiben implizit und können nicht von anderen Teilsystemen „referenziert“ werden. Zum Beispiel werden in den bekannten Schnittstellenbaukästen Anwendungsfunktionen, die über Menüinteraktion oder direkter Manipulation aufgerufen werden, nicht über einen kohärenten Kommandotabellenmechanismus verwaltet. Wenn ein „Kommando“ bei einem bestimmten Anwendungszustand nicht verfügbar ist, sollten alle Aufrufmöglichkeiten automatisch gesperrt sein. Dem Benutzer wird dieses durch geeignete Anzeichen vermittelt (z.B. graue Menüeinträge, graphische Objekte sind nicht mehr maussensitiv, werden also nicht mehr hervorgehoben, wenn die Maus auf sie zeigt). Desweiteren bereitet die interaktive Spezifikation von Oberflächen mit Hilfe von Schnittstellenbaukästen Probleme bei der Portabilität, weil jeweils die *Form* von Interaktionselementen definiert wird. Die Form von Interaktionselementen kann sich allerdings je nach der Gestaltung (look and feel) der Umgebung entscheidend ändern. Es ist daher notwendig, nicht nur die Form, sondern die *Funktion* von Elementen zu spezifizieren. Um die Unzulänglichkeiten von Schnittstellenbaukästen zu überwinden, wurden seit längerer Zeit in Forschungsprojekten sogenannte „modellbasierte Schnittstellenwerkzeuge“ entwickelt.

2.1.3 Modellbasierte Schnittstellenentwicklungsumgebungen

Zur Diskussion der praxisbezogenden Entwicklung von modellbasierten Schnittstellenwerkzeugen möchte ich die Systeme UIDE, HUMANOID und den gemeinsamen Nachfolger MASTERMIND diskutieren. Auf eher theoretische Ansätze aus dem Bereich Mensch-Computer-Interaktion gehe ich später ein.

Foley und Sukaviriya geben in [93] einen Überblick über die Entwicklungsgeschichte von UIDE und vermitteln eine ausführliche Literaturübersicht über eigene und andere Arbeiten. Die Hauptaspekte der Forschungsarbeiten betreffen die folgenden Punkte:

- Spezifikation von interaktiven graphischen Applikationen in Form von Objekten, Aktionen auf Objekten sowie Vor- und Nachbedingungen von Aktionen,
- Transformationen von Spezifikationen von Schnittstellen, so daß verschiedene Designalternativen betrachtet werden können,
- Automatisches Layout für Dialogboxen und Menüs unter Berücksichtigung der Anwendungs- und Aktionensemantik (ausgedrückt durch Vor- und Nachbedingungen) [148],
- Kontextsensitive animierte Hilfe (durch Vorwärts- und Rückwärtsverkettung von Vor- und Nachbedingungen von Aktionen).

Mit modellbasierten Schnittstellenwerkzeugen werden explizite Modelle aufgestellt, mit denen festgelegt werden kann, was ein Oberflächenelement leisten bzw. wie es sich verhalten soll. Die folgenden Aspekte werden in dem System HUMANOID [309] [310] betrachtet:

1. Semantik der Anwendung: Objekte und Operationen der Domäne,
2. Präsentationsschemata (presentation templates): visuelle Gestaltung einer Oberfläche durch Standard-Interaktionsbausteine (engl. widgets: `line`, `icon`, `text`, `menu`, `button`, `column`, `row`, `table`, `graph`),
3. Verhaltensmodelle: Eingabegesten, die auf präsentierte Objekte angewendet werden können, Bausteine für spezielle Interaktionsformen (z.B., `new-point-interactor`, `angle-interactor`, etc.),
4. Definition einer Reihenfolge für die Aktivierung von Kommandos,[18]
5. Seiteneffekte von Aktionen und automatisch auszuführende Folgeaktionen (z.B. Markierung eines gerade erzeugten Objektes, so daß dieses das „aktuelle“ Objekt wird).

HUMANOID unterstützt mit seinen „Presentation Templates“ in erster Linie die Gestaltung von formularbasierten Oberflächen (im weiteren Sinne) mit Standard-Interaktionswerkzeugen (siehe auch die Arbeiten zu ITS [338], ADEPT [140] [141] und TRIDENT [23]).

Es wird deutlich, daß im HUMANOID-Projekt ähnliche Ziele verfolgt wurden wie bei UIDE [91], dessen Schwerpunkt auf der Beschreibung des Effektes eines Kommandos liegt (z.B. zur Generierung von interaktiven Hilfesystemen). Die zentrale Komponente ist eine „Designwissensbasis“. UIDE modelliert einfache Aktionen, die auf Standard-Interaktionselemente anwendbar sind: Erzeugen, Löschen, Attributmodifikation (z.B. Änderung der Liniendicke), Kopieren, Ausschneiden, Einsetzen. Basisklassen für graphische Objekte sind Linien, Formen und Text. Ein Frame-System modelliert die Objektattribute und speichert die jeweils möglichen Aktionen, die auf den Objekten definiert sind. Die Betrachtung von ClassWorks machte deutlich, daß Ideen dieser Art zum Teil aber schon in kommerzielle UIMS eingeflossen sind und teilweise auch erweitert wurden (siehe z.B. Konzepte zur Beschreibung von Bewegungseinschränkungen).

In UIDE werden Aktionen mit Vor- und Nachbedingungen modelliert, die sich jeweils auf einen aktuellen Weltzustand beziehen. Neue Objektklassen können durch Spezialisierung von vordefinierten Klassen eingeführt werden. Weiterhin ist es möglich, zusätzliche Aktionentypen einzuführen. Modelle für Objekte und Aktionen werden für automatische Konsistenz und Vollständigkeitsüberprüfungen der Wissenbasis eingesetzt. Zur Analyse einer Schnittstelle werden eine Menge von Vollständigkeits- und Konsistenzregeln bereitgestellt. Die Regeln sind mit einer Kombination von ART (einem Frame-System) und Lisp formuliert und können von einem erfahrenen UIDE-Entwickler erstellt und erweitert werden. Obwohl der Versuch unternommen wurde, deklarative Modelle zu erstellen, ist die Semantik jedoch nicht formal durch ein mathematisches Modell definiert, sondern stark durch die Inferenzmaschine von ART bestimmt. Die formale Analyse und Bewertung der Modelle fällt also schwer. Zur Zeit werden die modellbasierten Schnittstellenwerkzeuge UIDE und HUMANOID weiterentwickelt und verschmolzen. Das resultierende System heißt MASTERMIND.

18. Reihenfolgeeinschränkungen für Kommandos sollten mit einem groben Granularitätsgrad angegeben werden. Generell sollte ein modusarmer Dialog bevorzugt werden.

MASTERMIND wird eine objektorientierte Beschreibungssprache für Anwendungsobjekte, Benutzeraufgaben (tasks), und Präsentationskomponenten bereitstellen. Ein Aufgabenmodell wird von den Schnittstellendesignern erstellt, um zu repräsentieren, welche Aktionen der Endbenutzer mit den Anwendungsobjekten durchführen kann. Hierzu werden Aufgaben durch Ziele und Parameter spezifiziert. Es werden verschiedene Untertypen von Aufgaben definiert: zerlegte Aufgabe (abstract task), Interaktionstechniken, Präsentationsaufgaben, Anwendungsaufgaben, Fehlerbehandlungsaufgaben usw. Bei einer Zerlegung wird die Zerlegungsart angegeben (sequentiell, alternativ usw.). Weiterhin kann spezifiziert werden, daß eine Aufgabe wiederholt durchgeführt werden muß. Auf der untersten Ebene werden Präsentationsaufgaben mit Interaktionstechniken spezifiziert (Anklicken einer Schaltfläche (button), Selektion eines Menüpunktes, Anklicken eines Graphikobjektes). Hinter „Anwendungsaufgaben“ verbergen sich Berechnungs- und Speicherfunktionen. Mit der Aufgabenbeschreibung ist durch Präsentationsaufgaben auch eine Beschreibung der Oberflächenkomponenten verbunden, die benötigt werden, um die Aufgaben durchzuführen. Die Ziele einer Aufgabe dienen dazu, im Laufzeitsystem zu ermitteln, ob eine Aufgabe durchgeführt werden muß bzw. kann. Weiterhin sind verschiedene Werkzeuge zur Spezifikation des Layouts von Präsentationskomponenten vorgesehen (z.B. mit Grid-Techniken).

Eines der Hauptziele von MASTERMIND ist die Entwicklung von expliziten Aufgabenmodellen, um Interaktionsgesten auf die semantische Ebene der Anwendungsfunktionen abzubilden. Dieses wird zur Generierung von automatisierten Hilfesystemen benötigt. Die geplanten Arbeiten sehen vor, Modelle für die Aufgabenanalyse (task analysis) und Benutzerbeobachtung (user monitoring) in MASTERMIND einzubeziehen [230]. Es werden Ansätze zur Aufgabenanalyse aus der kognitiven Psychologie betrachtet: insbesondere GOMS (goals, operators, methods, selection rules). GOMS wurde von Card, Moran und Newell [38] eingeführt, um ein Modell zur Vorhersage der Performanz von erfahrenen Benutzern von ASCII-orientierten Texteditoren zu erstellen und nicht um einen solchen Editor zu erstellen. Allerdings sind bei GOMS sogar die Fähigkeiten zur Performanzanalyse umstritten (siehe die Ausführungen über die Probleme von GOMS und seinen Nachfolgern weiter unten).

Die Aufgabenbeschreibung von MASTERMIND ist vergleichbar mit der Aktionenzerlegung in HAMVIS. Ähnlich wie in HAMVIS realisiert (s.u.), soll von MASTERMIND ein effizientes Laufzeitsystem erzeugt werden. Im Gegensatz zu HAMVIS muß aber in MASTERMIND die Aufgabenbeschreibung auch eine explizite Beschreibung der Präsentation mit allen gezeigten Graphikelementen enthalten (MASTERMIND wird hierfür eine explizite Beschreibungssprache bereitstellen). Die Spezifikation der Interaktionstechniken erfolgt ebenfalls explizit „bis zum letzten Mausklick“ in der Aufgabenbeschreibung. Das Präsentationsmodell von MASTERMIND unterstützt die Spezifikation von Oberflächen in ähnlicher Weise wie die oben diskutierten Schnittstellenbaukästen (interface builder). Aufgabenmodelle dienen also m.E. in erster Linie zur Steuerung des Designprozesses, der auf der unteren Ebene mit Standard-Werkzeugen aber wie gewohnt vollständig durchgeführt werden muß. Allerdings sollen die Aufgabenmodelle zur Generierung von Hilfesystemen eingesetzt werden. Wie dieses genau erfolgen soll, geht aus den zur Verfügung stehenden Veröffentlichungen allerdings noch nicht hervor.

Wie die Autoren selbst zugeben [311], kann für die eingeführten Modellierungskonstrukte keine formale Semantik angegeben werden. Die Bedeutung der Konstrukte verbirgt sich in dem prozeduralen Code der von MASTERMIND bereitgestellten Werkzeuge (tools), mit denen die Modelle verarbeitet

werden. Damit würde ich die vorgeschlagenen Repräsentationsformen bestenfalls als explizit, nicht jedoch als deklarativ bezeichnen. Im weiteren Verlauf dieser Arbeit wird gezeigt, daß ein theoretisch fundierter Ansatz zur Aktionenspezifikation mit Beschreibungslogiken auch die deklarative Beschreibung der notwendigen Ableitungsalgorithmen erlaubt, daß aber trotzdem interaktive Oberflächenkomponenten so gestaltet werden können, daß Aspekte der praktischen Benutzbarkeit durch einen Entwickler berücksichtigt werden können.

Der Ansatz von MASTERMIND unterscheidet sich wesentlich von dem in dieser Arbeit mit HAMVIS verfolgten Ansatz, da bei MASTERMIND visuelle Oberflächenkomponenten explizit durch den Designer vorgegeben werden müssen. Weder der Inhalt noch die konkrete Ausgestaltung von Visualisierungen, noch die Beziehungen zwischen Visualisierungen und Visualisierungsteilen werden systematisch aus vorgegebenen generischen Modellen und hinzugefügten anwendungsspezifischen Modellen hergeleitet, sondern müssen explizit vorgegeben werden. Die in der Einleitung skizzierten Teilprobleme bei der Erstellung von anwendungsspezifischen Visualisierungen werden durch modellbasierte Schnittstellenwerkzeuge nicht adäquat behandelt.

Weiterhin erscheint es mir sinnvoll, die Probleme von Schnittstellenbaukästen zu vermeiden und statt der konkreten Form von Präsentationen die *Funktion von Präsentationsteilen* hervorzuheben, um Portabilitätsproblemen vorzubeugen. Im Bereich der UIMS wurden schon Fortschritte in diesem Bereich erzielt.

2.1.4 UIMS-Abstraktionen und Programmbibliotheken

Die von einem UIMS auf der Softwareebene bereitgestellten Dienste sind wichtig für die Generierung des Laufzeitsystems einer mit HAMVIS modellierten Anwendung. Im folgenden werden die gebräuchlichen Abstraktionen in modernen, kommerziell verfügbaren UIMS-Architekturen genauer betrachtet, so daß klar wird, auf welche Mechanismen HAMVIS aufbauen kann und welche für die Generierung eines Laufzeitsystems notwendigen Informationen aus den anwendungsspezifischen Modellen und den von HAMVIS bereitgestellten Wissensbasen abgeleitet werden müssen. Insbesondere folgende Punkte werden näher betrachtet:

- Konzepte zur Strukturierung einer Oberfläche in Teilfenster und zur Beschreibung des (relativen) Layouts für Teilfenster,
- Konzepte zur Einbindung von Anwendungsfunktionalität („Kommandos“ der Anwendung),
- Organisation des Eingabe-Berechnungs-Ausgabezyklus,
- Organisationsformen zur Verwaltung von Interaktionsgesten (Mausklicks, Menüauswahlgesten, textuelle Kommandoeingaben) in Abhängigkeit von dem aktuellen Anwendungszustand,
- Möglichkeiten zur *portablen* Bereitstellung von standardisierten Interaktionsbausteinen (gadgets) und Möglichkeiten zur Einbindung von direktmanipulativen Interaktionsobjekten, die für eine Anwendungsklasse benötigt werden und eine angemessene semantischer Rückkopplung bereitstellen (z.B. durch angezeigte Verschiebungsbereiche).

Obwohl im UIMS-Bereich (noch) keine sprachübergreifenden Standards existieren, sind Abstraktionen, die von verschiedenen Systemen verwendet werden, im Kern relativ gleichwertig. Hinsichtlich der jeweiligen Nomenklatur gibt es jedoch beträchtliche Unterschiede. Um nun die relevanten Kon-

zepte und Begrifflichkeiten zu erläutern, ist es notwendig, sich auf ein konkretes System zu beziehen. In diesem Abschnitt werden kurz die Konzepte des für HAMVIS verwendeten UIMS erläutert. Für HAMVIS wurde CLIM (Common Lisp Interface Manager) verwendet, der Quasi-Standard für die Entwicklung von „Common Lisp"-Systemen. Sofern in anderen Systemen (z.B. Smalltalk) unterschiedliche Abstraktionen verwendet werden, wird explizit darauf hingewiesen. CLIM wurde als Basis für HAMVIS nicht nur wegen der kommerziellen Verfügbarkeit ausgewählt, sondern weil es m.E. in den für HAMVIS relevanten Punkten einzigartige und richtungsweisende Konzepte enthält, die weit über das hinausgehen, was andere Systeme anbieten.

CLIM ermöglicht es z.B., das Layout von Teilfenstern und Interaktionselementen zu beschreiben, und stellt Techniken zur Verwaltung von maussensitiven Bereichen bereit. Weiterhin werden verschiedene Programmiertechniken auf hoher Abstraktionsebene zur Verfügung gestellt: ein vom Konzept her ausgefeiltes Graphikmodell mit affinen Transformationen und verschiedenen Möglichkeiten zur Farbdefinition, formatierte Ausgabe von Tabellen, Verarbeitung von textuell eingegebenen Kommandos (kontextsensitive Eingabe von Parametern für textuelle Kommandos mit der Maus), Integration von Anwendungen in das Wirtssystem, interaktive Hilfe. Viele Ideen von UIDE und HUMANOID sind auch in CLIM realisiert. Durch die spezielle Art der Definition von Interaktionskomponenten versucht CLIM, einen gewissen Grad an Portabilität zu erreichen. Programme sind in einer Art und Weise strukturiert, so daß sie automatisch an das Aussehen und Verhalten (look and feel) des Wirtssystems angepaßt werden können, d.h. Deklarationen werden auf einer funktionalen Ebene und nicht auf der Ebene der graphischen Erscheinung (Form und Position) angegeben. Ich würde CLIM als modellbasiertes UIMS bezeichnen, da auch hier in gewissem Sinne explizite Modelle bereitgestellt werden müssen, aus denen CLIM dann oberflächennahe Repräsentations- und Interaktionsformen generiert. Die wesentlichen Konzepte werden in den nachfolgenden Abschnitten kurz besprochen.

Konzepte zur Strukturierung der Oberfläche

Zur Strukturierung der Oberfläche verwendet CLIM eine Terminologie, die noch an den sichtbaren Objekten orientiert ist. Ein *Anwendungsrahmen* (*application frame*) ist das Hauptanwendungsfenster, das durch den Fensterverwalter (window manager) des Wirtssystems gesteuert wird. Ein Anwendungsrahmen besteht aus mehreren *Teilfenstern* (panes), die durch eine deklarative Spezifikationssprache angeordnet werden können.[19] Die Idee der Layoutsprache besteht darin, horizontale und vertikale Einschränkungen durch geschachtelte Boxen mit Minimum- und Maximumangaben für Höhen und Breiten von Teilfenstern auszudrücken. Die Größe eines Anwendungsrahmens kann auch durch den Inhalt der Teilfenster bestimmt werden.[20] Ein Anwendungsrahmen kann mehrere verschiedene Layoutspezifikationen enthalten. Ähnliche Strukturen finden sich auch in Smalltalk-Systemen.

---

19. Eine Deklaration eines Anwendungsrahmens ist tatsächlich eine CLOS-Klassendeklaration, d.h. es können verschiedene Oberklassen angegeben werden, die jeweils das Verhalten eines Rahmens beeinflussen können. Ein konkreter Rahmen wird zur Laufzeit der Anwendung als Instanz dieser Klasse erzeugt. Eine Deklaration für einen Anwendungsrahmen besteht aus Angaben für Instanzvariablen (slots), die den Anwendungszustand speichern, Deklarationen für Teilfenster und ein oder mehreren Layoutspezifikationen sowie weiteren Optionen.

20. Eine detaillierte Beschreibung des Layoutalgorithmus von CLIM findet man in [307] S. 229f.

Verwendung der Konstrukte im Kontext von HAMVIS

Es wird deutlich, daß für eine „Applikation“ im Sinne der HAMVIS-Aktionenzerlegung (siehe Kapitel 1.3) ein CLIM-Anwendungsrahmen als oberflächennahe Repräsentation vorgesehen werden muß. Eine Applikation wird in einer HAMVIS-Aktionenzerlegung in mehrere „Aktivitäten“ zerlegt. Eine Aktivität ist wiederum durch eine Menge von alternativen „zusammengesetzten Handlungen“ des Endbenutzers beschrieben. Zur Darstellung der visuellen Informationen, die für alle Handlungsmöglichkeiten einer Aktivität benötigt werden, sind Teilfenster zu deklarieren. Für alle Teilfenster einer Aktivität ist jeweils eine CLIM-Layoutspezikation für den Anwendungsrahmen zu bestimmen. Die Ausprägung der Layoutspezifikation und damit die Gestaltung der Teilfenster wird durch die Beziehungen zwischen den Fensterinhalten beeinflußt (vgl. die Diskussion zu Abbildung 3). Im Dialogmodell müssen diese Beziehungen zur Entwicklungszeit explizit gemacht und in verschiedenen Varianten verwaltet werden.

Der Inhalt eines Teilfensters wird in CLIM durch eine Zeichenfunktion festgelegt. In Abhängigkeit von der Klasse des Teilfensters werden unterschiedliche Zeichen und Aktualisierungsstrategien eingesetzt. Teilfensterklassen können Eingabefenster für Kommandos sein oder z.B. formularorientierte Dialogfenster (in CLIM sog. accept-values panes). Für Graphikfenster müssen spezielle Zeichenfunktionen definiert werden, die jedoch im Eingabe-Berechnungs-Ausgabezyklus (s.u.) von CLIM bei Bedarf automatisch evaluiert werden.

HAMVIS-Erweiterungen zu CLIM (siehe auch die Komponentenübersicht in dem Szenario aus Abbildung 5) stellen spezielle Fensterklassen für Graphikfenster mit speziellen Zeichenfunktionen bzw. -methoden bereit. Wichtig ist im Kontext von HAMVIS z.B. die Skalierung von Objekten und die Transformation von Weltkoordinaten in Fenster bzw. Viewport-Koordinaten. Für jedes HAMVIS-Fenster wird ein sogenannter „*Ausgabenverwalter*“ (display manager) erzeugt, an den zur Laufzeit die im Eingabe-Berechnung-Ausgabezyklus jeweils zu zeichnenden Objekte „übermittelt“ werden.

Die zu übermittelnden Objekte sind durch die Aktionenmodellierung einer Anwendung und insbesondere durch die Konzepte der verwendeten „elementaren Benutzeraktionen“ bestimmt, die wiederum Bestandteil einer „zusammengesetzte Aktion“ zur Beschreibung einer Handlungsmöglichkeit innerhalb einer Aktivität in Sinne von HAMVIS sind. Innerhalb einer zusammengesetzten Aktion werden elementaren Benutzeraktionen mit automatischen Berechnungsfunktionen verknüpft. In der XKL-Anwendung wird z.B. zur Laufzeit durch Berechnungsfunktionen die Menge der verschiebbaren Objekte und die Menge der möglichen Plazierungsbereiche berechnet (siehe die Diskussion von Abbildung 1). Das von HAMVIS generierte Laufzeitsystem für CLIM enthält entsprechenden Code, der dafür sorgt, daß erstens die Berechnungs- und Speicherfunktionen zur „richtigen“ Zeit aufgerufen werden und zweitens die zu präsentierenden Objektmengen an den Ausgabenverwalter des vorgesehenen Teilfensters übermittelt werden. Der Ausgabenverwalter hat (zur Laufzeit der Anwendung) Zugriff auf die geometrischen Daten und auf die während der Entwicklungszeit festgelegten Zeichenattribute von Domänenobjekten. Der Ausgabenverwalter eines Teilfensters speichert außerdem die gezeichneten Objekte in einer „Diskurshistorie“ und organisiert den Neuzeichnungsprozeß während des Interaktionszyklus.

Es wird deutlich, daß die Abbildung von Aktionenmodellen auf UIMS-Dienste nicht übermäßig schwierig ist, wenn ein mächtiges UIMS wie CLIM verwendet wird. Ich werde später detaillierter auf die Mechanismen des Ausgabenverwalters eingehen und hier die Schilderung der angebotenen Dien-

ste von CLIM fortsetzen. Wir werden sehen, daß auch zusammengesetzte Aktionen ihr Pendant auf der UIMS-Ebene haben.

Kommandos als genereller Abstraktionsmechanismus

Bei der Interaktion mit modernen Fenstersystemen werden Funktionen des Anwendungssystems auf vielfältige Weise ausgelöst: durch Auswahl eines Menüeintrags, durch Anklicken eines graphischen Objekts, durch Eingabe eines textuellen Kommandos oder auch durch Tastaturkürzel usw. Die Anbindung der Anwendungsfunktion an die konkrete Auslösefunktion ist jedoch in den meisten Systemen verschieden (z.B. durch „callbacks“ für Standard-Interaktionsobjekte oder „actions“ für graphische Objekte).

In CLIM wird zwischen Kommandoauslösung und dem Kommandoobjekt, das ein bestimmtes Kommando beschreibt, unterschieden. Ein Kommandoobjekt wird durch den Benutzer durch Menüauswahl, Mausklicks, Tastatureingaben usw. generiert, kann aber auch per Programm erzeugt werden. Es ist charakterisiert durch einen Namen sowie auch bestimmte Aktualparameter, die notwendig oder optional (Schlüsselwortparameter) sein können.

CLIM stellt Deklarationsformen bereit, mit denen Kommandos definiert werden können. Kommandos werden in hierarchische Kommandotabellen eingeordnet. Man gibt bei der Kommandodefinition an, welche formalen Parameter ein Kommando hat und wie deren Typen definiert sind. Weiterhin wird spezifiziert, durch welche „Gesten“ ein Kommando ausgelöst werden soll.[21] Der Rumpf der Kommandodefinition enthält den mit dem Kommando verbundenen Anwendungscode.

Der Vorteil der Kommando-Abstraktionsebene ist, daß Anwendungsfunktionen durch einen einheitlichen Mechanismus an die Oberfläche angebunden werden und nicht für jede Auslösemethode ein anderes Anbindungsverfahren angewendet werden muß. Wenn ein Kommando (oder eine ganze Kommandotabelle) deaktiviert wird, so werden durch CLIM automatisch alle Auslösemethoden gesperrt, d.h. Menüeinträge werden in grau dargestellt und sind nicht mehr auswählbar, der Parser (bzw. Vervollständiger) für die textuellen Kommandos akzeptiert die deaktivierten Kommandos nicht mehr usw. Die expliziten Angaben für ein Kommando nutzt CLIM zur Bereitstellung von interaktiver Hilfe (inklusive Präsentation von einer jeweils möglichen Menge von akzeptierbaren Objekten).[22]

Wenn zur Laufzeit ein Kommandoobjekt „bearbeitet“ wird, verwendet CLIM die hierarchischen Kommandotabellen, um den zugehörigen Anwendungscode zu finden, die Aktualparameter zu übergeben und dann die zugehörigen Anwendungsfunktion (Kommandorumpf) auszuführen. CLIM stellt einen Parser für textuell eingegebene Kommandos bereit, der neben inkrementeller Vervollständigung inklusive Alternativendarstellung auch noch ein Kernsystem für interaktive Hilfen enthält.

---

21. Eine Kommandotabelle kann an der Oberfläche als Menü zur Verfügung gestellt werden. Diejenigen Kommandos, die durch das Menü ausgelöst werden können, werden dann als Menüeinträge dargestellt. Der Menüeintragsname wird standardmäßig aus dem Kommandonamen abgeleitet, kann aber auch extra angegeben werden.

22. Es wurden in jüngster Zeit weiterentwickelte Systeme zur Generierung interaktiver, situationsspezifischer Hilfe vorgestellt (siehe die umfangreiche Arbeit von Thies [318]). Da jedoch von CLIM die Kommandoinformation verwendet wird, die in jedem Fall zum Parsen von Argumenten für Kommandos benötigt wird, stellt der CLIM-Ansatz einen recht guten Kompromiß zwischen Aufwand und Leistung dar.

Bedingt durch die notwendige Ebene der Kommandos, lassen sich Ideen von Schnittstellenbaukästen nicht auf einfache Weise mit denen von CLIM in Einklang bringen. Es gibt jedoch auch erste Ansätze, für CLIM direktmanipulative Schnittstellenbaukästen zu entwickeln [58]. Weil jedoch HAMVIS Anwendungen unterstützt, die mit (dynamischen) geometrischen Daten umgehen, werden die Möglichkeiten von Schnittstellenbaukästen, graphische Objekte interaktiv zu zeichnen, nicht in Betracht gezogen. Die Abstraktionsebene der Kommandos ist einzigartig für CLIM (und seinen Vorgänger „Dynamic Windows“, siehe unten).

Verwendung der Konstrukte im Kontext von HAMVIS

Betrachtet man die Abstraktionsebene der Kommandos mit Blick auf HAMVIS, so wird deutlich, daß „zusammengesetzten Handlungen“ einer mit HAMVIS erstellten Aktionenmodellierung auf CLIM-Kommandos abgebildet werden können. HAMVIS bestimmt also zur Entwicklungszeit aus den Angaben einer Aktionenzerlegung die für die Anwendung zu generierenden Kommandos und deren Rümpfe. Wenn zur Laufzeit der Anwendung durch eine Interaktionsgeste ein Kommando ausgelöst wird, werden die im Rumpf eines Kommandos vorgesehenen Berechnungs- bzw. Speicherfunktionen evaluiert. Dadurch werden neue Objekte erzeugt oder die Daten der Anwendung manipuliert und die Präsentation auf dem Bildschirm muß ggf. aktualisiert werden.

Für die weitere Realisierung von zusammengesetzten Aktionsmöglichkeiten durch die UIMS-Ebene werden noch weitere Konzepte benötigt, die unten erläutert werden.

Eingabe-Berechnungs-Ausgabezyklus

Für die Aktualisierung der in einem Teilfenster eines Anwendungsrahmens dargestellten Graphiken gibt es zwei grundlegende Vorgehensweisen. Beim *Bottom-Up-Vorgehen* werden die Änderungen von Anwendungsobjekten gesammelt, und anschließend werden Aktualisierungsanforderungen für die veränderten Objekte erzeugt, die in verschiedenen Teilfenstern der Anwendung präsentiert werden. Dieses Vorgehen wird von Smalltalk mit dem Model-View-Controller-Schema [165] unterstützt (auch andere Systeme verwenden gleichartige Ideen). Ein Anwendungsobjekt wird „Modell“ genannt, das mit ein oder mehreren „Sichten“ assoziiert ist. Steuerungsobjekte (controllers) organisieren die Manipulation von graphischen Objekten und übersetzen die Änderungen dieser Objekte in sog. Änderungsnachrichten (change messages), die den Modellen (Anwendungsobjekten) gesandt werden. Obwohl der Bottom-Up-Ansatz nicht direkt durch CLIM unterstützt wird, könnte dieses Vorgehen jedoch mit geringem Aufwand implementiert werden. Allerdings ist es zur Realisierung dieses Ansatzes notwendig, daß es möglich ist, die Änderungen an Anwendungsobjekten zu registrieren. Für Standard-Datenstrukturen wie z.B. Felder und einfache Objekte wie Zahlen ist dieses nicht in jedem Fall zu gewährleisten.

Der andere Weg besteht in der Behandlung der Aktualisierung von Fensterinhalten mit einer *Top-Down-Strategie*. Dieses wird durch CLIM direkt unterstützt. Um zu verstehen, wie die Top-Down-Methode funktioniert, muß zunächst das Zeichenmodell für Anwendungsfenster betrachtet werden. Jedem Teilfenster eines Anwendungsrahmens ist eine Zeichenfunktion zuzuordnen. Das Zeichenmodell von CLIM sieht vor, daß elementare Ausgaben (z.B. mit Funktionen wie draw-line, draw-rectangle etc.) in sogenannten Ausgabeaufzeichnungen (output records) vermerkt und bei Bedarf automatisch wieder abgespielt werden. Wenn z.B. ein Teilfenster durch den Benutzer gerollt wird, werden die aufgezeichneten Zeichenaufträge einfach wieder abgespielt. Ausgabeaufzeichnungen kön-

nen in einer Teil-Ganzes-Hierarchie angeordnet sein. Spezielle Ausgabeaufzeichnungen können verwendet werden, um den Aufnahme- und Abspielvorgang zu steuern. Es ist dadurch möglich, bei hierarchisch angeordneten Ausgabeaufzeichnungen inkrementelle Neuaufzeichnungen einzelner Aufzeichnungen vorzunehmen, ohne die anderen Strukturen zu zerstören.

Zur Realisierung der Top-Down-Aktualisierungsstrategie für graphische Ausgaben wird die Ausgabe einzelnen Aufzeichnungen zugeordnet. Der CLIM-Programmierer gibt an, wie die hierarchische Struktur der Ausgabeaufzeichnungen aussieht und für welche Objekte der Anwendung eine Aufzeichnung gelten soll. Die Objekte werden in einem sogenannten Cache gespeichert. Nur wenn sich der Cache-Wert geändert hat, werden Neuaufzeichnungen durchgeführt. Die Teil-Ganzes-Hierarchie für Ausgabeaufzeichnungen bestimmt den Suchraum für geänderte Anwendungsobjekte. CLIM berechnet vor dem Abspielen der Gesamtaufzeichnung automatisch, welche Aufzeichnung zu Lösch- und Zeichenoperationen auf dem Ausgabestrom (Bildschirm) führen müssen (inkrementelle Aktualisierung von Graphikausgaben).

Andere UIMS stellen ebenfalls Dienste bereit, graphische Ausgaben hierarchisch zu strukturieren, verwenden diese Angaben aber nicht wie CLIM zusätzlich zur Organisation des Neuzeichnens, sondern nur zum Verwalten von Mausklicks.[23] Meines Erachtens sind die beiden Ansätze Top-Down- oder Bottom-Up-Aktualisierung weitestgehend gleichwertig. Beide haben ihre Vor- und Nachteile. Der Ausgabenverwalter von HAMVIS nutzt den Top-Down-Ansatz von CLIM zusammen mit den Möglichkeiten zur Ausgabeaufzeichnung und inkrementellen Aktualisierung innerhalb des Eingabe-Berechnungs-Ausgabe-Zyklus.

Der prinzipielle Eingabe-Berechnungs-Ausgabezyklus des CLIM-Laufzeitsystems ist in Abbildung 9 dargestellt. Die oberen und unteren Teile werden durch generelle von CLIM bereitgestellte Mechanismen unterstützt und brauchen nicht für jede Applikation neu programmiert zu werden.

---

23. Die Möglichkeit, Ausgaben aufzuzeichnen, scheint zuerst vielleicht „überdimensioniert“ zu sein. Tatsächlich handelt es sich jedoch um einen Basismechanismus, der sich vielseitig anwenden läßt. Er wird in HAMVIS z.B. dazu verwendet, graphische Objekte mit metagraphischen Symbolen (z.B. Pfeilen) hervorzuheben. Wie in der Einleitung erläutert, werden den zu zeigenden Objekten je nach rhetorischem Status verschiedenen Zeichenattribute zugeordnet. Zeichenattribute werden um mögliche Unterlegungen und obenliegende Markierungen erweitert. Obwohl jedes Objekt für sich gezeichnet wird, muß sichergestellt werden, daß auch bei beliebiger Zeichenreihenfolge Unterlegungen in jedem Fall unten und Markierungen in jedem Fall über anderen Hauptobjekten zu liegen kommen. Zur Lösung dieses Problems lassen sich Ausgabeaufzeichnungen verwenden.

Ausgabeaufzeichnungen können wie Filmbänder verschnitten und in eine andere Reihenfolge gebracht werden. Wenn also die Objekte eines Teilfensters gezeichnet werden, so verwendet HAMVIS für jedes Objekt drei Aufzeichnungsbänder. Ein Band speichert eventuelle Unterlegungen, das zweite Band zeichnet das eigentliche Objekt auf und das letzte Band wird mit obenliegenden Markierungsobjekten gefüllt. Vor dem Abspielen der Bänder für die Generierung der Graphiken in den Fenstern werden die Aufzeichnungen in drei Gruppen sortiert. Als erstes werden die Unterlegungsbänder, dann die Hauptbänder und anschließend die Bänder mit den Markierungsobjekten abgespielt. Dadurch ist gewährleistet, daß Unterlegungen in jedem Fall unter allen Hauptobjekten und Markierungsobjekte immer darüber zu liegen kommen, unabhängig davon, in welcher Reihenfolge die Einzelobjekte gezeichnet werden.

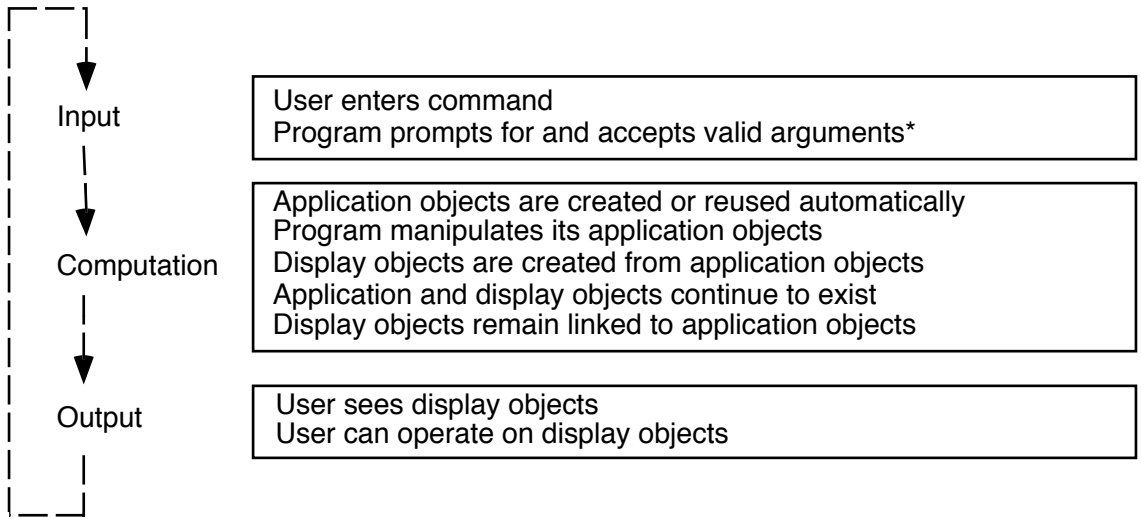


Abbildung 9. CLIM-Ansatz zur Verteilung der Aufgaben beim Eingabe-Berechnungs-Ausgabezyklus (aus [307]).

Bedeutung der Konstrukte im Kontext von HAMVIS

Mit den in diesem Abschnitt vorgestellten Konzepten von CLIM kann die Realisierung der Graphikausgabe innerhalb des Interaktionszyklus durch den Ausgabenverwalter von HAMVIS leicht beschrieben werden. Für die Darstellung einer Menge von Anwendungsobjekten wird jeweils eine inkrementelle Ausgabeaufzeichnung mit Cache verwendet. Der Cache enthält das gezeichnete Anwendungsobjekt oder eine Menge von Attributen, die das Anwendungsobjekt kennzeichnen.

Nach jeder Aktion des Benutzers innerhalb des Interaktionszyklus werden die Berechnungsfunktionen erneut evaluiert, d.h. es ergeben sich ggf. neue Objekte, die an den Ausgabenverwalter übermittelt werden. Objekte, die nicht mehr in einer Informationseinheit und damit auch nicht mehr im Cache der inkrementellen Ausgabeaufzeichnung enthalten, aber auf dem Bildschirm noch zu sehen sind, werden gelöscht, neuhinzugekommene werden gezeichnet. Nach einer weiteren Interaktion des Benutzers beginnt der Kreislauf von neuem.

Die Gestaltung eines Laufzeitsystems zum Zeichnen und zur Aktualisierung der Graphik mit dem Top-Down-Konzept von CLIM ist ohne komplexe Programmierung möglich. Die Konzepte von CLIM sind nicht unbedingt einfach zu verstehen, aber doch sehr mächtig. Für den Oberflächenentwickler, der mit HAMVIS Visualisierungen für eine Anwendung zusammenstellen will, ist diese Ebene jedoch transparent. Er kann sich im HAMVIS-Kontext mit der Modellierung der Aktionen und den dafür benötigten Informationseinheiten beschäftigen und braucht sich nicht um die Graphikverwaltung usw. im Laufzeitsystem zu kümmern. Arbeiten aus dem Bereich Mensch-Computer-Interaktion (s.u.) belegen, daß die Modellierung von interaktiven Systemen mit einem Vokabular, das sich an die Aktionen des Endbenutzers orientiert, eine große Vereinfachung gegenüber der „technischen“ Modellierung auf der UIMS-Ebene darstellt.

Benutzungsschnittstellenbaukästen stellen zwar Bibliotheken für standardisierte Schnittstellenelemente wie Schaltflächen, Rollbalken, Armaturen etc. bereit. Jede dieser Komponenten kann mit der Maus manipuliert werden, wobei der zugrundeliegende Code handkodiert ist und auf tiefer Ebene mit

Ereignissen (events) des Betriebs- bzw. Fenstersystems umgehen muß. Für die Bereitstellung von Interaktionsformen für nichtstandardisierte Elemente, die in Graphikfenstern präsentiert werden, ist der Programmierer in konventionellen Systemen aber selbst verantwortlich. Die Erstellung von Code zur Behandlung von Ereignissen wie Mausklicks zusammen mit der Abbildung von Bildschirm- bzw. Koordinaten für den Fensterausschnitt (viewport) in Weltkoordinaten ist extrem aufwendig und fehleranfällig.

Ein wesentlicher Beitrag von CLIM ist die automatische Verwaltung von maussensitiven Bildschirmbereichen. CLIM gestattet, graphische Objekte, die mit Zeichenfunktionen wie z.B. `draw-line`, `draw-rectangle`, `draw-text` und ähnlichen Basisfunktionen gezeichnet wurden, mit wenigen Angaben „anklickbar“ zu machen. Obwohl andere UIMS wie z.B. ClassWorks ebenfalls erlauben, graphische Objekte (die mit geometrischen Primitiven explizit modelliert wurden) maussensitiv zu machen, ist die Architektur von CLIM viel klarer und dabei noch flexibler. Zu beachten ist, daß in Applikationen, die mit Schnittstellenbaukästen aufgebaut wurden, die graphischen Objekte nicht identisch zu den entsprechenden Anwendungsobjekten sind. Informationen werden also ggf. redundant verwaltet. In CLIM besteht nicht die Notwendigkeit, separate Datenstrukturen für Graphik und Anwendung explizit zu erzeugen, nur um Objekte anklicken zu können. Oberflächen werden in der Terminologie der Anwendungsobjekte erzeugt und nicht in der von speziellen Graphik- oder Werkzeugkastenobjekten. Die Darstellung eines Anwendungsobjektes zusammen mit der von CLIM verwalteten Assoziation der Ausgabeaufzeichnungen mit dem zugehörigen Anwendungsobjekt wird in CLIM *Präsentation* (presentation) genannt.

Der Hauptbeitrag von CLIM zur Weiterentwicklung von UIMS-Konzepten ist, dem Programmierer zu gestatten, Präsentationen mit Typinformationen (presentation types) versehen zu können. Typen werden von CLIM zur Laufzeit der Anwendung eingesetzt, um zu überprüfen, ob bestimmte Operationen auf graphische Objekte (und damit auf den zugeordneten Anwendungsobjekten) in einem speziellen Interaktionskontext angewendet werden können.

Assoziation von Typinformation mit graphischen Ausgaben

Präsentationstypen sind eine Erweiterung des Typsystems von Common Lisp. Sie können als parametrierte CLOS-Klassen angesehen werden. CLOS Klassen sind per definitionem auch Präsentationstypen, d.h. eine mit CLOS modellierte Anwendung kann auf einfache Weise mit CLIM mit einer Graphikoberfläche versehen werden. In der Zeichenfunktion zur Ausgabe eines Objekts wird angegeben, mit welchem Präsentationstyp die zugehörige Ausgabeaufzeichnung versehen werden soll (Präsentationen sind also spezielle, typisierte Ausgabeaufzeichnungen).

Der Einsatz von Präsentationstypen ist mit dem Kommandomechanismus gekoppelt. Die für Kommandos definierten Parametertypen sind Präsentationstypen. Unter Berücksichtigung von Typinformationen kann CLIM automatisch vorherige Ausgaben maussensitiv machen, wenn ein Argumenttyp benötigt wird, der zu dem der Ausgabe „passend“ ist. In CLIM-Terminologie wird statt „passend“ der Begriff „akzeptierbar“ verwendet. Zur Definition von „akzeptierbar“ wird das Vererbungsprinzip herangezogen. Wenn ein akzeptierter Präsentationstyp T1 z.B. eine bestimmte CLOS-Klasse ist, so sind auch diejenigen Objekte akzeptierbar, die mit einem Klassen-Präsentationstyp ausgegeben wurden, von dem T1 erbt. Präsentationstypen können jedoch noch parametriert werden. Parameter können beispielsweise verwendet werden, um einen Unterbereich von Zahlen zu repräsentieren (notiert als

`(integer 1 10)`). Auch parametrierte Präsentationstypen werden in eine Vererbungshierarchie eingeordnet, d.h. `(integer 1 10)` ist ein Untertyp von `(integer 1 100)`.

Ein Beispiel illustriert die Verwendung der Typen bei der Eingabe von Werten. Nehmen wir an, die Zahl 8 wird ausgedruckt. Wenn eine Zahl (später) wieder eingelesen werden soll, z.B. eine Zahl zwischen 1 und 10 (`(integer 1 10)`), so ist die vorher ausgegebene 8 maussensitiv. Wenn allerdings eine Zahl zwischen 10 und 20 verlangt wird, so ist die 8 nicht anklickbar (siehe die Bildschirmabzüge in Abbildung 10 und Abbildung 11). Die Basisfunktionen für diesen Mechanismus sind `present` und `accept`.

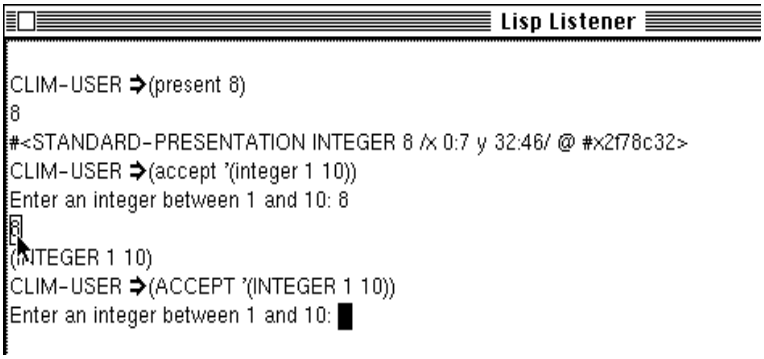


Abbildung 10. Akzeptierung einer Zahl zwischen 1 und 10. Die vorher ausgegebene 8 ist maussensitiv (dieses wird durch ein Rechteck angedeutet).

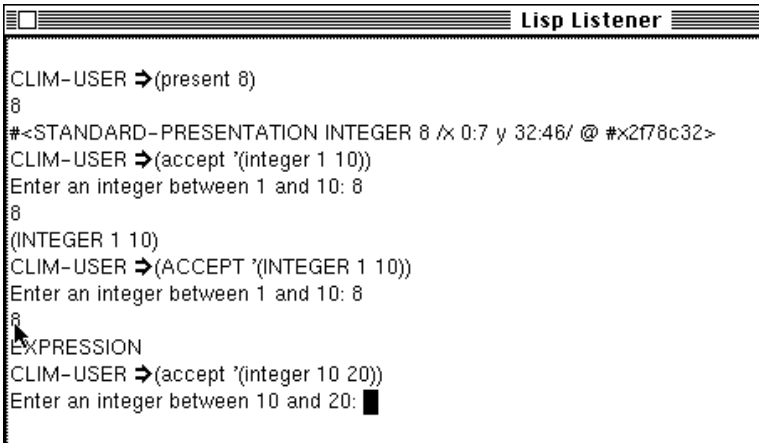


Abbildung 11. Akzeptierung einer Zahl zwischen 10 und 20. Die 8 ist nicht maussensitiv.

Der Akzeptierungsmechanismus ist nicht nur auf textuelle sondern auch auf graphische Ausgaben anwendbar. Dieses möchte ich an einem anderen Beispiel erläutern.

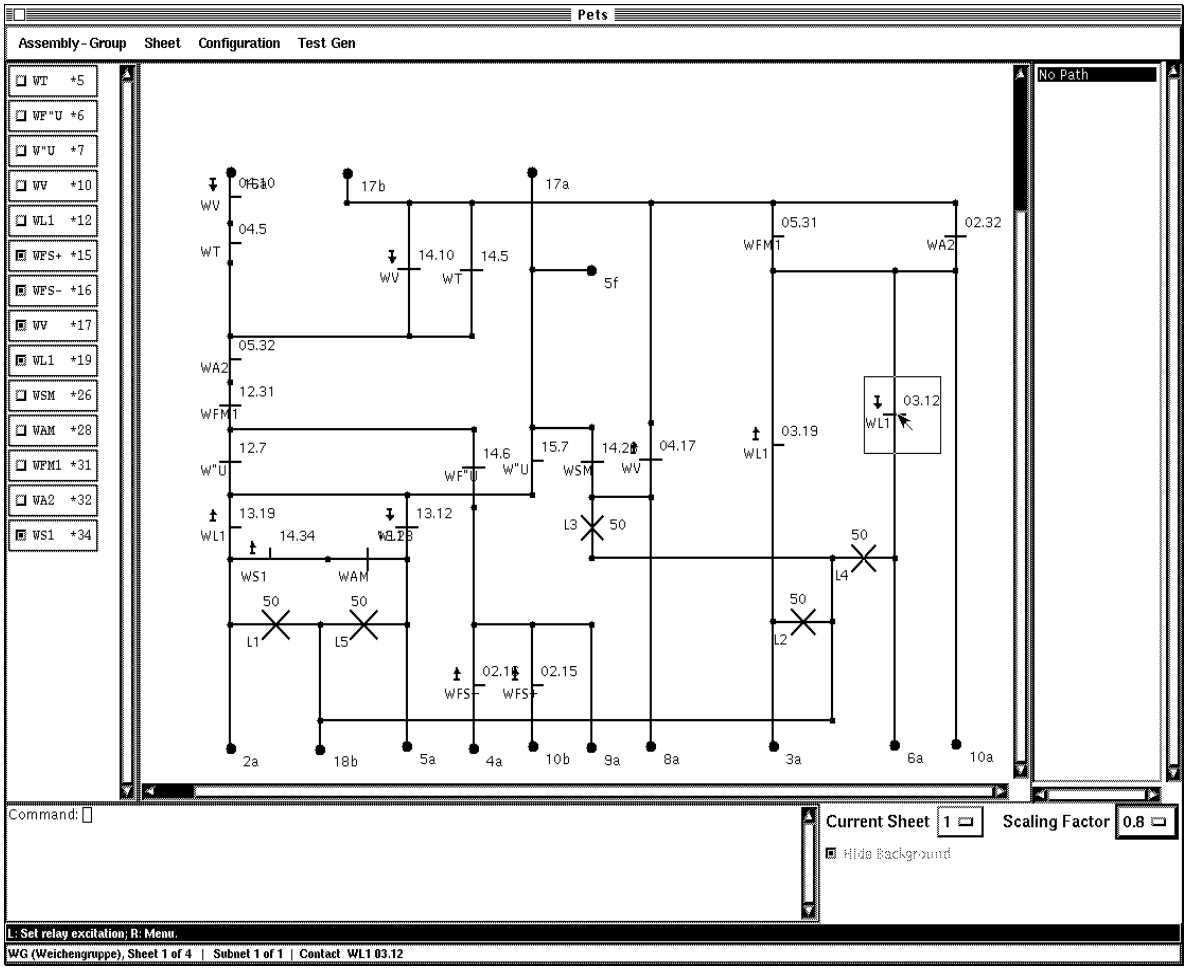


Abbildung 12. Bildschirmabzug eines graphischen Editors für Relaisschaltkreise.

Abbildung 12 zeigt den Anwendungsrahmen einer Anwendung, die mit CLIM entwickelt wurde. Die Anwendung wird zur Generierung von Testplänen für Relaisbaugruppen zur Steuerung von Eisenbahnanlagen verwendet. Im Hauptfenster werden Schaltkreiselemente wie z.B. Relaiskontakte (kleine Balken, die die Drahtstücke berühren oder schneiden), Klemmen (große Punkte), Lampen (als Kreuze) usw. dargestellt. Jedes Schaltkreiselement ist ein CLOS-Objekt und wird mit dem jeweils assoziierten Präsentationstyp ausgegeben.

Im rechten Teil der Schaltkreisdarstellung (siehe den Mauszeiger) kann man erkennen, daß gezeichnete Objekte maussensitiv sind. Das bedeutet, für das Anwendungsobjekt, das sich hinter der Darstellung verbirgt, sind ein oder mehrere Kommandos aktiviert. Durch Drücken der rechten Maustaste erhält man ein Menü aller anwendbaren Kommandos. Im unteren Teil des Anwendungsrahmens wird ein Interaktionsfenster für textuelle Kommandos gezeigt. Wenn ein Benutzer ein Kommando textuell eingibt, z.B. `Delete Object` und dieses Kommando einen Parameter vom Typ `circuit-element` benötigt, so sind automatisch alle graphischen Objekte maussensitiv, die mit diesem oder einem spezielleren Typ ausgegeben wurden. Wenn speziellere Typen verlangt werden, z.B. bei einem Kommando `Switch Contact`, so sind nur Objekte maussensitiv, die mit dem (spezielleren) Präsentationstyp `contact-element` ausgegeben wurden. Wenn ein graphisches Objekt aus mehreren Primitiven

besteht (z.B. `draw-line` und `draw-text`), führt CLIM automatisch eine „Gruppierung“ durch, d.h. alle Teilobjekte, die in einem Präsentationskontext gezeichnet werden, werden automatisch zusammengefaßt und dynamisch bezüglich der Maussensitivität als eine Einheit betrachtet.

Das Beispiel zeigt, welche Dienste durch moderne UIMS-Rahmenwerke schon bereitgestellt werden. Bei Verwendung von CLIM als Zielsystem für HAMVIS brauchen viele „Kleinigkeiten“ der Oberflächenprogrammierung (Aufklappmenüs, textuelle Kommandoeingabe, Maussensitivität von Objekten usw.) nicht mehr explizit durch generierten Code für das Zielsystem berücksichtigt zu werden.

Auch Interaktionsgesten wie z.B. Ziehen-und-Fallenlassen können auf hoher Ebene spezifiziert werden. Dieses wird im nächsten Abschnitt verdeutlicht.

Abbildungen zwischen Präsentationstypen

Für den Ausdruck von Datei-Informationen (z.B. in einem Textfenster mit Dateinamen, Erzeugungsdatum, Größe etc.) könnte ein Präsentationstyp `file-info` verwendet werden. Falls in einem anderen Eingabekontext eine Zeichenkette akzeptiert wird, können Ausgaben vom Typ `file-info` eine adäquate Eingabe darstellen. Ausgaben dieses Typs können in den Typ `string` abgebildet werden, indem nur der Dateiname verwendet wird. CLIM gestattet die Definition von Abbildungsfunktionen von einem Präsentationstyp auf einen anderen (genannt „presentation translation“). Um nun Dateiinformationen maussensitiv zu machen, wenn eine Zeichenkette verlangt wird, ist es ausreichend, eine entsprechende Abbildungsfunktion zu definieren.

Die Basisidee des Eingabe-Berechnungs-Ausgabezyklus besteht darin, einen Präsentationstyp `command` zu akzeptieren. Um ein Kommando durch Anklicken eines graphischen Objekts auszulösen, das mit einem gewissen Präsentationstyp T_0 gezeichnet wurde, kann eine Präsentationsabbildung von T_0 auf `command` definiert werden (ein sog. presentation-to-command-translator). Das bedeutet, daß Ausgaben vom Typ T_0 als Kommando verwendet werden können, ein Kommando, das von der Interaktionsschleife von CLIM akzeptiert wird. Bei der Definition der *Präsentation-nach-Kommando-Abbildungsfunktion* gibt man das Zielkommando und die auslösende Geste an (z.B. linke Maustaste oder rechte Maustaste usw.).[24] Der Rumpf der Abbildung ermittelt aus dem angeklickten Objekt (formaler Parameter) die Parameter des hinterher von CLIM erzeugten Kommandoobjekts.

Eine besondere Art von Präsentation-nach-Kommando-Abbildungsfunktion ist eine *Ziehen-und-Fallenlassen-Abbildungsfunktion* (drag-and-drop-translator). Die Definition einer solchen Abbildungsfunktion ist recht einfach, sie wird in ähnlicher Weise wie eine Präsentation-nach-Kommando-Abbildungsfunktion beschrieben. Es muß aber ein Startpräsentationstyp und ein Zielpräsentationstyp angegeben werden, d.h. eine Ziehen-und-Fallenlassen-Abbildungsfunktion ist insbesondere zum Auslösen von Kommandos mit zwei Parametern geeignet. CLIM stellt schon standardmäßig ein adäquates Rückkopplungsverhalten für die interaktive Realisierung der Geste bereit.[25]

---

24. Man definiert sog. hardwareunabhängige Gesten die mit einem Namen versehen sind. Die Abbildung auf konkrete Hardware erfolgt tatsächlich noch in einem indirekten Abbildungsschritt. CLIM stellt einige vordefinierte hardwareunabhängige Gesten bereit (`:select`, `:edit`, `:describe`, `:menu`).

25. Spezielle Rückkopplungstechniken können durch Hinzufügen von Methoden für bestimmte generische Funktionen realisiert werden.

Man beachte, daß durch Deaktivierung eines Kommandos auch die Präsentation-nach-Kommando-Abbildungsfunktionen deaktiviert werden, die Präsentationstypen auf dieses Kommando abbilden. Dieses bedeutet wiederum, daß bestimmte Objekte automatisch nicht mehr maussensitiv sind.

Verwendung der Konstrukte im Kontext von HAMVIS

Bei Betrachtung der Abstraktionsebene der Präsentation-nach-Kommando-Abbildungen wird deutlich, was für die Realisierungsmöglichkeit einer zusammengesetzten Handlung durch UIMS-Dienste zu leisten ist. Wie schon erläutert, wird jede zusammengesetzte Handlung auf ein Kommando abgebildet, wobei die letzte Berechnungs- oder Speicherfunktion den Kommandorumpf definiert. Die Parameter der Funktion bestimmen die Parameter des Kommandos (sowohl in Anzahl als auch Präsentationstyp). Ein Kommando kann – wie hier diskutiert – durch eine Präsentation-nach-Kommando-Abbildung ausgelöst werden. Das bedeutet, daß die oberflächennahe Ausführungsmöglichkeit einer zusammengesetzten Handlung durch Bestimmung einer geeigneten Präsentation-nach-Kommando-Abbildung zur Entwicklungszeit gewährleistet werden kann. Betrachten wir noch einmal das Beispiel aus Abbildung 1. Der Oberfläche liegt eine zusammengesetzte Handlungsmöglichkeit zugrunde, die – auf der in diesem Abschnitt betrachteten technischen Ebene – durch eine Ziehen-und-Fallenlassen-Abbildungsfunktion umgesetzt wird. Damit die Ziehen-und-Fallenlassen-Abbildungsfunktion zur Laufzeit angestoßen werden kann, ist es erforderlich, vorher die Start- und Zielpräsentationen unter Verwendung von geeigneten Präsentationstypen auszugeben. Zur Entwicklungszeit der Anwendung muß also ein Präsentationstyp bestimmt werden, der bei der Ausgabe der jeweiligen Objekte (zur Laufzeit der Anwendung) zu verwenden ist. Die restlichen Dienste auf niedrigerer Ebene übernimmt dann CLIM. Zusammengesetzte Handlungen von HAMVIS können auf der UIMS-Ebene also durch Kommandos und Präsentation-nach-Kommando-Abbildungen umgesetzt werden. Die Art der Präsentation-nach-Kommando-Abbildung richtet sich jedoch nach den Konzepten der elementaren Benutzeraktionen, die in der Zerlegung der zusammengesetzten Handlung vorgesehen sind.

Portabilitätsbetrachtungen

In einigen UIMS werden Schnittstellenbaukästen zur interaktiven Auswahl und Plazierung von Interaktionsobjekten (gadgets) in Dialogfenstern verwendet. Ein Beispiel für ein Standard-Interaktionsobjekt, das für eine Auswahl zwischen wenigen Alternativen Verwendung findet, ist eine Gruppierung von gekoppelten Umschaltknöpfen (radio buttons). Jeder Knopf repräsentiert ein Anwendungsobjekt oder ein Attribut eines Anwendungsobjekts. Die Verwendung von Schnittstellenbaukästen bedingt meist die exakte Plazierung mit genauen Koordinaten. Ausrichtungen zwischen Objekten (gleiche x- oder y-Koordinaten) sind in den seltensten Fällen explizit repräsentiert. Problematisch ist es, wenn zur Laufzeit die Oberfläche in einer unterschiedlichen Umgebung dargestellt wird (z.B. mit größeren Knöpfen oder anderen Zeichensätzen für die Knopfbeschriftungen). Es kommt ggf. zu Überlappungen oder verschobenen Darstellungen, da Abstände und Ausrichtungen nicht explizit repräsentiert wurden. Der im ersten Augenblick einfache Schritt der Verwendung eines Schnittstellenbaukastens sorgt hier für eine unangenehme Inflexibilität.

Wenn später im Entwicklungsprozeß der Oberfläche klar wird, daß der Benutzer zwischen zu vielen Alternativen wählen kann, kann die Technik der gekoppelten Umschaltknöpfe nicht mehr verwendet werden. Es wird also der Schnittstellenbaukasten erneut herangezogen, und die Umschaltknöpfe werden z.B. durch eine rollbare Liste ersetzt. Häufig sind nun komplizierte manuelle Ausrichtungen der

Dialogelemente vorzunehmen, unter Umständen ist auch der Code zur Kopplung von Schnittstelle und Anwendung neu zu schreiben.

Auch in CLIM können Dialogfenster auf der Ebene der speziellen Interaktionsobjekte (gadgets) erstellt werden. CLIM unterstützt zur Vermeidung des Portabilitätsproblems aber noch einen etwas allgemeineren Ansatz. Anstatt Interaktionsobjekte auf der Ebene der Baukastenobjekte zu wählen, kann der Schnittstellenprogrammierer die benötigte Interaktionseinheit auf einer „tieferen“ Ebene bestimmen, d.h. es wird die Funktion und nicht die Form des benötigten Interaktionsobjekte angegeben. Ein Beispiel ist eine Auswahl von Objekten. Man spezifiziert, daß eine Eingabe vom Präsentationstyp `member` erwartet wird. `member` ist ein sog. Präsentationstypkonstruktor. Weitere von CLIM bereitgestellte Konstruktoren sind `member-sequence`, `alist`, `completion`, etc. Die Idee besteht nun darin, zur Eingabe von Werten (mit einem bestimmten Präsentationstyp) eine Abbildung auf den jeweiligen Typ der vom Wirtssystem verwendeten Interaktionsobjekte vorzunehmen und entsprechende Objekte automatisch zu erzeugen und anzuordnen.[26]

Relevanz für HAMVIS

Eine mögliche Abbildung wäre die Realisierung der oben betrachteten Auswahl durch gekoppelte Umschaltknöpfe. Die Abbildung kann durch den Oberflächenprogrammierer gesteuert werden, indem er für die Eingabeoperation eine bestimmte *Sicht* (view) vorgibt. Durch die Beschreibung der *Funktion* eines Eingabeelements, versucht CLIM, portablere Oberflächen zu erzielen. Bei der Definition von Interaktionselementen verfolgt HAMVIS einen ähnlichen Gedanken. Für Aktionen des Benutzers wird nicht direkt die Graphik an der Oberfläche spezifiziert, sondern es wird die Funktion der Benutzerinteraktion in den Vordergrund gerückt (z.B. „Auswahl zur weiteren Bearbeitung“, „Verschiebung“ oder „Lokalisierung“ usw.). Allerdings sind nicht alle Aktionen des HAMVIS-Aktionenmodells sinnvoll mit jeder (geometrischen) Darstellung der Anwendungsobjekte durchführbar. Auch für diese Aspekte muß auf der UIMS-Ebene eine Abbildung gefunden werden. Eine wichtige Abstraktion, die hierfür verwendet wird, ist das UIMS-Konzept der Sicht.

Sichten

In CLIM sind Sichten (views) Objekte erster Klasse. Sichtenklassen sind in einer taxonomischen Hierarchie angeordnet. Eine Sicht ist eine Instanz einer speziellen Sichtenklasse. Je nach verwendeter Sicht kann die Präsentation und Eingabe von Werten mit verschiedenen Techniken erfolgen (z.B. Eingabe eines numerischen Wertes per Tastatur oder über einen Schieberegler).

Vordefinierte Sichtenklassen in CLIM sind z.B. `textual-view` und `gadget-view` (z.B. mit Unterklasse `slider-view`). CLIM unterstützt verschiedene Wege der Präsentation und der Eingabe von Objekten über Sichtenklassen. Sichten sind nicht relevant für den Ableitungsmechanismus der Präsentationstypen, sondern bestimmen nur das visuelle Aussehen der zur Interaktion verwendeten Eingabe- bzw. Ausgabeobjekte. Für Standardinteraktionsbausteine sorgen die von CLIM bereitgestellten Sich-

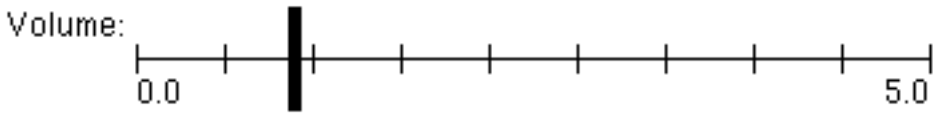
26. CLIM definiert weiterhin die Präsentationtypkonstruktoren (Metapräsentationstypen) `and`, `or` und (mit Einschränkungen) `not` und erlaubt die Verwendung eines allgemeineren `satisfies` Prädikats. Nicht für alle Kombinationen von Präsentationstypen sind aber vordefinierte Abbildungen in Interaktionsobjekte des Wirtssystems vorgegeben.

ten für eine Anpassung der Interaktionsformen an das Wirtssystem, in das eine Anwendung eingebettet wird (z.B. Macintosh oder Motif).

Präsentation und Akzeptierung von Werten

Eine Präsentation assoziiert graphische Ausgaben mit anwendungsspezifischen Objekten und wird durch einen Präsentationstyp definiert. Ein spezieller Präsentationskontext gruppiert mehrere „einfache“ graphische Objekte (oder andere Präsentationen) in einer Ausgabeaufzeichnung. Ein solcher Präsentationskontext kann dynamisch aufgespannt werden oder durch eine sogenannte Präsentationsmethode festgelegt werden. Präsentationsmethoden diskriminieren über das Anwendungsobjekt, den Präsentationstyp, den Ausgabestrom und die Sicht.[27]

```
(accept 'number
        :view '(simple-slider-view
                    :width 300
                    :tick-number 10
                    :min-value 0.0
                    :max-value 5.0)
        :prompt "Volume"
        :default (volume ...)
        :stream ...)
```




```
(accept 'number
         :view +text-field-view+
        :prompt "Volume"
        :default (volume ...)
        :stream ...)
```

Abbildung 13. Beispiele für die Akzeptierung von Werten mit unterschiedlichen Eingabetechniken.

Bei der Eingabe (Akzeptierung) kann die gewünschte Eingabetechnik über die Sicht angegeben werden. Abbildung 13 zeigt zwei Beispiele für die Eingabe einer Zahl. Der Aufruf von `accept` beschreibt die *Funktion* des gewünschten Interaktionsobjekts, d.h. es wird definiert, welcher Präsenta-

27. Eine Präsentationsmethode ist im wesentlichen eine normale Methode, jedoch wird eine Erweiterung von CLOS zur Methodendiskriminierung benutzt. Man beachte, daß parametrierte Präsentationstypen behandelt werden müssen. Zur Einführung in den von CLOS verwendeten Mechanismus zur Multimethodendiskriminierung siehe [146].

tionstyp geliefert werden soll (hier `number`). Neben einer Eingabeaufforderung (prompt) wird noch ein initialer Wert (default) angegeben.

Die über die Sicht spezifizierte Eingabetechnik kann durch CLIM automatisch an das Wirtssystem angepaßt werden.[28] Für das Textfeld wurde in diesem Fall Motif verwendet. Spezielle Eingabetechniken können durch sog. Akzeptierungsmethoden bereitgestellt werden. In Abbildung 13 wurde oben ein Schieberegler zur Eingabe der Zahl verwendet.

Verwendung der Konstrukte im Kontext von HAMVIS

Die Idee der Akzeptierungsmethoden kann im HAMVIS-Kontext für Präsentationstypen des von HAMVIS bereitgestellten, anwendungsübergreifenden Grundmodells angewendet werden. HAMVIS stellt vordefinierte Sichtenklassen für die Standardperspektiven von graphischen Objekten bereit (Seitenansicht, Aufsicht, usw.) und definiert entsprechende Präsentationsmethoden für graphische Objekte. Die Ausgabe aller Objekte eines Teilfensters erfolgt mit einer bestimmten Sicht (z.B. einer Grundrißsicht). In der Präsentationsmethode wird auf die geometrischen Daten zugegriffen, die in der jeweiligen Sicht relevant sind. Um z.B. eine Benutzerhandlung vom Konzept „Verschiebung“ zu ermöglichen, definiert HAMVIS eine Akzeptierungsmethode für den Typ „Position mit Einschränkung“ und die Sicht „Aufsicht“.

Nehmen wir an, im Laufzeitsystem einer Anwendung sei für eine „räumliche Verschiebung“ ein CLIM-Kommando „Verschiebe Objekt“ z.B. mit zwei Parametern definiert. Als Parameter werden das verschobene Objekt und die neue Position (eigentlich eine Positionseinschränkung mit Standardwert) übergeben. Nehmen wir nun an, der Benutzer der XKL-Anwendung soll zur Laufzeit der Anwendung eine Küche anklicken (siehe Abbildung 14), um die Verschiebung direktmanipulativ durchzuführen. In CLIM-Terminologie wird beim Anklicken eine entsprechende Präsentation-nach-Kommando-Abbildungsfunktion evaluiert. Es muß daher also durch HAMVIS eine Präsentation-nach-Kommando-Abbildung für den Präsentationstyp „räumliches Objekt“ auf das Kommando „Verschiebe Objekt“ generiert werden. Im Rumpf der Abbildungsfunktion wird die `accept`-Funktion für die Positionsbeschreibung des angeklickten Kabinenobjekts aufgerufen. Über den Vererbungsmechanismus von CLOS wird die von HAMVIS bereitgestellte Akzeptierungsmethode und damit die geeignete Eingabetechnik gefunden. Als Standardwert wird die „alte“ Position der Küche übergeben. Die `accept`-Methode aus der Bibliothek zeichnet den Verschiebebereich (dieser kann aus der Positionseinschränkung ermittelt werden) und sorgt dafür, daß die notwendigen graphischen Rückkopplungen für die Interaktion bereitgestellt werden (siehe Abbildung 14).

28. Eine Sicht kann beim Aufruf von `accept` durch eine Beschreibung und nicht direkt als Objekt übergegeben werden. In Abbildung 13 wurde z.B. eine Liste bestehend aus einem Klassennamen und Initialisierungsargumenten angegeben. Die Funktion `accept` erzeugt aus diesen Angaben eine Sichteninstanz.

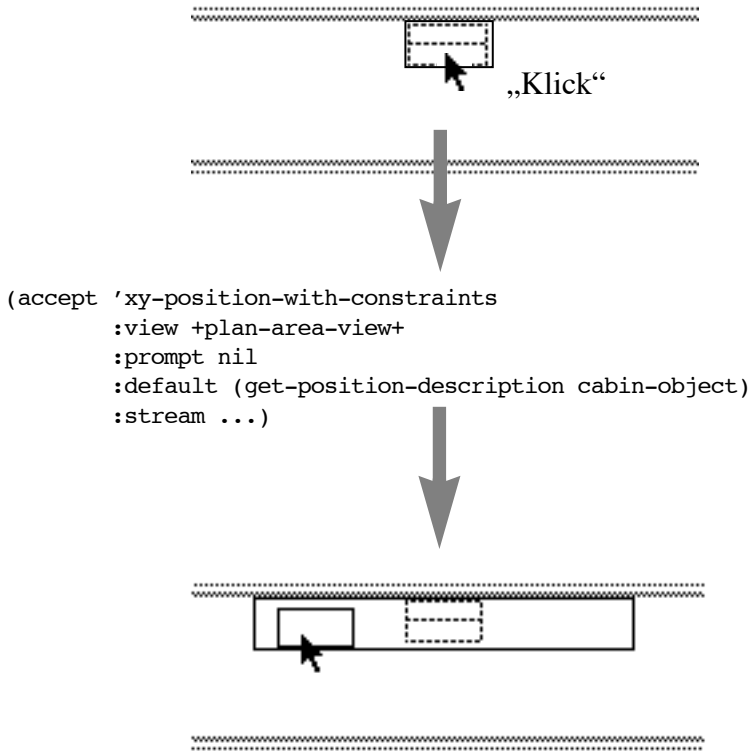


Abbildung 14. Verschiebung einer Küche. Das Anklicken eines Einrichtungsgegenstande führt dazu, daß ein neues Objekt zur Beschreibung einer Position auf der xy-Ebene erzeugt wird. In der `accept`-Methode werden die dazu benötigten Interaktionswerkzeuge bereitgestellt (unterer Teil der Abbildung). Durch das gezeichnete Rechteck werden die möglichen Positionen angezeigt. Es ist nicht möglich, ein Objekt aus dem Plazierungsbereich herauszubewegen.

Der Code der Akzeptierungsmethode von HAMVIS enthält die komplexen Zeichenoperationen, die für diese Interaktionsform nötig sind. In dem hier betrachteten Beispiel werden detaillierte Koordinateninformationen gezeigt und bei der Verschiebung aktualisiert. Es handelt sich als schon um eine recht spezielle Verschiebung. Denkbar zur Realisierung eines anderen Verschiebungskonzepts für eine sehr exakte Positionierung wäre z.B. die Eingabe von Koordinaten über ein Texteingabefeld (siehe Abbildung 13).

Interaktionstechniken zur Umsetzung von Handlungen lassen sich für eine Anwendungsklasse generisch bereitstellen: Für die im HAMVIS-Grundmodell modellierten Aktionenkonzepte wie z.B. „Position mit Einschränkung“ werden Akzeptierungsmethoden bereitgestellt. Die direktmanipulative Eingabe eines solchen Wertes ist vergleichbar mit der Eingabe einer Zahl über einen Schieberegler wie in Abbildung 13.

Die Menge der Interaktionsobjekte (gadgets) muß für die Anwendungsklasse des HAMVIS Grundmodells durch den HAMVIS-*Entwickler* entsprechend erstellt werden. Die Interaktionsobjekte stehen dann für den Oberflächenentwickler (HAMVIS-*Benutzer*) über die Aktionenkonzepte zur Verfügung. Zur Spezifikation der Konzepte von anwendungsspezifischen Domänenobjekten werden die durch das HAMVIS-Grundmodell bereitgestellten Konzepte als Oberkonzepte angegeben. Wenn zur Laufzeit *domänenspezifische* Objekte an die Akzeptierungsmethoden übergeben werden, so werden die „rich-

ten“ Akzeptierungsmethoden über den Vererbungsmechanismus gefunden. Mit einem Grundmodell für eine Anwendungsklasse werden also die entsprechenden Akzeptierungsmethoden als Softwarebibliothek für eine Anwendungsklasse bereitgestellt. Nicht für jede Handlung sind jedoch sinnvoll Methoden für alle Sichten definierbar. Für Verschiebehandlungen beispielsweise sollten sich die Objekten möglichst nicht überlappen. Die Softwarebibliothek kann also „Dienste“ nur für bestimmte Objektkonzepte und Sichten bereitstellen. Das HAMVIS-Rahmensystem bietet dem Entwickler einer speziellen Anwendung eine Unterstützung bei der Auswahl von Sichten, bei der Kombination von Interaktionsobjekten usw., so daß Fehler im Laufzeitsystem vermieden werden (z.B. daß zur Laufzeit eine Akzeptierungsmethode für eine spezielle Sicht nicht gefunden werden kann). Näheres hierzu schildert Kapitel 3.

Geschichte, Status und Weiterentwicklung der Ideen von CLIM

Inspiriert durch die Arbeiten von Zdybel et al. an dem AIPS (Advanced Information Presentation System [347]), von Friedell [94] und von Shneiderman [285] hat Ciccarelli die ersten Ideen bezüglich der Verwendung von Präsentationen entwickelt (publiziert in seiner Dissertation von 1984 [49]). Während AIPS Modellierungsstrukturen von KL-ONE verwendete, um Domänendatenstrukturen und Darstellungsdatenstrukturen explizit zu repräsentieren (physikalische Objekte, Fahrzeuge, Schiffe usw.), sollte das Präsentationssystem von Ciccarelli vollständig domänenunabhängig sein.

Ciccarelli verfolgte die Idee, daß ein Benutzer eine explizit modellierte Präsentationstruktur manipuliert (diese ist vergleichbar mit den hierarchisch geschachtelten Ausgabeaufzeichnungen von CLIM). Interaktionsgesten auf Elementen der Präsentationsstruktur werden übersetzt in Kommandos der Anwendung (in Ciccarellis Arbeiten ein Datenbanksystem). Nachdem ein Kommando abgearbeitet ist, wird die Präsentationsstruktur und damit die Bildschirmausgabe reorganisiert. Die Präsentationsstruktur wird abgebildet auf ein Graphikpaket (Ciccarellis System wurde auf einer frühen Lisp-Maschine implementiert). Weiterhin führte er den Begriff eines Präsentationsstils (presentation template) ein, der vergleichbar zu dem Begriff der Sicht in CLIM ist.

Lieberman stellte 1985 eine objektorientierte Implementation seines Oberflächenkonstruktionssystems EZWIN vor, bei dem Präsentationsobjekte als eine visuelle Repräsentation von Anwendungsobjekten verwendet wurden [175]. Operationen auf Präsentationsobjekten bedingen wiederum die Erzeugung von Kommandoobjekten. Schon recht früh wurden Benutzerinteraktionen also explizit in Form von Kommandoobjekten repräsentiert. Lieberman entwickelte die Idee, daß vorherige Ausgaben maussensitiv werden sollten und als Eingabe für nachfolgende Kommandos dienen können. Um anzuzeigen, welche Operationen mit der Maus durchgeführt werden können, erzeugte EZWIN eine Hervorhebung des graphischen Objekts, wenn sich die Maus auf dem Objekt befindet.

Ein Pioniersystem, das diese Ideen in die Praxis umsetzte, ist „Dynamic Windows“, das Fenstersystem des Betriebssystem der Symbolics Lisp-Maschine. Die ersten Versionen von „Dynamic Windows“ sind mit Genera 7.0 etwa 1986 erschienen [306]. Präsentationen wurden in Genera für die Gestaltung des Lisp-Programmierumgebung und für die Eingabe von Kommandos verwendet („Listener“, heutzutage als „Shell“ bezeichnet). Ein Kommando wie `Show Directory host:>dir>*.lisp` druckt Dateiinformationen in den Listener. Wenn später ein Kommando `Edit File` eingegeben wird, so sind die Präsentationen der Lisp-Dateien maussensitiv und können angeklickt werden.[29] Weiterhin ist es möglich, sich die auf ein Objekt anwendbaren Kommandos per Menü (popup menu) anzuzeigen und z.B. `Edit File` auszuwählen.[30]

CLIM ist ein Nachfolgersystem von „Dynamic Windows“, ist aber auch für Lisp-Systeme unter UNIX und anderen Betriebssystemen verfügbar. Einige andere Systeme haben die Ideen von „Dynamic Windows“ aufgegriffen (siehe den „Presenting Listener“ für den XEmacs-Editor als Teil der Entwicklungsumgebung von „Allegro Common Lisp“ und auch Teilkomponenten des XEmacs selbst). Allerdings werden von diesen „Nachbauten“ keine anwendungsspezifische Graphiken unterstützt, sondern nur Text, und es werden keine Informationen zu Präsentationstypen verwaltet. Die Architektur von CLIM ist wesentlich konsistenter aufgebaut.

Allerdings ist CLIM in der momentan vorliegenden Fassung (Version 2.0/2.1) nicht geeignet für Graphiken, bei denen hohe Performanz benötigt wird z.B. für Animation und Video (Multimedia-Anwendungen).[31] Jüngste Ansätze versuchen, die Ideen der Präsentationstypen in Präsentationen zu integrieren, die über das Netzwerk versandt werden (WWW). Wenn Typinformationen mit den möglichen Eingaben verknüpft werden, kann das WWW-Darstellungsprogramm auf dem lokalen Rechner ein erweitertes Parsing durchführen. Es ist keine Kommunikation mit dem Server-Rechner des Dokumentes vonnöten. Wenn ein Objekt in einem bestimmten Eingabekontext keine sinnvolle Eingabe darstellt, so ist es gar nicht mausssensitiv.

Wie schon diskutiert, wird bei dem CLIM-Ansatz zur Erreichung von Portabilität nicht die Form, sondern die Funktion von Interaktionselementen bestimmt. Bei Schnittstellenbaukästen liegt hingegen der Schwerpunkt auf der Form. Die Funktionalitäten von CLIM kommen am besten bei komplexeren Anwendungen mit anwendungsspezifischen interaktiven Graphiken zum tragen. Für kleinere Oberflächen mit formularähnlichen Eingabefenstern sind andere UIMS-Ansätze mit Schnittstellenbaukästen vermutlich besser geeignet, da sie kompakter sind, d.h. weniger Ressourcen verbrauchen.

2.1.5 Zusammenfassung

Für viele Auftraggeber ist die Oberfläche mit der Plazierung von Interaktionsobjekten mit einem Schnittstellenbaukasten fertig. Die Erfahrung zeigt, daß dann die Arbeit erst beginnt. Interaktive Schnittstellenbaukästen stellen zwar eine Unterstützung für die Erstellung einfacher, statischer Oberflächen bereit. Inzwischen sind auch kommerzielle Produkte mit guten Leistungsvermögen verfügbar. Wenn Anwendungsobjekte aber dynamisch erzeugt werden und die graphische Darstellungsform von Informationen abhängt, die nur zur Laufzeit der Anwendung verfügbar sind, sind Schnittstellenbaukästen mit Zeicheneditoren für graphische Objekte nicht mehr (direkt) anwendbar. Weiterhin ist zu beachten, daß die direkte Spezifikation von Form und Position von Dialogelementen Probleme bei der Portabilität bedeutet. Für die notwendige dynamische Neuanordnung von Dialogelementen (z.B.

---

29. Tatsächlich kann mehr als eine Datei als Parameter an das Kommando `Edit File` übergeben werden, denn es wird eine Sequenz von Dateien akzeptiert (siehe den Präsentationtypenkonstruktor `sequence`). „Dynamic Windows“ stellt hierzu einen Mechanismus bereit, mehrere Objekte anzuwählen (rubber-banding rectangle).

30. Die Verfügbarkeit von `Edit File` als ein Menüelement ist allein durch die Deklaration eines Kommandos gegeben. Wie in CLIM, werden die Menüs intern von „Dynamic Windows“ verwaltet und brauchen nicht „von Hand“ programmiert zu werden.

31. Die Entwicklung von CLIM ist noch nicht abgeschlossen. Es gibt zur Zeit keine Abstraktionen zur Koordination von mehreren Anwendungsrahmen, die jeweils einer Anwendung zuzuordnen sind. Dieses ist auf der semantischen Ebene allerdings auch sehr schwierig zu definieren, so daß anwendungsübergreifende Abstraktionen vermutlich kaum gefunden werden können. Da CLIM versucht, portable Oberflächen zu unterstützen, fehlt der direkte Zugriff auf bestimmte Dienste des Betriebs- bzw. Wirtsfenstersystems.

wenn in einem bestimmten Wirtssystem von einem kurzsichtigen Benutzer ein großer Font verwendet wird) werden deklarative Beschreibungen für das Layout und für die Aufgabe von Dialogelementen benötigt. Zu einem gewissen Grad können diese Angaben heute schon mit Hilfe von Schnittstellenbaukästen gemacht werden. Es sind dann allerdings Erweiterungen mit textuellen Beschreibungssprachen vorgesehen, um eine adäquate Ausdrucksstärke zu erreichen.

Die interaktive Plazierung von (prototypischen) Dialogelementen in einem Fenster stellt nicht genügend Informationen „auf tiefer Ebene“ bereit, die benötigt werden, um das dynamische Verhalten von Oberflächenelementen unter Berücksichtigung des Anwendungszustands zu realisieren. Daher wurden modellbasierte Schnittstellenbaukästen entwickelt, mit denen z.B. spezielle Interaktionsbausteine („templates“) bereitgestellt werden, deren Vor- und Nachbedingungen explizit modelliert werden können. Die für modellbasierte Schnittstellenbaukästen entwickelten Spezifikationen für Aktionen und Eingabetechniken beziehen sich allerdings auf graphische Objekte, d.h. die Verbindung zu Anwendungsobjekten wird nicht oder nur indirekt hergestellt, da Vor- und Nachbedingungen für Aktionen sich auf graphische Attribute und nicht auf Attribute von Anwendungsobjekten beziehen. Modellbasierte Schnittstellenbaukästen sind als Prototypen realisiert.

Inzwischen wurden UIMS ebenfalls weiterentwickelt und es stehen in kommerziellen Produkten Abstraktionsebenen bereit, mit denen viele notwendige Dienste für die Programmierung von Oberflächen zur Verfügung gestellt werden. Die für die Oberflächenerstellung relevanten Fachtermini sind mittlerweile recht komplex. Meine Erfahrungen zeigen, daß sie für die meisten Programmierer nicht intuitiv zugängig sind. Im Gegensatz zu Schnittstellenbaukästen geht der Trend (u.a. aus Gründen der Portabilität) hin zu einer abstrakteren Beschreibung der in Anspruch genommenen Dienste. In dieser Arbeit habe ich die wesentlichen UIMS-Konzepte anhand des UIMS CLIM präsentiert. Dabei wurde aus der Perspektive eines UIMS geschildert, wie die von HAMVIS vorgesehenen Konzepte zur Modellierung von Benutzeraktionen in oberflächennahe Strukturen umgesetzt werden. Für jede Applikation definiert HAMVIS einen CLIM-Anwendungsrahmen, für jede Aktivität wird in diesem Anwendungsrahmen ein Layout von Teilfenstern vorgesehen. Nebenläufige Aktivitäten benötigen jeweils einen eigenen Anwendungsrahmen. Zusammengesetzte Handlungen werden durch Kommandos realisiert, wobei Kommandos ggf. mit Hilfe von Präsentation-nach-Kommando-Abbildungen interaktiv ausgelöst werden können. Falls dieses nicht möglich oder nicht gewünscht wird, so können Kommandos durch textuelle Eingabe initiiert werden. Die Ebene der Präsentation-nach-Kommando-Abbildungen stellt bei CLIM die Ebene der Programmierung dar. HAMVIS stellt auch auf dieser Ebene noch deklarative Modelle bereit (Ebene der Konzepte für elementare Benutzeraktionen), die von dem Entwicklungsteam zur Spezifikation einer Aktionendekomposition für eine Anwendung verwendet werden. Interaktionstechniken werden durch das Grundmodell für eine Anwendungsklasse bereitgestellt.

Offensichtlich sind die fortgeschrittenen Konzepte und Abstraktionen von CLIM besonders geeignet für die Realisierung des von HAMVIS für eine Anwendung zu generierenden Laufzeitsystems. Die Verwendung eines anderen UIMS scheint aber ebenfalls möglich. CLIM versucht m.E., die für die Erstellung von Benutzungsschnittstellen notwendigen Abstraktionen explizit zu modellieren. Aus Gründen der Portabilität wird schon in CLIM zum Teil nicht die Form von Oberflächenelementen definiert, sondern deren Funktion. Für die immer komplexer werdenden Probleme bei der Oberflächenentwicklung ist dieses ein notwendiger Schritt. Die mit HAMVIS eingeführten elementaren

Benutzeraktionen und die Konzepte zur expliziten Dialogmodellierung und Präsentationsstrukturierung durch Markierungen führen diese Gedanken fort.

Der Code für komplexe direktmanipulative Interaktionsformen mit graphischen Elementen wird durch verschiedene Akzeptierungsmethoden für Konzepte des HAMVIS-Grundmodells bereitgestellt (siehe z.B. die Realisierung einer Verschiebung in Abbildung 14). Wenn jedoch eine Vielzahl von Akzeptierungsmethoden für verschiedenste Einsatzzwecke angeboten wird, so ist die Gefahr groß, daß der Anwendungsentwickler nicht weiß, ob für seinen Einsatzzweck schon eine Methode existiert. Die „Infrastruktur“ einer großen Softwarebibliothek von Klassen und Methoden ist für einen Schnittstellenentwickler nicht unbedingt leicht erlern- und anwendbar. Erfahrungen mit Smalltalk-Systemen belegen dieses ebenfalls in eindeutiger Weise. Einem Nutzer einer Klassenbibliothek ist vielfach nicht klar, ob schon ein verwendbarer Softwarebaustein für sein Anwendungsproblem existiert. Wir haben außerdem in der Einleitung gesehen, daß die Anwendbarkeit einer bestimmten Akzeptierungsmethode z.B. durch Wahl einer bestimmten Sicht für verschiedene Aktionen erst sichergestellt werden muß. Es muß zur Entwicklungszeit einer Anwendung sichergestellt werden, daß zur Laufzeit auch eine Methode bzgl. der Aktualparameter gefunden werden kann. HAMVIS modelliert die bereitgestellten Dienste explizit mit Hilfe von Modellen für Benutzeraktionen. Einige Benutzeraktionen werden durch Akzeptierungsmethoden realisiert, andere werden durch Präsentation-nach-Kommando-Abbildungen umgesetzt. Ob eine solche Umsetzung in direktmanipulative Interaktionsformen möglich ist, wird durch HAMVIS bestimmt und dem Anwendungsentwickler im Dialogmodell vorgeschlagen.

Das Ziel der Modellierung von interaktiven Systemen auf der Ebene der Endbenutzeraktionen ist, einen Schnittstellenentwickler über die iterative Beschreibung von Aktionen an die bereitgestellten „Dienste“ der Bibliothek von generischen Werkzeugen heranzuführen. Die HAMVIS-Aktionenmodelle enthalten Deklarationen zur Sichteneinschränkung, so daß im Dialogmodell nur solche Sichten und Akzeptierungsaufrufe „eingeplant“ werden, für die zur Laufzeit auch Methoden gefunden werden können. HAMVIS konstruiert also *keine* neuen Interaktionseinheiten für eine spezielle Anwendung („from first principles“), sondern erlaubt die systematische Auswahl und Komposition von Interaktionsobjekten aus einer Bibliothek von Eingabetechniken einer Anwendungsklasse. Zu beachten ist auch, daß für die Interaktionswerkzeuge auch das für die menschliche Informationsverarbeitung notwendige „Umfeld“ geschaffen werden muß (siehe hierzu die HAMVIS-Komponenten zur Dialog- und Präsentationsstrukturierung).

Dieses Kapitel zeigt, wie die von HAMVIS und anderen Ansätzen verwendeten Modellierungsformen (Aktionenzerlegung, Benutzerhandlungen mit Domänenobjekten) auf der technischen Ebene auf Abstraktionen eines UIMS abgebildet werden. Aus softwaretechnischer Sicht strukturieren die von HAMVIS bereitgestellten Aktionenmodelle die zur Verfügung stehende Bibliothek von `present`- und `accept`-Methoden für Konzepte einer Anwendungsklasse (wie z.B. Layoutsysteme) und machen die erweiterten UIMS-Dienste für einen Anwendungsentwickler leichter verfügbar. HAMVIS kann also aus einer Bottom-up-Perspektive als UIMS-Erweiterung betrachtet werden.

Neben der in diesem Kapitel verfolgten technischen Sicht der Umsetzung auf UIMS-Dienste können die Aktionenmodelle von HAMVIS auch top-down aus der Sicht der Mensch-Computer-Interaktion betrachtet werden. Wir werden im nächsten Abschnitt sehen, daß mächtige UIMS wie CLIM als eine praktische Realisierung von eher theoretischen Arbeiten aus dem MCI-Bereich angesehen werden können. Aus der MCI-Perspektive wird jedoch deutlich, daß die in Kapitel 1.3 skizzierten Modellierungsformen von HAMVIS nicht nur aus der Software-Sicht eine Vereinfachung der Programmierung

darstellen, sondern auch einen Beitrag zur methodischen Anwendungsentwicklung und Systemgestaltung leisten.

2.2 Aspekte des Forschungsgebiets Mensch-Computer-Interaktion

In diesem Kapitel wird ein kurzer Abriß über die für HAMVIS relevanten Arbeiten aus dem Gebiet der Mensch-Computer-Interaktion gegeben. Aus Sicht der Mensch-Computer-Interaktion kann eine Anwendung aus verschiedenen Perspektiven modelliert und konstruiert werden (Maaß, Oberquelle [185], siehe auch die Diskussion von Reiterer in [255]). Die Perspektiven verwenden unterschiedliche Metaphern, die jedoch nicht disjunkt sind, sondern jeweils verschiedene Aspekte der Systemorganisation und Interaktion hervorheben:

- Maschinenperspektive

  Aus der Maschinenperspektive heraus interpretiert der Systemdesigner einen Computer als Blackbox mit vordefinierten Ein-Ausgabeverhalten und Fokus auf Funktionalität, Effizienz usw. Das Ziel des Systemdesigns ist die Eliminierung von menschlicher Arbeit, wo immer dieses möglich ist. Die Maschinenperspektive vernachlässigt die Fähigkeiten eines Benutzers einer Anwendung und unterstützt seine Stärken nicht.

- Systemperspektive

  Benutzer und Computer werden als interagierendes System innerhalb der Informationsverarbeitungs- und Kommunikationsstruktur einer Organisation betrachtet. Ähnlich wie bei der Maschinenperspektive werden Benutzer als Datenempfänger, Datenverarbeiter und Dateneingeber für andere Systeme betrachtet. Das Hauptziel der System- und Interaktionskonzeption ist die Beschleunigung der Übertragung von Daten bei reduzierter Fehlerrate (mit Begriffen wie optimaler Informationsfluß, minimale Redundanz und effektiver Verteilung von Datenverarbeitungsaufgaben zwischen Mensch und Computer).

- Werkstattperspektive (auch Werkbank-Perspektive oder Werkzeug-Material-Perspektive genannt)

  Bei der Werkstattperspektive werden Objekte visuell dargestellt (eventuell nach vorheriger Transformation bzw. „Materialisierung“). Objekte können in ein oder mehreren Arbeitskontexten bzw. Arbeitsräumen (workspaces, rooms) bearbeitet werden [191]. Die dargestellten Objekte (oder die „Materialien“) können mit bestimmten Werkzeugen manipuliert werden. Verschiedene Werkzeuge, die durch den Benutzer interaktiv ausgewählt werden können, unterstützen verschiedene Arten der Manipulation (z.B. in Zeichenprogrammen oder CAD-Systemen). Eine Instantiierung dieses Schemas ist auch die Schreibtisch-Metapher für Betriebssysteme. Wichtig ist, daß Metaphern nur die Grundstruktur der Werkstattperspektive bestimmen. Auf einem Computer können auch rein fiktive Operationen und Werkzeuge unterstützt werden.

- Medienperspektive

  Als Erweiterung der Werkstattperspektive können Computer als Medium betrachtet werden. Im Prinzip ist ein Computer sogar als Metamedium anzusehen, da verschiedene Ausprägungen simuliert werden können (Bücher, Videogeräte, CD-Spieler etc.). Unterschiedliche Medien werden eingesetzt zur Unterstützung der Kommunikation von Mensch und Computer oder auch zwischen

verschiedenen Menschen unter Verwendung eines Computers (CSCW: computer-supported cooperative work). Durch die Imitation von verschiedene Medien kann der Computer für den Benutzer transparent werden. Die Benutzer haben die Illusion, direkt auf ihrer Anwendungs- oder Problemebene zu agieren. Bei der Medien- und Werkstattmetapher ergeben sich die gezeigten Objekte direkt aus den Interaktionen des Benutzers, zumindest aus der Benutzerperspektive gibt es keine „Instanz“, die die in einer bestimmten Interaktionssituation gezeigten Darstellungen geplant hat. Dieses ist bei der Kommunikationsperspektive der Fall.

- Kommunikationsperspektive

  Bei der Kommunikationsperspektive gibt es zwei verschiedene Ausprägungen: die Dialogpartnerperspektive und die Perspektive der formalen Kommunikation.

  Aus der *Dialogpartnerperspektive* zeigt ein Computer ein kommunikatives Verhalten, das mit dem eines Menschen verglichen werden kann. Die Dialogpartnerperspektive wird insbesondere durch die Agentenmetapher der Computerinteraktion vertreten. In diesem Paradigma wird der Computer als Agent[32] angesehen, der sich durch einen bestimmten Charakter und bestimmte Eigenschaften auszeichnet wie z.B. Anwortbereitschaft, Kompetenz, Verfügbarkeit, Lernfähigkeit und adaptives Verhalten, Erklärungs- und Kooperationsbereitschaft. Agenten können aktive Kommunikationspartner sein (z.B. in natürlichsprachlichen Systemen) oder eine untergeordnete, passive Rolle spielen (siehe die Begriffe „Assistenzcomputer“ [66] oder, etwas allgemeiner, „interface agent software“). Das Verhalten von Agenten sollte auf die Erwartungen von menschlichen Kommunikationsteilnehmern abgestimmt sein.[33]

  Der aus der Agentenmetapher resultierende Anthropomorphismus hat viele kritische Argumente gegen die Kommunikationsperspektive der Mensch-Computer-Interaktion ausgelöst (cf. Shneiderman [286], zitiert nach Reiterer [255], S. 36f). Die Gegner der (KI-orientierten) Dialogpartnerperspektive argumentieren wie folgt: Falls der Benutzer den Computer nicht als Artefakt betrachtet, ist er eventuell nicht auf ein unflexibles und eingeschränktes Systemverhalten vorbereitet, das der Computer als Artefakt in einigen Fällen zeigen wird. Benutzer sollten daher in die Lage versetzt werden, sich ein klares, zur Vorhersage geeignetes Bild des Systemverhaltens aufbauen zu können und zwar sowohl mit den Schwächen wie auch den Stärken des Artefaktes. Die Stärken kommen sogar unter Umständen bei quasi-natürlichen Interaktionsformen nicht adäquat zum tragen. Bei einer artifiziellen Interaktionsform können die Fähigkeiten von Computern erst richtig verfügbar

32. Der Begriff „Agent“ des Deutschen ist nicht direkt mit dem Begriff „agent“ aus dem Englischen identisch. Ich verwende ihn aber in Ermanglung einer passenderen Übersetzung. Von einigen Autoren wird auch der Begriff „Wichtel“ verwendet. Wichtel kommen von der Wortbedeutung eher den ursprünglichen „agents“ aus Minskys „Society of Mind“ entgegen. Das wichtige Prinzip der Emergenz im Verhalten von Minskys „agents“ wird aber auch durch den Begriff „Wichtel“ kaum besser reflektiert.

33. Allerdings stellen Bonar und Liffick heraus: „an interface that merely matches the user’s expectations is stuck with those expectations. In particular, the user can never go beyond those expectations to use more powerful facilities than that expectation allows“ ([24], S. 132). Mit anderen Worten: Frühe Automobile ähnelten Kutschen, d.h. sie waren auf die Erwartungen der zeitgenössischen Fahrer abgestimmt. Heutzutage ist klar, daß es bessere Techniken gibt, ein Automobil zu konstruieren.

gemacht werden. Geleitet durch diese Überlegungen hält Shneiderman die Dialogpartnerperspektive (bei denen Computer ähnliche Kommunikationsformen verwenden wie Menschen) für kontraproduktiv.

Die Gegner der Dialogpartnerperspektive betonen den Unterschied zwischen Mensch-Mensch- und Mensch-Computer-Interaktion und schlagen eine sogenannte *formale Kommunikationsperspektive* vor.[34] Ein eingeschränktes Kommunikationsverhalten wird durch Oberflächendesigner vorgeplant, so daß das Systemverhalten an die Erwartungen und Fähigkeiten der Benutzer angepaßt ist. Die Oberfläche vermeidet dabei den Anschein eines (kommunikativen) Agenten.

Während die Maschinen- und Systemperspektiven kaum noch für das Systemdesign verwendet werden, sind die Werkbankmetapher und ihre Erweiterung, die Medienmetapher, beliebte Konstruktionsleitlinien. Die Kommunikationsperspektive ist in einer gewissen Weise orthogonal zu den anderen Perspektiven und insbesondere relevant, wenn die Präsentation von Informationen auf dem Bildschirm automatisiert werden soll (z.B. bei Informationssystemen).

Nicht in allen Anwendungen können die Objekte, die auf dem Bildschirm erscheinen sollen, durch den Benutzer selbst spezifiziert werden (z.B. durch direktmanipulative Operationen wie Öffnen eines Ordners oder Öffnen eines Werkzeugkastens). Das bedeutet, auch wenn ein System aus der Werkbankperspektive (s.o.) heraus konstruiert wird, müssen die Informationen, die für Entscheidungsprozesse relevant sind, dem Benutzer auf adäquate Art und Weise präsentiert werden. Mit anderen Worten: Auch bei einer Werkbankperspektive bzw. bei einer formalen Interaktionsperspektive sind zur Gestaltung der Mensch-Computer-Interaktion Gesichtspunkte eines „Dialogs“ zu berücksichtigen. Die Wissensquellen, die im Kontext der formalen Interaktionsperspektive zur Oberflächen- und Visualisierungskonstruktion herangezogen werden, sind weitgehend mit denen aus der Dialogpartnerperspektive identisch. Der Unterschied liegt vielmehr darin begründet, daß bei der Dialogpartnerperspektive der Dialog automatisch erzeugt werden soll und nicht durch einen menschlichen Oberflächendesigner. Dieses bedeutet, daß bei der Dialogpartnerperspektive keine Evaluierung der „Dialogbeiträge“ durch einen menschlichen Oberflächenentwickler erfolgt, d.h. es gibt keine Unterscheidung zwischen Entwicklungszeit- und Laufzeitaktivitäten.

Man kann weiterhin zwischen einer internen und einer externen Sicht der Kommunikationsperspektive unterscheiden. Systeme können aus interner Sicht (aus Sicht des Designers) aus der Dialogpartnerperspektive heraus konstruiert werden, sich aber nach außen hin (aus Sicht des Benutzers) wie ein formales Kommunikationssystem verhalten. Ein konversationales Modell dieser Art für multimodale Interaktionsformen in Informationssystemen wurde z.B. von Thiel, Sitter und Stein vorgeschlagen ([287], [295], [317]).

Aus der Dialogpartnerperspektive findet die Kommunikationsplanung zur Laufzeit der Anwendung statt (unter Betrachtung von Modellen für die Kommunikationspartner, die kommunikative Situation etc.). Im HAMVIS-Szenario wird zwischen Entwicklungszeit (Entwurfszeit) und Laufzeit einer Anwendung unterschieden, d.h. es gibt einen verantwortlichen Designer (der die HAMVIS-Dienste

34. Der Terminus „formale Kommunikationsperspektive“ ist einigermaßen mißverständlich, da Systeme, die nach der Dialogpartnermetapher konstruiert sind, auch mit formalen Techniken modelliert werden. Maaß und Oberquelle verwenden daher den Begriff „algorithmisches Kommunikationsverhalten“ für die formale Kommunikationsperspektive [185].

zur Entwicklungszeit verwendet) und einen Benutzer, der das Anwendungssystem zur Laufzeit verwendet. Ein methodischer Ansatz zur Visualisierungskomposition für Benutzungsschnittstellen muß m.E. Aspekte der Kommunikation und Informationspräsentation zur Entwurfszeit unterstützen, auch wenn zur Laufzeit eine formale Interaktionsperspektive vorherrschend ist.

In der Domäne der graphischen Objektmodellierung und des Layouts von graphischen Objekten wurden von Kochhar, Marks und Friedell verschiedene Ansätze zur Gestaltung der Mensch-Computer-Kooperation einschließlich der Verteilung von Zuständigkeiten klassifiziert [155]:

- Manuelle Ansätze:

  Der Benutzer ist verantwortlich für alle Modellierungs- und Designentscheidungen (z.B. ursprünglicher CAD-Ansatz).

- Einschränkungsbasierte Paradigmen:

  Die manuelle Modellierung wird durch ein Computersystem vervollständigt, das automatisch überprüft, ob bestimmte Einschränkungen erfüllt sind und den Konstruktionsraum für den Benutzer dementsprechend einschränkt.

- Kritiker-basierte Ansätze

  Der Benutzer agiert mit domänenspezifischen Interaktionsobjekten. Die manuelle Arbeit wird observiert und automatisch gegen bestimmte Optimalitätskriterien abgeglichen. Mängel des aktuellen Designobjekts werden dem menschlichen Operateur präsentiert (vgl. die weiter unten diskutierten Arbeiten von Fischer et al.).[35]

- Kooperative oder verbesserungsbasierte Paradigmen:

  Mängel werden nicht nur dem menschlichen Operateur präsentiert, sondern es werden auch mögliche Alternativen automatisch bestimmt und dem Operateur zur Auswahl angeboten.

- Automatische Ansätze:

  Der Computer berechnet ein Design automatisch nach formalen Spezifikationen ohne menschliche Eingreifmöglichkeiten.

Computersysteme für die letzteren Metaphern sind schwieriger zu erstellen. Zu beachten ist, daß nicht in jedem Fall die notwendigen Wissenstrukturen formal repräsentiert werden können (entweder aus Mangel an Ressourcen (Geld und Zeit) oder weil das Wissen nicht auf einfache Weise erfaßt werden kann). Entscheidungsgrundlagen des Kunden im XKL-Szenario sind z.B. kaum faßbar und in Optimalitätskriterien umsetzbar.

Im HAMVIS-Kontext sind zwei Perspektiven wichtig. Auf der einen Seite steht die zu erstellende Anwendung (z.B. XKL). XKL wurde so konzipiert, daß Einschränkungen für die Plazierung von Objekten durch den Computer verwaltet werden (Einschränkungsbasiertes Paradigma). Wichtig ist, daß die Einschränkungen vor den jeweiligen Handlungen des XKL-Benutzers bestimmt werden, daß

35. Obwohl nicht unmöglich, werden gute Lösungen, die der menschliche Designer findet, nicht explizit belohnt. In vielen Fällen ist die Berechnung einer guten Lösung sehr schwierig. Die Verifikation einer guten Lösung hingegen kann einfach sein (insbesondere in Konstruktionsdomänen, in denen es um Layoutprobleme geht).

XKL also nicht im Nachhinein als Kritiker auftritt. Auf der anderen Seite ist auch HAMVIS selbst als ein interaktives System konzipiert, das Designalternativen und Designbeschränkungen für Visualisierungen verwaltet und dem Oberflächendesigner präsentiert.

HAMVIS stellt für eine Anwendungsklasse ein anwendungsübergreifendes Grundmodell für Domänenobjekte bereit. Durch Definition von Unterkonzepten kann das Systementwicklungsteam domänenspezifische Konzepte einbinden. Weiterhin erstellt das Entwicklungsteam eine Aktionendekomposition zur Spezifikation der Aufgaben des Benutzers und zur Strukturierung der Anwendung. Im vorigen Abschnitt haben wir gesehen, daß die Realisierung der generischen Aktionenkonzepte zur Laufzeit durch UIMS-Dienste übernommen werden kann. Exemplarisch wurden die Dienste von CLIM und deren Anwendung im HAMVIS-System aufgezeigt. Für das Design einer Oberfläche sind jedoch nicht nur die Aspekte der Softwarebausteine relevant. Es müssen Aktions- und Dialogstrukturen explizit gemacht und formal modelliert werden. Hierzu finden sich viele Arbeiten im Forschungsgebiet „Mensch-Computer-Interaktion“. Die nächsten Abschnitte diskutieren die Spannbreite von allgemeinen Aspekten zur Gestaltung von MCI-Systemen über theoretische Arbeiten zur Aufgaben- und Dialogmodellierung bis zu den Grundlagen von implementierten Systemen zur Unterstützung der Aufgabenklasse des in dieser Arbeit als Leitdomäne verwendeten XKL-Systems.

2.2.1 Allgemeine Aspekte der Mensch-Computer-Interaktion

In seinem inspirierenden Buch „The Psychology of Everyday Things“ diskutiert Norman das Design von technischen Gebrauchsgegenständen [234]. Er unterscheidet zwischen drei konzeptuellen (mentalen) Modellen, die im Zusammenhang mit dem Design und mit der Benutzung von Artefakten ein Rolle spielen:

> *„The design model is the designer's conceptual model. The user's model is the mental model developed through interaction with the system. The system image results from the physical structure that has been built (including documentation, instructions, and labels). The designer expects the user's model to be identical to the design model.“([235], S. 189f.)*

Ein Artefakt kann auch ein Computerprogramm sein. Bei der Diskussion von Normans *Theorie der konzeptuellen Modelle* beschränke ich mich auf diese Anwendungssicht.

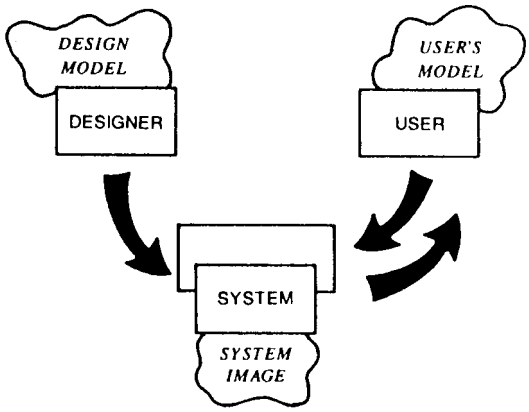


Abbildung 15. Konzeptuelle Modelle nach Norman (aus [235], S. 190).

Nach der Theorie der konzeptuellen Modelle (Abbildung 15) besteht also der erste Schritt beim Entwurf eines Anwendungssystems in der Bildung eines Designmodells, das Entwurfsaspekte der Bedienung, der Benutzer usw. enthält. Anschließend wird das Designmodell mit Hilfe eines Systembilds umgesetzt. Norman betont, daß der Benutzer einer Anwendung i.a. nicht direkt mit dem Anwendungsdesigner kommuniziert, sondern über das Systembild nur mit dem System. Falls das Systembild das Designmodell nicht adäquat reflektiert, ist es für den Benutzer schwierig, sich ein korrektes Modell (mentales Modell) über das System zu machen. Das Ziel des Designs ist es also, das Systembild und das Designmodell aufeinander abzustimmen, so daß ein korrektes Benutzermodell auf einfache Weise abgeleitet werden kann. Dieses allerdings ist i.a. ein sehr schwieriger Prozeß.

Die Verwendung von präsentierten Informationen innerhalb des Interaktionszyklus wird durch Normans Aktionstheorie skizziert (Abbildung 16).

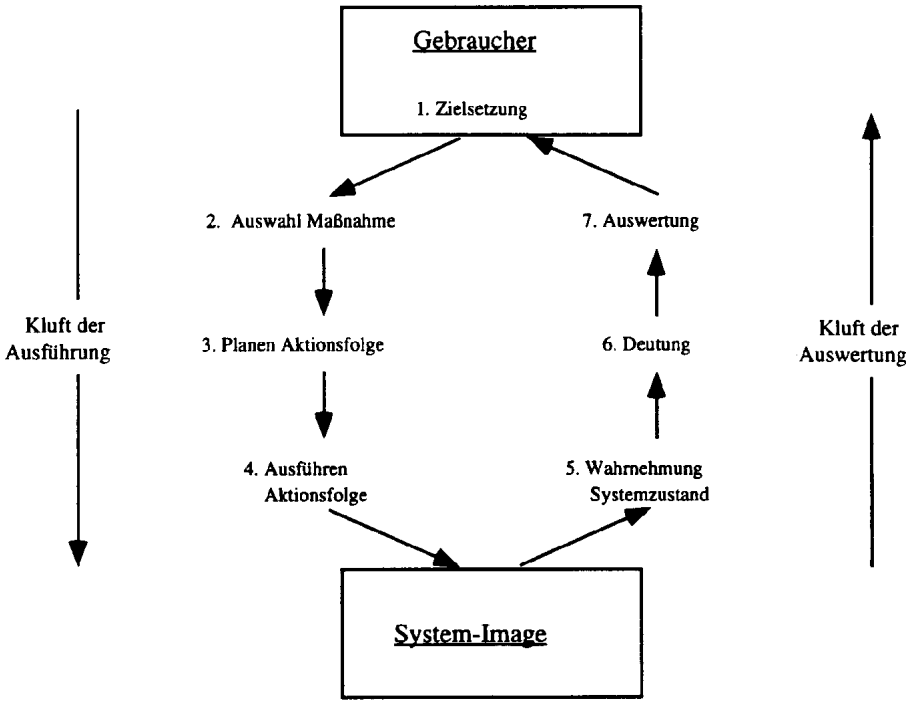


Abbildung 16. Interaktionszyklus mit Zielen und Aktionen (aus [130], Seite 172).

Der Benutzer wählt zur Erreichung seiner Ziele eine *Maßnahme* aus und plant zu deren Umsetzung eine Folge von Aktionen. Nach der Ausführung wird der Systemzustand wahrgenommen und ausgewertet, wodurch sich ggf. neue Ziele ergeben usw. Aus dieser Charakterisierung der Interaktion lassen sich gestalterische Maßnahmen für die Benutzungsoberfläche ableiten (siehe z.B. die Arbeit von Holz [130], Seite 172 ff.).

Das System sollte Aktionen bereitstellen, die mit den Intentionen des Benutzers übereinstimmen. Norman bezeichnet die Differenz zwischen den Benutzerintentionen und den vorgesehenen Aktionen als Ausführungskluft (gulf of execution). Der Zustand eines Systems muß sich kohärent im Systembild widerspiegeln. In Normans Terminologie wird der Aufwand des Benutzers, um das Systembild zu interpretieren und mit den Erwartungen abzugleichen, als Bewertungskluft (gulf of evaluation) bezeichnet. Offensichtlich sollte das Ziel des Entwurfs darin bestehen, beides so klein wie möglich zu

halten.[36] Die Ausführungskluft ist schmal, wenn die Oberfläche die vom Benutzer intendierten Aktionen direkt unterstützt, und die Bewertungskluft wird minimiert, wenn der Zustand eines System leicht vom Benutzer aufgenommen werden kann. Obwohl diese Definitionen einleuchtend sind, darf jedoch das Problem nicht übersehen werden, daß es tatsächlich sehr schwierig ist, formale Modelle zu bestimmen, mit denen es gelingen kann, die „Distanzen“ objektiv zu messen. Die folgenden Abschnitte diskutieren einige der bekannten Ansätze, verschiedene Aspekte der konzeptuellen Modelle zu formalisieren.

Um die „Benutzbarkeit“ (usability) von ASCII-orientierten Texteditoren (insbesondere „`vi`“) zu bewerten, wurde von Card, Moran and Newell das GOMS-Model (Goals, Operators, Methods and Selection rules [38]) vorgeschlagen. In diesem Modell werden Informationsverarbeitungsprozesse auf einfache Basisprozesse reduziert, die mit einer Menge von Operatoren zu höheren Einheiten verknüpft werden, so daß bestimmte Ziele erfüllt werden. Ziele werden auf verschiedenen Granularitätsstufen definiert. Das Modell setzt voraus, daß für die Verarbeitungsprozesse kleine Prozessoren existieren (z.B. für perzeptuelle, motorische und kognitive Aufgaben bzw. Ziele). Die Gesamtheit aller Prozessoren wird als „Model Human Processor“ (MHP) bezeichnet. Die Idee ist, daß Ausführungszeiten als „Kosten“ für die elementaren Teilprozesse des MHP gemessen oder geschätzt werden können, wobei die Ausführungszeiten von Einzelschritten unter Verwendung bestimmter Verrechnungsformeln akkumuliert werden können. Das Modell war konzipiert für die Vorhersage von Ausführungszeiten von fehlerfreien Editiervorgängen von erfahrenen Benutzern.

Die kognitionswissenschaftliche Literatur im Bereich der Mensch-Computer-Interaktion enthält zahlreiche Publikationen mit Kritiken am GOMS-Modell. Arend gibt einen präzisen Überblick über die Hauptargumente ([8], pp. 17-23). Mehrere Erweiterungen von GOMS wurden inzwischen vorgestellt: z.B. Cognitive Complexity Theory (CCT, entwickelt von Kieras und Polson [147] [250]) und andere Ansätze [257]. CCT verwendet ebenfalls eine Beschreibung der Ziele des Benutzers, die auch auf einer GOMS-ähnlichen hierarchischen Darstellung basiert. Operationen werden durch Produktionssysteme und Aktionensequenzen durch generalisierte Übergangsnetze (transition networks) repräsentiert. CCT ist nicht nur für die fehlerfreie Ausführung von Operationen und Aktionensequenzen ausgelegt.

Die Verwaltung von Zielen im Modell sollte dem menschlichen Verhalten angepaßt werden. Ziele können z.B. bestimmte Prioritäten haben. Eine wichtige Organisationsform von Zielen ist die eines Kellers. Dix et al. ([73]) belegen dieses an einem interessanten Beispiel. Benutzer von frühen Geldausgabeautomaten haben häufig ihre Karten vergessen, obwohl Warnhinweise auf dem Bildschirm ausgegeben wurden. Mit Hilfe einer Theorie über die Verwaltung von Zielen läßt sich dieses Verhalten erklären. Das Primärziel ist, Geld aus dem Automaten zu entnehmen. Offensichtlich, werden durch die Benutzer „offene Unterziele“ (die sich auf dem Keller befinden) nicht mehr bearbeitet, wenn das Primärziel erreicht ist. Nach Norman ist die Lösung der Banken, das Geld nicht herauszugeben, bevor nicht die Karte entnommen wurde, ein Beispiel für eine Zwangsfunktion (forcing function, [235], S. 132ff.), die zur expliziten Umsortierung der Ziele auf dem Keller führt.

36. Unter Betrachtung von Sicherheitsaspekten kann es sinnvoll sein, bestimmte Dinge absichtlich schwieriger zu machen. Ein Beispiel ist eine Maschine, die nur durch Drücken von zwei weit auseinanderliegenden Knöpfen gestartet werden kann (Norman in [235], S. 203ff.).

Die Modellierungsformen, die zur Zeit existieren, basieren auf einer kognitiven Architektur (z.B. mit Langzeit- und Kurzzeitgedächtnis), die auf einer Theorie des rationalen Verhaltens aufgebaut ist. Der zentrale Ansatz besteht dabei in einer Dekomposition von komplexen Handlungen und Zielen (oder Aufgaben) in eine hierarchische Struktur. Die vorgeschlagenen Modellierungstechniken (z.B. GOMS und CCT) sind von ihrer Grundausrichtung her Post-hoc-Techniken. Nach meiner Einschätzung ist es zwar möglich, bestimmte Teilaspekte bestehender Artefakte zu modellieren und diese Modelle für Performanzschätzungen über den Umgang von Benutzern mit den Artefakten einzusetzen (siehe auch Dix et al. [73] S. 201f.). Es ist jedoch nicht offensichtlich, in welcher Weise die vorgeschlagenen Repräsentationsformen (GOMS, CCT u.a.) zur *Konstruktion* von Systemen eingesetzt werden können. Die Verwendung von analytischen Modellen im Bereich des Designs von Mensch-Computer-Interaktionsoberflächen wird z.B. von Gugerty [107] untersucht. In seinem Überblick zeigt er auf, wie analytische Modelle von einem Oberflächenentwickler eingesetzt werden können: als Werkzeug zur Aufgabenanalyse, um zu ersten Entwurfsideen zu gelangen, und möglicherweise als Werkzeug, um bei vorläufigen Entwürfen eine erste Übersicht über Schwierigkeiten zu erzielen, die zukünftige Benutzer im Umgang mit dem Endprodukt haben werden.

Auch wenn mit einem Modell Probleme beim Gebrauch von Software-Systemen abgeleitet werden können, so erfolgt dieses lediglich aus Gebrauchersicht, d.h. es ist nicht möglich, direkte Hinweise für den Designer abzuleiten, die helfen, das Gebrauchsproblem zu lösen und das Design zu verbessern. Holz argumentiert in seiner ausführlichen Analyse über den Entwurf von Gebrauchssoftware [130] in ähnlicher Weise. Holz wendet sich auch gegen die Idee, daß es möglich ist, a priori ein Designmodell aufzustellen, das dann „nur noch“ in ein Systembild umgesetzt werden muß. Er hält auch den Vorschlag von Gould und Lewis (und anderen) für ein iteratives Vorgehen bei der Systementwicklung ([101], zitiert nach [130]) für nicht adäquat, da auch hier das Designmodell als fest angenommen wird. Der Grundgedanke des *iterativen Designs* nach Gould und Lewis besteht darin, nach empirischen Untersuchungen des Systembildes unter realistischen Bedingungen dieses solange zu überarbeiten bis es dem Design-Modell entspricht. Holz argumentiert, daß es auch möglich sein muß, das Design-Modell aufgrund der mit dem Systembild gemachten Erfahrungen zu überarbeiten.

Holz bezeichnet das Problem des Entwurfs von Gebrauchssoftware in Übereinstimmung mit der Terminologie von Rittel als „*bösartiges Problem*“. Kennzeichen für ein bösartiges Problem ist, daß Problemspezifikation und -lösung eine Einheit bilden. Es gibt keine direkt abgrenzbaren Teilprobleme, d.h. die Teilprobleme ergeben sich aus der jeweils angestrebten Lösung. Wichtig ist die Wahl der „richtigen“ Problemebene, um zu einer umfassenden Lösung zu gelangen, die nicht nur bestimmte „Symptome“ kuriert. Ein klassischer Systemanalyse-Ansatz als Entwurfsstrategie scheidet nach dieser Überlegung also aus, da hierbei eine Teile-und-Herrsche-Strategie von oben herab angestrebt wird. Auch ein reiner Prototypenansatz löst das Problem nicht, da es aus Kostengründen kaum möglich ist, mehrere Varianten zu erstellen, zwischen denen dann eine Wahl getroffen werden kann. Holz befürwortet also folgerichtig die Aufstellung von Entwurfsmodellen, die jeweils unterschiedliche Teilaspekte des Systemdesigns beleuchten. Es muß möglich sein, mehrere Designmodell/Systembild-Paare zu erzeugen (Varietätserzeugung), die anschließend verglichen und diskutiert werden können (Varietätsreduktion). Eine Diskussion des Systembilds („Reflection-in-Action“) kann dazu führen, daß das Designmodell überarbeitet werden muß.

Das Aufstellen eines Designmodells aufgrund von Gesprächen mit den Benutzern (oder, wie Holz sie bezeichnet, den Gebrauchern), kann kaum im ersten Anlauf gelingen. Holz weist auf die Schwierig-

keiten bei der Wissensakquisition beim Entwurf von KI-Systemen hin. Das „Know-How“ eines (potentiellen) Gebrauchers ist nicht ohne weiteres verbalisierbar (bzw. kommunizierbar oder dem Bewußtsein überhaupt zugängig). Unter Hinweis auf neuere Entwicklungen des „Partizipativen Systemdesigns“ nach Ehn [79], bei denen der Benutzer in seinem Arbeitsumfeld beobachtet wird, folgert Holz, daß ein wesentlicher Teil der Beschreibung des Entwurfs auch im Systembild liegt. Anders ausgedrückt: Das Systembild kann nicht automatisch aus dem Designmodell gewonnen werden, d.h. bei der Erstellung des Systembildes wird weiteres Wissen einbezogen (Wissen aus der partizipativen Analyse der Anwendung oder anwendungsübergreifendes Wissen z.B. zur Oberflächengestaltung).

Holz betont außerdem, daß ein System nicht nur aus Benutzersicht entwickelt werden kann. Es sind im Gesamtentwurf auch die Realisierungskosten für die Software-Komponenten zu bestimmen: Kann eine gewünschte Funktionalität mit vertretbarem Aufwand an Zeit und Personal so implementiert werden, daß bei der Berechnung annehmbare Zeit- und Platzressourcen benötigt werden? Zum Designmodell und Systembild kommt also bei Holz konsequenterweise noch ein *Implementationsmodell* hinzu.

Das ursprüngliche Bild der konzeptuellen Modelle aus Abbildung 15 muß daher erweitert werden. Abbildung 17 zeigt die Teilmodelle und deren wechselseitige Abhängigkeiten nach Holz. Designmodell, Systembild und Implementationsmodell sind als Funktionsmodell zusammengefaßt. Im Entwicklungsprozeß sind verschiedene Personengruppen beteiligt: Designer, Betroffene, Gebraucher, Kunden und Software-Ingenieure.[37]

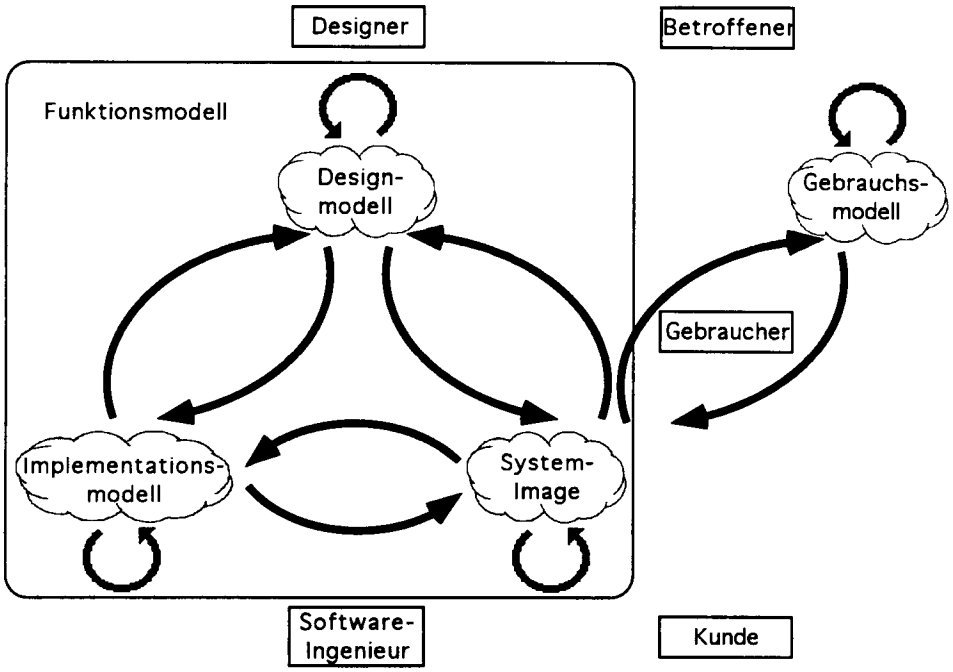


Abbildung 17. Erweiterung der Theorie der konzeptuellen Modelle nach Holz ([130], S. 85).

37. In der Abbildung wird die Stärke der primären Einflußnahme der Persongruppen auf die jeweiligen Modelle durch räumliche Nähe ausgedrückt.

Jedes Teilmodell etabliert gewisse Einschränkungen, d.h. die Entwurfsaufgabe besteht aus der Festlegung und Ausbalancierung von Einschränkungen. Für die Formulierung von Modellen und Ableitung von Einschränkungen werden formale Beschreibungskonzepte benötigt. Wie schon in der Einleitung diskutiert, stehen hierfür verschiedene Metaphern zur Verfügung. Die Metaphern müssen zur Erzielung eines transparenten Entscheidungsprozesses die Kommunikation aller am Designprozeß beteiligten Personengruppen gestatten.

Holz schlägt vor, Konzepte und Beschreibungsformen aus der Arbeits- und Organisationspsychologie zu verwenden (Ulich [324]). Insbesondere das Finden einer adäquaten Problemebene wird durch die Aufteilung zwischen menschlichen und maschinellen Arbeitsaufgaben erleichtert. Die Rolle des Benutzers kann je nach Ausprägung von Softwarekomponenten im Primären Arbeitssystem oder Personifizierten Arbeitssystem als Auftragbearbeiter (exekutive Arbeitsform) oder Auftraggeber (deskriptive Arbeitsform) formuliert werden. Exekutive Arbeitsformen z.B. werden durch eine Gestaltung von Softwarekomponenten als Werkbank mit „Material“ und Werkzeugen unterstützt.[38] Bei einer deskriptiven Arbeitsform führt der Benutzer die Arbeiten nicht direkt aus, sondern beschreibt als Auftraggeber den Auftrag genauer. Während für Holz die Kommunikationsmetapher (siehe die Einleitung zu diesem Kapitel) zur Beschreibung und Entwicklung deskriptiver Arbeitformen schon problematisch erscheint, da nicht jede der am Gesamtentwurf beteiligten Personengruppen mit den hierfür relevanten Begriffen und Konzepten vertraut ist, hält er diese Metapher für die Gestaltung von exekutiven Arbeitsformen sogar für irreführend. Exekutive Arbeitsformen werden durch Interaktionstechniken mit direkter Manipulation realisiert, und Holz argumentiert, daß bei Verwendung von Werkzeugen (Werkbank-orientierter Gestaltungsansatz) die Kommunikationsmetapher von vielen als überzogen empfunden wird: Wer kommuniziert mit seinem Fön?

In diesem Zusammenhang scheinen mir folgende Überlegungen wichtig: Bei vielen speziellen interaktiven Anwendungen verschwimmt der Unterschied zwischen exekutiven und deskriptiven Arbeitsformen. Die in Abbildung 1 betrachtete Beispielanwendung XKL sieht z.B. als mögliche Handlungsalternative für den Benutzer eine Objektlokalisierung vor. Obwohl während der Interaktion zur Berechnungs der Plazierungsbereiche Berechnungsfunktionen aufgerufen werden, wird eine Lokalisierung als exekutiv empfunden. Es hängt aber von der Perspektive ab, ob man die Einfügung eines neuen Kabinenobjektes in das Layout als direkte Ausführung der Aufgabe oder als Folge von Teilaufträgen interpretiert. Letztendlich ist die direkte Manipulation eine Illusion, die bei der Benutzung eines Systems entstehen soll. Aus Konstruktionssicht ergeben sich in jedem Fall bestimmte Teilaufträge, die der Benutzer in vielen Fällen unbewußt an das System richtet. Wenn die Teilaufträge auch zur Benutzungszeit der Anwendung unbewußt bleiben, so ist nicht unbedingt zwingend zu folgern, daß es vorteilhaft ist, wenn dieses auch zur Entwicklungszeit der Fall ist.

Auch bei einem Werkbank-orientierten Entwurfsansatz ist eine (erweiterte) Kommunikationssicht notwendig. Wenn die manipulierten Objekte „nicht im leeren Raum schweben“ sollen, muß der Aktionsraum nach bestimmten Kriterien gestaltet werden. Werkzeuge müssen bereitgestellt werden, so daß bestimmte Arbeiten ausgeführt werden können. Um bestimmte Arbeiten durchzuführen und Entscheidungen zu treffen, benötigt der Benutzer gewisse Informationen. Aus Sicht der Kommunikati-

38. Kritikersysteme kommen dort zum Einsatz, wo dem Benutzer zwar exekutive Arbeitsformen zugedacht werden, er aber aufgrund von mangelnder oder überschätzter Kompetenz die Aufgabe nicht vollständig oder optimal bearbeiten kann.

onsmetapher geht es darum, welche Informationen benötigt werden und wie diese dann verknüpft und kommuniziert werden. Wenn auch zur Benutzungszeit die explizite Kommunikationssicht unbedeutend ist, so folgt daraus nicht, daß sie es auch zur Entwicklungszeit sein muß. Im Gegenteil, auch bei Verwendung von Modellen, die Teilaspekte des Designmodells aus Sicht der Kommunikationstheorie modellieren, kann es das Ziel sein, eine Kommunikationsmetapher zur Benutzungszeit vollständig zu vermeiden.

Im Vergleich mit Normans Aktionstheorie entsprechen die möglichen „Maßnahmen“ aus Normans Interaktionstheorie im HAMVIS-Kontext den Handlungsalternativen (also den zusammengesetzten Handlungen) innerhalb einer Aktivität. Aktionen(folgen) dieses Modells korrespondieren grob mit (Folgen von) elementaren Benutzerhandlungen. Durch die Kombination von Benutzeraktionen mit Berechnungs- und Speicherfunktionen innerhalb einer Aktionenmodellierung kann die Aufgabenverteilung zwischen Mensch und Computer explizit repräsentiert werden. Durch die Wahl der Aktionenzerlegung (mit Aktivitäten zur Strukturierung der Anwendung und zur Bündelung von alternativen zusammengesetzten Handlungen) wird gleichzeitig auch das Systembild beeinflußt. Aspekte des Implementationsmodells werden durch Typangaben für Parameter und Werte von Berechnungs- und Speicherfunktionen einer Aktionenzerlegung modelliert.

Es wird deutlich, daß der Entwurfsprozeß für Visualisierungen, die innerhalb einer Benutzungsoberfläche zur Ausführung von Aktionen verwendet werden sollen, ausgehend von einem Modell für diese Aktionen als Ausbalancierung von Einschränkungen verstanden werden kann. Die von HAMVIS bereitgestellten Repräsentationsformen und Dienste versuchen, die Einschränkungen zu formalisieren und für Ableitungen von notwendigen Inhalten und möglichen Darstellungsformen zu verwenden.

2.2.2 Explizite Modellierung des Systembilds

Im vorigen Abschnitt wurden die konzeptuellen Modelle von Norman (inklusive Erweiterungen) diskutiert: Designmodell, Implementationsmodell und Systembild. Von Mark wurden Konzepte zur Formalisierung der Beziehungen zwischen Designmodell und Systembild vorgestellt: das Systembildmodell (*system image model*) [196]. Die Idee des Systembildmodells war es, explizit zu repräsentieren, was Programmteile aus Sicht der Systembilds leisten, um einem Programmierer eine Unterstützung beim Programmentwurf zu bieten.

In den Arbeiten von Mark zur Modellierung der Realisierung eines Systembilds mit Hilfe von Programmen wurden schon sehr früh KL-ONE-artige Systeme eingesetzt. Ein Systembildmodell soll nicht für jede Anwendung neu aufgebaut werden, sondern basiert auf einem vordefinierten Systembildmodell, das für Mark den Kern des Wissens über ein Systembild enthält ([196], S. 223). Aufbauend auf den Erfahrungen des Consul-Systems ([142], [195]) verwendet Mark die Domäne der Büroautomatisierung. Das vordefinierte Kern-Systembildmodell modelliert Grundstrukturen der in dieser Anwendungsklasse auftretenden Objekte und Aktionen. Abbildung 18 vermittelt einen Eindruck der von Mark definierten Konzepte und Aktionen.

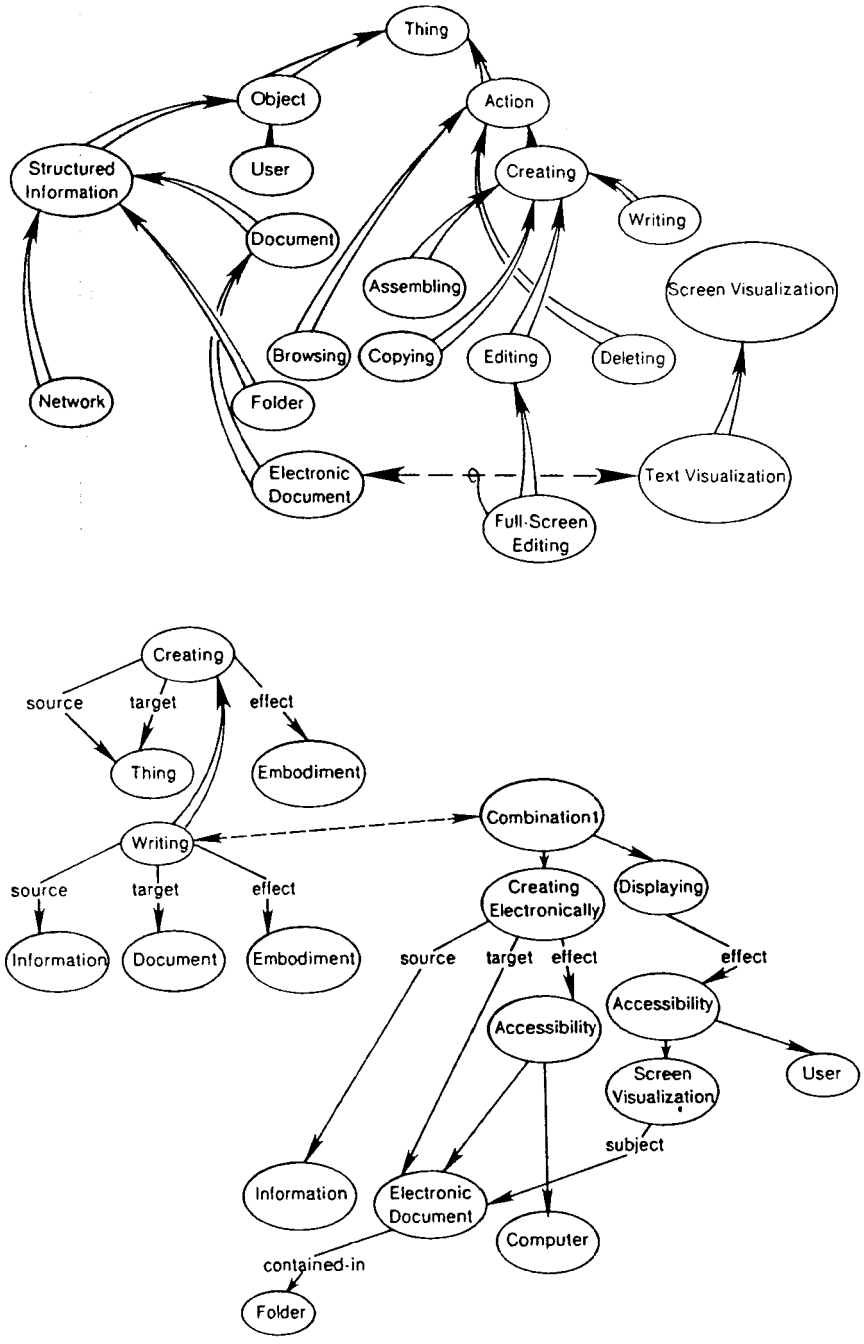


Abbildung 18. Teile des vordefinierten Systembildmodells von Mark [196]. Charakteristische Beziehungen sind mit durchgezogenen Pfeilen dargestellt, Inferenzregeln werden mit gestrichelten Verbindungen angedeutet. Elektronische Dokumente können z.B. durch eine Textdarstellung beschrieben werden, wenn ein Bildschirmeditor zur Verfügung gestellt wird. Unter bestimmten Bedingungen kann „Schreiben“ durch „Erzeugen“ und „Darstellen“ realisiert werden (bzw. aus diesen Teilen zusammengesetzt werden).

Ein Modell für eine spezielle Anwendung muß mit dem Kern-Systembildmodell verbunden werden. Das System von Mark sieht vor, für die Erstellung eines anwendungsspezifischen, erweiterten Systembildmodells interaktive Darstellungen (browser) und formularähnliche Eingabetechniken zu

verwenden. Domänenspezifische Anwendungsfunktionen werden unter Zuhilfenahme der Konzepte des erweiterten Systembilds definiert (Typen der Formalparameter und Werte). Mark et al. argumentieren, daß die Vorteile von expliziten Modellierungstechniken besonders deutlich werden, wenn eine Applikation von mehreren Personen(gruppen) erstellt wird: „Stating design models explicitly in terms of prebuilt abstractions [...] allows programmers to sort the features of their designs into conceptual categories. This greatly reduces the burden of consistent delivery [of the design model] in the implementation." ([196], p. 235).

Marks Modell stellt einen ersten Versuch dar, auf Domänenobjekte angewendete Aktionen formal zu modellieren. Allerdings stellt das Kern-Systembildmodell keine Funktionen zur Realisierung der Aktionen (und Objekte) bereit. Der Modellierungsmehraufwand der von Mark vorgestellten Methode sollte nicht unterschätzt werden. Im Gegensatz zu dem Ansatz von Mark, bei dem formale Modelle nur zur Erreichung von Konsistenz zwischen Designmodell und Systembild dienen (wie immer Konsistenz genauer definiert sein mag), muß es das Ziel der Entwicklung eines Kern- oder Grundmodells sein, Dienste bereitzustellen, die tatsächlich zur Generierung von Oberflächen bzw. Visualisierungen verwendet werden können. Mit anderen Worten: Im HAMVIS-Kontext muß der Modellierungsmehraufwand mehr als nur Konsistenzüberprüfungen bewirken. In Kapitel 3 dieser Arbeit wird gezeigt, wie dieses Ziel erreicht werden kann.

Im vorigen Abschnitt über die Konzepte von UIMS wurde die Bedeutung des Grundmodells für eine Anwendungsklasse für die Bereitstellung von Interaktionstechniken auf UIMS-Ebene herausgestellt. Die Arbeiten von Mark machen deutlich, daß die Definition eines Grundmodells für eine Anwendungsklasse auch notwendig ist für die Spezifikation der Konzepte von Aktionen des Benutzers. Aktionen stehen in engem Zusammenhang mit dem Begriff der Aufgabe (task). Durch Ausführung einer Aktion versucht der Benutzer, eine Aufgabe zu erledigen bzw. die mit der Aufgabe verbundenen Ziele zu erfüllen. Obwohl eine Aufgabe durch mehrere Aktionen bearbeitet werden kann und eine Aktion eventuell mehrere Aufgaben lösen kann, werden die Begriffe meist synonym (bzw. dual zueinander) verwendet. Im folgenden werden einige Arbeiten vorgestellt, die Aspekte von Aktionen aus Sicht der damit assoziierten Aufgabe formalisieren.

2.2.3 Aktionen- und Aufgabenmodellierung

Zur Analyse und Beschreibung von Benutzertätigkeiten und der jeweiligen Arbeitsumgebungen werden in Aufgabenmodellen hierarchische Dekompositionstechniken eingesetzt. Im Gegensatz zu GOMS liegt hier der Schwerpunkt auf dem beobachtbaren Verhalten und nicht auf den internen „mentalen Zuständen" (oder Zielen) der Benutzer. In fast jeder Konferenz aus dem Bereich Mensch-Computer-Interaktion werden neue Modelle zur Aufgabenanalyse und -modellierung diskutiert. In frühen Ansätzen bestand der Zweck der Modelle besteht darin, die Implementation einer Applikation vorzubereiten. Die konkrete Implementation einer Oberfläche erfolgte dann vielfach in einer Standard-Programmiersprache (wenn dieses überhaupt diskutiert wurde). Erst in jüngster Zeit, wird untersucht, wie Aufgabenbeschreibungen direkt in die Generierung von Programmcode für ein Zielsystem einmünden können („Executable Task Analysis").

Aufgabenanalysemodelle beschreiben, in welcher Reihenfolge und unter welchen Bedingungen Unteraufgaben bearbeitet werden. Neben der Zerlegung ist auch die Parametrierung von Aufgaben sowie die Vererbung von Parametern innerhalb der Zerlegungshierarchie ein wichtiges Thema. Als (De-)Kompositionsoperatoren wurden vielfach Operatoren wie `AND`, `OR`, `XOR` verwendet: z.B. Task

Analysis for Knowledge Description (TAKD [139]) oder Task Descriptive Hierarchy (TDH, zitiert nach [73] sowie UAN [118]). Die Schachtelung von Aufgaben ist bei diesem Ansätzen allerdings frei wählbar. Ich sehe bei einer beliebig geschachtelten Zerlegung allerdings das Problem, daß die Beziehung von Aufgaben zu den Oberflächenelementen verloren geht. Man erhält einen willkürlichen Designraum. Auch andere Autoren argumentieren dafür, daß Aufgabenzerlegungen so gestaltet werden müssen, daß sich die Modelle methodisch in Oberflächenelemente überführen lassen [57].

In der Arbeit von Philips et al. wurde die Zerlegung von Aufgaben bzw. Aktionen eingeschränkt [248]. Philips et al. stellen Konzepte für eine hierarchische Aktionenmodellierung für die Anwendungsklasse der Systemüberwachung vor. Bei einer hierarchischen Dekomposition von Aufgaben können nach dem Modell von Philips verschiedene Granularitätsstufen unterschieden werden: „Aktivitäten“ (z.B. Ausführung von Systemüberwachungsaufgaben, Auflösen von Kapazitätsproblemen für Zwischenpuffer). „Unter-Aktivitäten“ (z.B. Überprüfung und Bewertung von Verkehrsflüssen, Auflösen von Übertragungsrückstaus), „Aufgaben“ (tasks, z.B.: Erkennen eines Nachrichtenrückstaus, Umdirigieren von Nachrichten auf andere Übertragungswege) und „Aufgabeelemente“ (task elements, z.B.: BETRACHTE Nachrichteneingang, SCHÄTZE Nachrichtenbearbeitungszeit):

> *„An ‘activity’ describes a high-level response to events. Activities are user actions that relate directly to the overall goals of the system [...]. The next level of analysis is the ‘sub-activity’ level. Sub-activities are the highest-level constituents of activities, and are directly associated with one or more events. ‘Tasks’ define what must be accomplished to complete a sub-activity. Finally, ‘task elements’ are the procedural components of tasks, the steps and actions by which a task is accomplished.“ ([248], S. 842)*

Ähnlich wie HAMVIS, verwendet Philips vier Ebenen der Aufgaben- bzw. Aktionenmodellierung. Die von Philips als „activity“ bezeichnete Ebene entspricht der Applikationsebene in HAMVIS. Die „sub-activities“ sind vergleichbar mit den von mir als Aktivitäten bezeichneten Arbeitseinheiten. „Tasks“ finden in HAMVIS ihre Entsprechung in den zusammengesetzten Handlungen. Nicht weiter zerlegbare Elemente (task elements) heißen in HAMVIS-Terminologie elementare Benutzerhandlungen.

In [248] stellen Philips et al. Operatoren bereit, um Einheiten auf verschiedenen Ebenen miteinander zu verknüpfen: `SOME` (gleichzeitige Verfolgung mehrerer Pfade), `REPEAT`, `IF`, `ONE` (Verfolgung genau einer Pfadalternative). In ihrer Arbeit verwenden Philips et al. Aufgabenzerlegungsmodelle, um die Gestaltung einer Oberfläche herzuleiten. Sie verwenden Elemente der Aufgabensprache zur Unterstützung der Entscheidungsfindung bzgl. der Wahl von Designalternativen (auf den verschiedenen Ebenen: konzeptuelle, semantische, syntaktische und lexikalische Ebene, siehe das historische Papier von Foley et al. [91]). Die Beispielanwendung, die von ihnen diskutiert wurde, ließ sich durch eine formularbasierte Oberfläche realisieren. Oberflächenelemente müssen so gestaltet werden, daß schließlich die elementaren Aufgaben (elementary tasks) durchgeführt werden können. Die höheren Aufgabenebenen liefern hierfür Einschränkungen und strukturieren die Benutzungsschnittstelle in sinnvolle Interaktionseinheiten.

Die von Philips et al. vorgeschlagenen analytischen Modelle sind darauf ausgerichtet, einen menschlichen Designer zu unterstützen. Es gibt keine formale Basis, auf der automatische Generierungs- und Kompositionsmethoden für Visualisierungen in Benutzungsschnittstellen aufgebaut werden können. Beziehungen zwischen Berechnungs- und Speicherfunktionen und menschlichen Aufgaben werden

nicht explizit modelliert. Informationen für Entscheidungsprozesse werden durch natürliche Sprache auf der Aufgabenebene (task level) beschrieben. Allerdings werden durch die Aktionen auf den unteren Ebenen (elementary tasks) keinerlei Hinweise für eine mögliche Realisierung durch Oberflächenelemente gegeben.

Nach den Erfahrungen von Hartson und Mayo [120] sind jedoch die unteren Ebenen der Aufgabenmodellierung, also der Übergang zu den Oberflächenelementen, gerade der schwierigste Teil der Modellierung. In einer Erweiterung zur „User Action Notation“ (UAN) betrachten Hartson und Mayo die Einbeziehung von „semantischen Informationen“ in höhere Ebenen der Aufgabenmodellierung, um die Modellierung auf der unteren Ebene (genannt „articulatory level“) steuern zu können. Im KI-Sinne sind mit „semantischen Informationen“ einfach nur Domänenobjekte gemeint. Hartson et al. verwenden den Begriff zur Abgrenzung von Domänenobjekten und graphischen Objekten. UAN selbst definiert eine Sprache, mit der beschrieben werden kann, was mit Graphikobjekten passiert, wenn z.B. die Maus bewegt wird, wenn ein Mausknopf gedrückt wird usw. Die Erweiterung sieht nun auch Beschreibungsformen für die jeweils entsprechenden Zustandsveränderungen von Domänenobjekten vor. Zur Strukturierung der Aufgabenmodellierung und zur Definition von wiederverwendbaren Bausteinen wird die Aktionenzerlegung in Schichten eingeteilt (semantische Ebene, syntaktische Ebene, artikulatorische Ebene). Domänenobjekte werden durch die Aufgabenzerlegung bis auf die artikulatorische Ebene „durchgereicht“. Auf der syntaktischen Ebene wird z.B. festgelegt, ob ein Objekt zuerst markiert und dann ein Kommando aufgerufen wird (z.B. über ein Menü) oder ob zuerst ein Kommando gegeben wird und das anschließend selektierte Objekt dann einen Parameter darstellt. Sie diskutieren den Ansatz an dem Beispiel des Öffnens einer Datei. Ich möchte zur Illustration das Beispiel hier kurz widergeben (Abbildung 19).

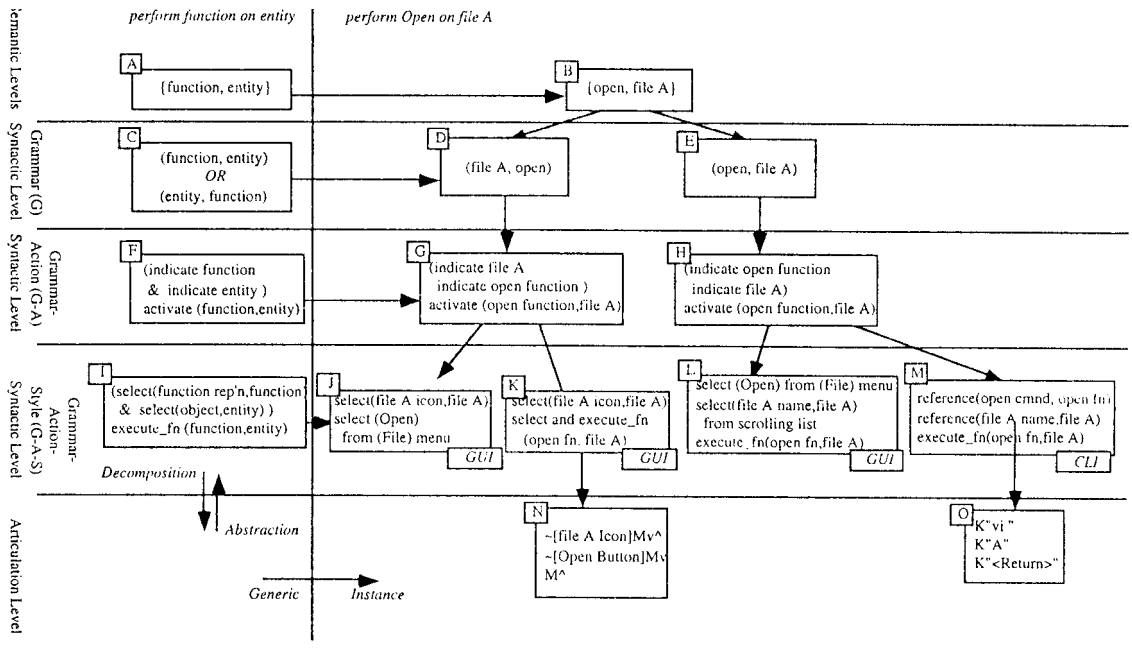


Abbildung 19. UAN-Aufgabenmodell für das Öffnen einer Datei (aus [120]).

Eine Aufgabe wie z.B. das Öffnen einer Datei wird zerlegt in zwei mögliche Alternativen. Jede der Alternativen wird durch Aufgaben auf der Ebene der Sprech- bzw. Kommunikationsakte weiter verfeinert.[39] Auf der nächsten Zerlegungsebene erfolgt dann die Zuordnung zu graphischen Objekten

(mit kommunikativen Akten auf niedriger Ebene), wobei anschließend (bei Bedarf) definiert wird, wie die Aufgabe auf der artikulatorischen Ebene durchgeführt werden kann.

Bei näherer Betrachtung wird deutlich, daß man statt Aufgabe auch Aktion sagen könnte, ohne daß sich die Modellierungsformen wesentlich ändern würden. Auf der Ebene drei und vier (von oben) wird die Umsetzung einer Aktion in Form von kommunikativen Akten direkt vorgegeben.

Es wird deutlich, daß UAN für Standard-Interaktionswerkzeuge ausgerichtet ist (Menüauswahl, Kommandoeingabe). Betrachtet man den Ansatz aus der CLIM-Perspektive, so erkennt man, daß die erste Ebene des Aufgabenmodells aus Abbildung 19 mit der Definition eines Kommandos vergleichbar ist. Als Parameter wird ein Objekt von Typ „File“ angegeben. Ebene fünf korrespondiert mit den jeweiligen Interaktionstechniken um ein Kommando auszulösen (z.B. Menüauswahl oder textuelle Kommandoeingabe). Die Ebenen zwei bis vier werden automatisch (und konsistent!) durch UIMS gehandhabt und brauchen z.B. bei CLIM nicht modelliert zu werden. Die UAN ist also eventuell geeignet, neue UIMS-Bausteine intern zu beschreiben, bietet jedoch für die Beschreibung von Benutzeraktionen ein zu niedriges Abstraktionsniveau. Auf der Gestenebene wird zwar sehr detailliert modelliert, auf der oberen, semantischen Ebene (semantic level, siehe Abbildung 19) werden jedoch kaum ausreichende Beschreibungsformen bereitgestellt. Außerdem werden in den entsprechenden Veröffentlichungen Aktionen bzw. Aufgaben mit graphischen Anwendungsobjekten in keiner Weise erfaßt, sondern nur Aktionen besprochen, die mit Standard-Interaktionsbausteinen realisiert werden.

Interessant ist aber der Ansatz, generische Schemata herauszuarbeiten, wobei jedoch m.E. unterschiedliche Kategorien der Abstraktionsbildung vermengt werden:

- Parametrierung (Ebene 1)
- Zerlegung (Ebene 2)
- Klassifikation (Ebene 3)
- Spezialisierung von Teilkomponenten (Ebene 4)

Hartson und Mayo beziehen sich bei ihrer Klassifikation von Aufgaben (bzw. Kommunikationsakten) auf Arbeiten von Lenorovitz et al. (zitiert nach [120]). Das verwendete Klassifikationsschema möchte ich hier in Abbildung 20 skizzieren.

39. Interessanterweise vermeiden die Autoren die Sprechakt-Nomenklatur.

CREATE	ASSOCIATE	NAME	Give title to or attach label to for purposes of identification/reference.
		GROUP	Link together or associate for purposes of identification.
	INTRODUCE	INSERT	Make space for and place an entity at a selected location within the bounds of another such that the latter wholly encompasses the former, and the former becomes an integral component of the latter.
	ASSEMBLE	AGGREGATE	Combine two or more components so as to form a new composite entity.
		OVERLAY	Superimpose one entity on top of another so as to affect a composite appearance while still retaining the separability of each component layer.
	REPLICATE	COPY	Reproduce one or more duplicated of an entity (no links to *master*).
		INSTANCE	Reproduce an original (*master*) entity in such a way as to retain a definitional link to the master—i.e., such that any subsequent changes or modifications made to the master will automatically be reflected in each and every *instance* created therefrom.
INDICATE	SELECT *(POS/OBJ)*		Opt for or choose an entity (e.g., a position or an object) by *pointing* to it.
	REFERENCE		Opt for or choose an entity by invoking its name.
ELIMINATE	REMOVE	CUT	Remove a designated portion of an entity and place it in a special purpose buffer (residual components of the original entity usually close in around *hole* left by *cut-out* portion).
		DELETE	Remove and (irrevocably) destroy a designated portion of an entity.
	STOP	SUSPEND	Stop a process and temporarily hold in abeyance for future restoration.
		TERMINATE	Conclude a process such that it cannot be restarted from the point of interruption, only by complete re-initiation.
	DISASSOCIATE	RENAME	Change an entity's title or label, without changing the entity itself.
		UN-GROUP	Eliminate the common bond or reference linkage of a group of entities.
	DISASSEMBLE	SEGREGATE	Partition and separate an entity into one or more component parts such that the structure and identity of the original is lost.
		FILTER	Selectively eliminate one or more layers of an overlaid composite.
		SUPPRESS	Conceal or keep back certain aspects or products of a process without affecting the process itself (i.e., affects appearance only).
		SET-ASIDE	Remove entire contents of current (active) work area and store in a readily accessible buffer (for future recall).
MANIPULATE	TRANSFORM		Manipulate or change one or more of an entity's attributes (e.g., color, line type, character font, size, orientation) without changing the essential content of the entity itself.
ACTIVATE	EXECUTE __ *FN.*		Initiate or activate any of a set of predefined utility or special purpose functions (e.g., sort, merge, calculate, update, extract, search, replace).

Abbildung 20. Klassifikation von Benutzeraufgaben nach Lenorovitz (zitiert nach [120]).

Die Konzepthierarchie wird zur Spezialisierung von Teilkomponenten verwendet (Übergang von Ebene drei auf Ebene vier). Die Aufgabe `indicate` wird in einer Alternative zu `select` spezialisiert, in einer anderer Alternative zu `reference`.

Die Klassifikation von Benutzeraktionen ist ein interessanter Ansatzpunkt. Das Ziel von Lenorovitz war, generische Aktionen zu definieren. Für die Inhaltsbestimmung im Kontext von HAMVIS müssen Aktionenkonzepte jedoch weniger abstrakt sein.

Hartson, Brandenburg and Hix [119] betrachten frühe Aktivitäten beim Oberflächendesign (z.B. Aufgabenanalyse) als Modellierung des *Verhaltens* der Benutzer. Verhaltensorientiertes Design (behavioral design) betrifft die physikalischen und kognitiven Benutzeraktionen und die zur Unterstützung der Handlungen notwendigen Rückkopplungen der Oberfläche. Für die Entwurfsentscheidungen auf

niedrigerer Ebene (Programmierung von Ereignisbehandlungsprozeduren) verwenden sie den Term konstruktionelles Design (constructional design) und betonen, daß von einem Oberflächenentwickler zuerst die verhaltensorientierte Ebene betrachtet wird (bzw. werden sollte). Anschließend müssen Repräsentationen der Verhaltensebene auf die konstruktionelle Ebene „übersetzt“ werden.

Um die Erstellung von Oberflächen auf diesen Ebenen zu unterstützen, verwenden sie die UAN, deren primäre Abstraktion eine Benutzeraufgabe ist. Sie betrachten z.B. Benutzeraufgaben wie die Bewegung des Mauszeigers auf ein Dateipiktogramm, Drücken einer Maustaste, Bewegen des Mauszeigers auf den Mülleimer, Loslassen der Maustaste. Für spezielle Aufgabensequenzen dieser Art wird eine formale Notation vorgestellt. Bei Betrachtung der Konzepte von CLIM entspricht diese Abstraktionsebene der Ebene der Präsentation-nach-Kommando-Abbildungsfunktionen (hier eine spezielle Ziehen-und-Fallenlassen-Abbildung).

Die Ausführungen im vorigen Abschnitt belegen, daß sich die Modellierung von interaktiven Systemen mit modernen UIMS wie z.B. CLIM sehr vereinfacht hat. Ansätze aus der Mensch-Computer-Interaktion greifen vielfach Themen auf, die auf UIMS-Ebene schon gelöst sind (z.B. „konkrete Interaktionsobjekte“ und „abstrakte Interaktionsobjekte“, siehe die Ausführungen zur Portabilität im vorigen Abschnitt über UIMS). Um nun Portabilität zu gewährleisten, verwendet CLIM ein besseres Modell als die UAN, indem ein Kommando (z.B. delete-object) von der Interaktionstechnik, die das Kommando auslöst, getrennt wird. Die Präsentation-nach-Kommando-Abbildung wird durch eine Geste initiiert, die wiederum auf physikalische Geräte abgebildet werden kann (z.B. auf den linken Mausknopf). Eine Trennung dieser verschiedenen Ebenen wird von vielen UIMS nicht explizit vorgenommen. Auch die in diesem Kapitel vorgestellten Techniken zur Dialogspezifikation vernachlässigen zum Teil in diesem Punkt eine sorgfältige Modellierung. Die von Hartson et al. gemachte Trennung von verhaltensorientierter Beschreibung und konstruktionellem Design ist an sich sinnvoll. Das von ihnen verwendete Beispiel ist m.E. allerdings wenig überzeugend. Die Betrachtung von fortgeschrittenen UIMS-Konzepten zeigt allerdings deutlich, daß die Verhaltensebene von Hartson et al., aus Sicht eines modernen UIMS als Ebene der technischen Umsetzung angesehen werden kann (konstruktionelle Ebene). Einige der Aspekte, die durch die hier vorgestellten Dialogmodelle erfaßt werden, sind heutzutage durch bessere UIMS-Abstraktionen abgedeckt (insbesondere Ereignisbehandlung usw.).

Weiterhin werden kaum Anwendungen betrachtet, in denen geometrische Daten als Grundlage für domänenspezifische Graphen verwendet werden. Die Ansätze sind auf die Verwendung von Standard-Interaktionselementen ausgerichtet (z.B. [252]). Ansätze wie z.B. MASTERMIND verwenden abstrakte Graphiken mit Piktogrammen.

In den betrachteten Aktionen wird angegeben, welche Objekte eine Rolle bei der Interaktion spielen. Es stellt sich die Frage, ob man dieses eventuell sogar herleiten kann, wenn man eine noch „tiefere“ Modellierungsform anstrebt und z.B. „Informationsverarbeitungsprozesse“, die mit den Aufgaben verbunden sind, in den Vordergrund der Modellierung rückt. Ansätze zur Entwicklung von Oberflächen aus der Perspektive der „Informationsverarbeitungsziele“ wurden im GRADIENT-Projekt untersucht, wobei insbesondere die Gestaltung von Oberflächen zur Unterstützung von Entscheidungsprozessen des Benutzers betrachtet wurde. Inspiriert durch die Skandinavische Schule der Mensch-Computer-Interaktion („Cognitive Engineering and Human-Computer Interaction“ [254] [138]) diskutiert Sundström den GRADIENT-Ansatz zur Unterstützung der Oberflächenkonstruktion. Als Anwendungsklasse betrachtet GRADIENT ein Prozeßüberwachungsszenario für ein Hochdruck-Vorheizsystem eines Kraftwerks. Das Ziel der Arbeit wird von Sundström wie folgt beschrieben:

„[Reduce] the extent to which designers have to rely on ‘common sense’ and increase the extent to which design is based on explicit knowledge about different aspects of the target domain for which the system is built“ ([300], p. 568).

Eine Schnittstelle für ein Prozeßüberwachungssystem kann z.B. Flußdiagramme mit Piktogrammen für den Zugriff auf Prozeßvariablen enthalten. Für bestimmte Entscheidungsaufgaben müssen die jeweils benötigten Prozeßvariablen zusammengefaßt und (adäquat) dargestellt werden. Sundström nennt ihren hierzu vorgestellten Ansatz „Functional Information and Knowledge Acquisition Modelling“ (genannt FIKA-Modellierung). Die Idee hierbei ist, während der Entwicklungszeit der Oberfläche ein explizites Informationssuchmodell (information search model) für den Operateur der Anlage aufzubauen, das dazu verwendet wird, die relevanten Prozeßvariablen (Zustände und Zustandsänderungstendenzen) zu bestimmen und durch entsprechende graphische Elemente an der Oberfläche zu präsentieren.

Bei der FIKA-Modellierung besteht eine diagnostische Aktivität (z.B. eine Prozeßüberwachung) aus den folgenden Schritten, die auf entsprechende Informationsverarbeitungsziele des Operateurs abgebildet werden können (vgl. die Interaktionstheorie nach Norman):

- Einschätzung der Situation
- Auswahl von Aktionen aus einer Menge von möglichen Aktionen
- Beobachtung der Wirkung der gewählten Aktion(en).

Die Benutzungsoberfläche muß diese Informationsverarbeitungsziele, die direkt mit sogenannten „Informationssuchzielen“ zur Einschätzung der aktuellen Prozeßsituation gekoppelt sind, angemessen unterstützen. Informationsverarbeitungsziele werden durch eine Sequenz von „Informationsverarbeitungsschritten“ erfüllt. Informationsverarbeitungsschritte werden als Knoten im Informationssuchmodell repräsentiert. Dabei wird zwischen verschiedenen Knotentypen unterschieden: Suchknoten, Überprüfungsknoten, Zustandsknoten und Überwachungsknoten. Um die Grundstruktur eines Informationssuchmodells aus GRADIENT zu demonstrieren, wird in Abbildung 21 ein Beispiel betrachtet.

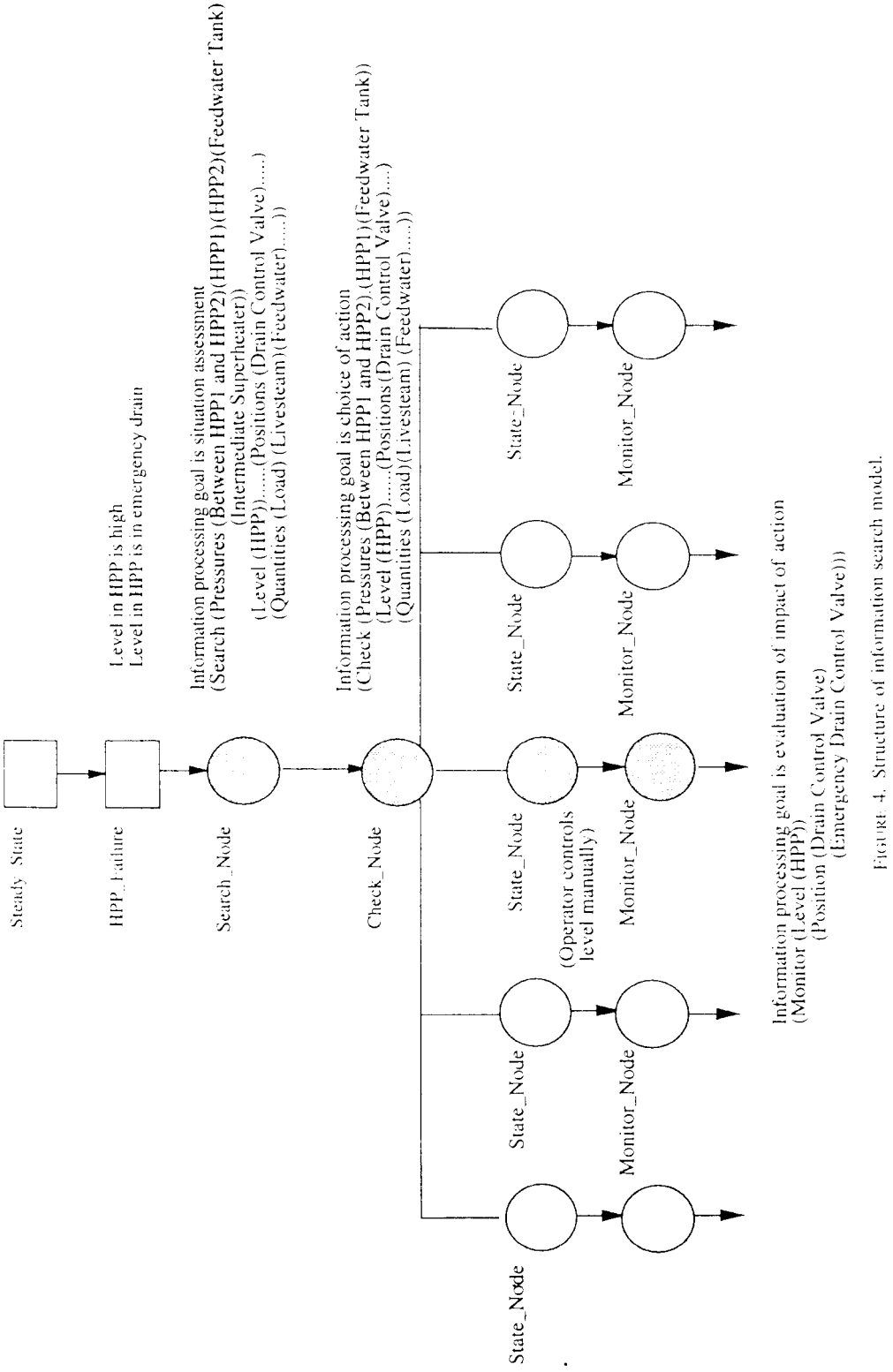


FIGURE 4. Structure of information search model.

Abbildung 21. Beispiel für ein GRADIENT-Informationssuchmodell

Suchknoten repräsentieren Informationsverarbeitungsziele, die während der Situationseinschätzungsphase relevant sind. *Überprüfungsknoten* modellieren die Auswahl einer Aktion, die wiederum durch Zustandsknoten modelliert werden. Letztendlich dienen Überwachungsknoten zur Repräsentation von Informationen, die für die Bewertung des Effektes einer Aktion benötigt werden. Suchknoten werden immer gefolgt von Überprüfungsknoten, die ein oder mehrere Zustandsknoten als Nachfolger haben können. Zustandsknoten werden jeweils von Überwachungsknoten gefolgt. Ein Pfad in diesem Baum, der als Entscheidungssatz (decision set) bezeichnet wird, entspricht jeweils einem Handlungsstrang des Operateurs.

Ein Entscheidungssatz wird aus einer Menge von Regeln und einem Frame-basierten Weltmodell (zur Repräsentation von Messungen für Drücke oder Mengen, Modellierung von Ventilpositionen usw.) automatisch bestimmt.[40] Für jeden Knotentyp, der im Entscheidungssatz vorkommen kann, gibt es einen bestimmten Regeltyp (siehe Abbildung 22). Die Modellierung der Anwendung erfolgt demnach mit Frames und Regeln., die durch den Oberflächenentwickler für jede Anwendung neu definiert werden.

```
Search rule class

IF Operator monitors the process &
  Message "Level in High-Pressure Preheater is in Emergency Drain" &
  Message "Level in High-Pressure Preheater is High"
THEN Search for all overview information
     Position of Drain Controller
     Position of Emergency Drain
     Controller Pressure in Feedwater Tank
     Pressure in High-Pressure
     Preheater Pressure between High-Pressure Preheaters
     Level in High-Pressure Preheater

Check rule class

IF Operator monitors the process &
   Current Load is observed
THEN Check
     Position of Drain Controller
     Position of Emergency Drain Controller
     Pressure Difference between Feedwater
     Tank and High-Pressure Preheater
     Tendency of Level in High-Pressure Preheater

Monitor rule class

IF Operator controls level in High-Pressure Preheater manually
THEN Monitor
```

Abbildung 22. Beispiele für Regeltypen (information search rules), die verwendet werden, um einen Entscheidungssatz für eine Anwendung zu bestimmen (nach [300], S. 575).

In einem Entscheidungsbaum werden Knoten mit Informationssuchobjekten (information search objects) versehen, die mehr oder weniger direkt die Informationen kodieren, die in den Regeln in

40. Das GRADIENT-System verwendet KEE als Repräsentationssprache für das Domänenmodell.

Form von Objekt-Attribut-Paaren vorgegeben wurden. Die Idee von GRADIENT ist also, Domänenobjekte, Aktionen und Informationsverarbeitungsziele (verschiedenen Typs) mit Frames und Regeln zu modellieren. Die relevanten Teile des Modells werden automatisch in einen Entscheidungsbaum transformiert. Knoten im Entscheidungsbaum (Informationsverarbeitungsschritte) werden mit Objekt-Attribut-Paaren versehen, die relevante Informationen für den Informationsverarbeitungsschritt darstellen. Die Entscheidungsbäume werden zur manuellen Konstruktion von Oberflächen verwendet. Sie stellen eine Checkliste dar, gemäß derer die erstellte oder konzipierte Oberfläche überprüft werden kann. Die Überprüfung erfolgt durch den Oberflächendesigner.

Sundström berichtet allerdings nicht über eine Evaluierung des GRADIENT-Ansatzes (z.B. eine Studie, in der eine Oberfläche für eine bestimmte Anwendung im Bereich der Prozeßüberwachungen einmal mit und einmal ohne den GRADIENT-Ansatz von verschiedenen Gruppen durchgeführt wurde). Weiterhin sind in Anwendungen wie XKL, in denen Layoutprobleme bearbeitet werden, räumliche Konstellationen zwischen räumlichen Objekten relevant. Konstellationen können erst zur Benutzungszeit der Anwendung berechnet werden. Die Definition eines Konstellationenmodells muß allerdings schon zur Entwicklungszeit (möglichst in deklarativer Form) erfolgen. Regeln, wie sie bei GRADIENT eingesetzt werden, sind hierfür kaum ein genügend mächtiges Repräsentationsmittel. Auch in Prozeßüberwachungssystemen spielen Beziehungen zwischen mehreren Prozeßvariablen eine große Rolle, so daß auch hier Modelle zur Vorgangserkennung und -überwachung notwendig sind. Für die Darstellung bzw. Definition dieser Modelle bietet GRADIENT keine Unterstützung (vgl. z.B. Weir und Alty [336]).

Der GRADIENT-Ansatz ist ein Versuch, explizit zu repräsentieren, welche Informationen zur Durchführung bestimmter Aufgaben benötigt werden. Zur Entwicklungszeit werden diese Informationen dem Designer in einer interaktiven Oberfläche präsentiert, um sicherzustellen, daß in der vom ihm konstruierten Oberfläche „nichts vergessen wird“. Bei diesem Anspruch ist allerdings zu hinterfragen, wie überprüft werden kann, ob die zuvor aufzustellenden Regelmengen vollständig sind! Es ist nicht klar, ob die aufgestellten Regeln z.B. auf Konsistenz geprüft werden können. Wenn mit den Regeln keine weiteren Aufgaben erledigt werden, könnte auch gleich ein Entscheidungsbaum mit entsprechenden „Beschriftungen“ erstellt werden (siehe Abbildung 21).

Es wird deutlich, daß Modelle dieser Art (second-principles models), also Modelle, die nicht wie Basisprinzipien-Modelle (engl. first-principles models) simulativ den Perzeptionsprozeß zu modellieren versuchen, nicht sehr viel zur Erhöhung der Modellierungskraft beitragen können. Wie wir in folgenden Abschnitten dieser Arbeit sehen werden, stecken aber Basisprinzipien-Modelle bei dem momentanen Stand der Kunst in den Disziplinen Mensch-Computer-Interaktion, KI, kognitiver Psychologie usw. erst in den Kinderschuhen. Für die Arbeiten zu HAMVIS habe ich mich daher entschieden, die für die Aktionen direkt relevanten Objekte auch direkt bei der Aktionenmodellierung anzugeben, also nicht erst den Umweg über Informationsverarbeitungsziele zu gehen, da das gleiche Ziel erreicht werden kann.

Im Bereich der wissensbasierten Systementwicklung wurden weitere Vorschläge zur Modellierung von Aufgaben gemacht, wobei insbesondere die Beziehungen zwischen Benutzeraufgaben und algorithmischen Programmkomponenten aus einer wissensbasierten Perspektive modelliert wurden. Eine der vorgestellten Methoden ist KADS [276].[41] Basierend auf den vier Ebenen des KADS-Vorgehens bei der Wissensrepräsentation (domain level, inference level, task level, strategic level) haben die KADS-Entwickler auch Aspekte der Benutzer-System-Kooperation betrachtet (Breuker und de Greef

[34]). In KADS wird die Inferenzschicht zur Modellierung von elementaren Problemlösungsprozessen verwendet. Weiterhin werden die Rollen erfaßt, die bestimmte Domänenobjekte in diesen Prozessen spielen. Aufgabenwissen wird in Form von Zielen repräsentiert (definiert durch Ein- und Ausgabedeklarationen). Die Aufgabenschicht ermöglicht eine Zerlegung von Aufgaben, so daß Aufgaben der höheren Ebene aus elementaren Inferenzschritten realisiert werden können. Elementare Inferenzschritte werden auf der sogenannten Inferenzschicht definiert. Eine Dekomposition von Aufgaben erfolgt mit Hilfe von Kontrollstrukturen, die aus Programmiersprachen bekannt sind (`repeat-until`, `do-for-each-until`, etc.). In einer Aufgabenzerlegung ist das zu erstellende Produkt der Aufgabe (die Ausgabe) wiederum eine Eingabe für eine anderen Unteraufgabe. Damit ergibt sich ein Abhängigkeitsnetzwerk zwischen Aufgaben. Strategisches Wissen bestimmt, *welche* Ziele relevant sind, um ein bestimmtes Problem zu lösen. *Wie* die Ziele erfüllt werden, wird auf der Aufgabenebene bestimmt. Die Begriffe „Aufgabe“ (task) und „Aktion“ sind m.E. bei KADS als synonym anzusehen. Eine bestimmte Aufgabe bewirkt genau eine festgelegte Aktion, und eine Aktion löst eine bestimmte, damit verbundene Aufgabe.

In der KADS-Denkweise wird die System-Benutzer-Interaktion auf der Aufgabenebene definiert.[42] Aus Sicht von KADS kann ein wissensbasiertes System die Rolle eines intelligenten Agenten übernehmen. „Konventionelle“ Systeme werden als (halb-)passive Werkzeuge betrachtet. Aufgaben können entweder einem Agenten oder einem Benutzer zugewiesen werden, der mit dem (oder den) Agenten interagiert. Die Verteilung von Aufgaben kann auch dynamisch durch „Verhandlung“ (negotiation) bestimmt werden. Wenn ein Aufgabenmodell (Dekomposition und Verteilung von Aufgaben) erstellt ist, wird es verfeinert zu einem *Kooperationsmodell*. Die Verteilung von Aufgaben impliziert sog. *Transferaufgaben* im Abhängigkeitsnetzwerk. Eine Transferaufgabe kann als Platzhalter betrachtet werden, der anzeigt, wenn Kommunikation zwischen Agenten stattfinden soll. Sie werden als zusätzliche Aufgaben in die Aufgabenzerlegung eingefügt. Für jede Transferaufgabe wird definiert, wer die Initiative für den Transfer übernimmt. Weiterhin enthält eine Transferaufgabe Angaben darüber, wie Informationen, die durch das Abhängigkeitsnetzwerk definiert sind, tatsächlich zwischen Benutzer(n) und System(en) ausgetauscht werden. Breuker und de Greef unterscheiden zwischen einer Dialogverwaltungsebene und einer Präsentationsebene, allerdings werden in den entsprechenden Veröffentlichungen keine Details über die genauen Aufgaben und über interaktive Schnittstellen zu diesen Teilsystemen berichtet.

Das KADS-Modell der System-Benutzer-Kooperation kann zur manuellen Schnittstellenentwicklung verwendet werden. Es ist nicht dafür ausgelegt, die UIMS-Ebene formal an die Aufgabenebene zu koppeln, so daß Gestaltungsaspekte von Oberflächen automatisch aus einer Aufgabendekomposition und Aufgabenverteilung abgeleitet werden können. Weiterhin gibt es keine Unterstützung zur Koordination von verschiedenen Aufgaben und die Kombination der jeweils zu präsentierenden Informationen über den gesamten Dialog hinweg. Visuelle Dialogelemente sollten so konzipiert werden, daß mehrere Aufgaben damit bearbeitet werden können. Zu einem gewissen Grad wird eine Koordination auf der Dialogebene durch J.U. Möller behandelt [210]. Er argumentiert, daß die Folge von Transfer-

41. Siehe die Diskusion von Wielinga et al. in [339] S. 42 über die Beziehungen von KADS zu anderen Ansätzen zur Modellierung wissenbasierter Systeme.

42. Breuker und de Greef erwähnen, daß die System-Benutzer-Kooperation auch auf der strategischen Ebene in das konzeptuelle Modell von KADS integriert werden kann ([34], p. 51). Leider werden hierzu keine Details erläutert, so daß diese Ideen nicht nachvollziehbar sind.

aufgaben, die bei einer speziellen Aufgabenzerlegung möglich sind, eingeschränkt werden muß, so daß bestimmte Kohärenzkriterien erfüllt sind (z.B. bezüglich Thema und Fokusentwicklung). Die Folgen von Transferaufgaben, die für eine konkrete Anwendung erlaubt sind, hängen auch von der Art und Weise ab, wie die Transferaufgaben realisiert werden. Um nun Kohärenzkriterien abzuleiten, wird vorgeschlagen, ähnlich wie in Planungssystemen für die Generierung natürlicher Sprache, explizite Modelle für die Dialogfunktionen von Bestandteilen der Transfers sowie deren jeweilige rhetorische Funktion zu modellieren (näheres hierzu siehe unten). Allerdings wurden Details zu diesen Ideen nicht weiter ausgearbeitet. Der Gedanke wurde in HAMVIS aufgegriffen. Der Abschnitt 2.4 schildert weitere Arbeiten zur Repräsentation von Kommunikationswissen. Zuvor jedoch werden die Arbeiten zur Modellierung von formalen Dialogen aus Sicht der Mensch-Computer-Interaktionsforschung weiterverfolgt. Der nächste Abschnitt führt kurz in die bekannten Modellierungsformen ein und diskutiert deren Vor- und Nachteile.

2.2.4 Dialogmodellierung auf der syntaktischen Interaktionsebene

In der Klassifikation nach Foley [91] korrespondiert die Dialogmodellierung mit der syntaktischen Ebene der Mensch-Computer-Interaktion. In UIMS-Architekturen gibt es verschiedene Spezifikationsmethoden, um die Struktur von Benutzeraktionen zu beschreiben: Vorbedingungen für Aktionen, Folgen von Aktionen, Verzweigungen und Synchronisation von verschiedenen Interaktionszweigen usw. Ein wichtiges Thema ist die Granularitätsebene bzw. Abstraktionsebene mit der Spezifikationseinheiten (z.B. durch schrittweise Verfeinerung) ausgewählt werden.

Ein detaillierter Überblick über mögliche Spezifikationstechniken wird von Dix et al. [73] vorgestellt. Goetze [99] gibt nicht nur eine genaue Literaturübersicht, sondern diskutiert auch Vor- und Nachteile von verschiedenen Modellen. Er führt ein neues Modell ein, das speziell für die Beschreibungen von Dialogen mit multimedialer Oberfläche geeignet ist (genannt ODIS).[43] Der Vollständigkeit halber fasse ich in diesem Abschnitt kurz die bekannten Spezifikationstechniken für Dialogmodelle zusammen.

- Zustandsübergangsnetzwerke (state transition networks)

  Knoten repräsentieren Dialogzustände und Bögen beschreiben mögliche Übergänge zwischen Zuständen. Ein Dialogzustand ist durch einen Anwendungszustand und die ausführbaren Kommandos gekennzeichnet. Anwendbare Kommandos sind durch die Übergänge zwischen den Zuständen definiert. Zur Erhöhung der Modularität einer Modellierung werden vielfach Untergraphen eingesetzt. Einige Techniken gestatten auch die rekursive Untergraphenaktivierung. Übergänge können mit Bedingungen versehen werden, die in Abhängigkeit von Zustandsvariablen einen möglichen Übergang erlauben oder einschränken. Ein Problem bei Zustandsübergangsnetzen ist die inflationäre Zahl von Knoten und Kanten, die schon benötigt werden, um selbst einfache interaktive Systeme zu modellieren.

43. In diesem Bereich sind insbesondere Beschreibungen zur Synchronisation von Aus- und Eingaben verschiedener Medien oder Modalitäten notwendig (siehe auch [289]).

- Zustandsdiagramme (statecharts)

  Zustandsdiagramme wurden durch Harel [117] als ein allgemeiner Modellierungsmechanismus, der auf Graphen basiert, vorgestellt. Die mathematische Basis von Zustandsdiagrammen sind hierarchische Hypergraphen (higraphs). Der Vorteil von Zustandsdiagrammen gegenüber Zustandsübergangsnetzwerken ist, daß eine Explosion der Anzahl der Knoten vermieden werden kann. Übergänge können zusätzlich mit zeitlichen Verzögerungen und allgemeinen Prädikaten versehen werden (siehe die Diskussion in [95], S. 36f.). Vorbedingungen und Synchronisationsangaben für verschiedene Aktionen können ebenfalls definiert werden.

- Petri-Netze

  Ein nichtsequentieller Kontrollfluß mit Synchronisationspunkten kann zusammen mit einer hierarchischen Struktur durch Petri-Netze modelliert werden. Daher werden Petri-Netze auch in einigen UIMS zur Deklaration der Dialogstruktur eingesetzt. Unglücklicherweise werden Petri-Netze für reale Anwendungen schnell sehr groß und unübersichtlich, wenn nicht auf einer geeigneten Abstraktionsebene modelliert wird. Goetze und andere argumentieren, daß Petri-Netze zur Spezifikation von Unterdialogen auf der Fensterebene eingesetzt werden können ([95], S. 41).

- Kontextfreie Grammatiken

  Inspiriert durch Konstruktionstechniken für Übersetzer wurden auch Versuche zur Beschreibung von Dialogstrukturen mit kontextfreien Grammatiken beschrieben (Terminalsymbole definieren mögliche Benutzerhandlungen, Nichtterminale und Produktionen der Grammatik definieren dann die Dialogstruktur). Kontextfreie Grammatiken sind besonders geeignet für textuelle Kommandosprachen. Aufgrund ihrer sequentiellen Struktur sind sie aber kaum geeignet für die Beschreibung von Interaktionsmöglichkeiten in direktmanipulativen Oberflächen. In der Literatur sind einige Erweiterungsvorschläge zu finden, die kontextfreie Grammatiken mit semantischen Attributen und prozeduralen Codestücken versehen.

- Zustandsbäume (state trees) und hierarchische Dialogspezifikationen

  Eine Kombination von prozeduralen Ereignisbehandlungsangaben und hierarchischen Zustandsübergangsnetzwerken wurde ebenfalls zur Dialogspezifikation vorgeschlagen. Sequenzen und Verzweigungen können durch die Einführung von UND- und ODER-Knoten als hierarchische Dekompositionsoperatoren beschrieben werden (Goetze, [95] p. 47f.).

Goetze und Dix et al. erwähnen einige andere Techniken (Aktiven Daten und Einschränkungen, siehe hierzu auch die obige Diskussion über Techniken zur Kopplung von Anwendung und Oberfläche). Mit Dialogmodellen wird die Kopplung zwischen Anwendung und Oberfläche explizit gemacht und über die Aktivierung bzw. Deaktivierung von Oberflächenkommandos koordiniert.

Interessant ist, daß Aufgaben- und Dialogmodelle nicht unabhängig von einander definiert werden sollten. Wir werden im folgenden sehen, daß HAMVIS Abhängigkeiten zwischen Berechnungsfunktionen und Benutzeraktionen innerhalb einer Aktivität (im HAMVIS Sinne) durch ein Petri-Netz modelliert. Dieses muß jedoch nicht „von Hand“ definiert werden, sondern es wird automatisch aus der Aktionendekomposition einer Anwendung abgeleitet (näheres hierzu folgt später).

Aktionenmodelle sind die Grundlage für die tiefere Modellierung von Oberflächen mit visuellen Darstellungen. Wenn sich jedoch aus den Aktionenmodellen die Dialogmodellierung auf der syntaktischen Ebene und auch die zu präsentierenden Objekte automatisch herleiten lassen sollen, so muß ein Grundmodell an Aktionenkonzepten bereitgestellt werden, mit denen der Modellierer die applikationsspezifischen Modelle definieren kann. Dieses kann für eine spezielle Klasse von Anwendungen erfolgen. Für die Konzeption von HAMVIS wird in dieser Arbeit die Klasse der Designumgebungen betrachtet. XKL ist eine spezielle Anwendung aus dieser Klasse. Um nun einen Überblick über andere Ansätze zu geben, möchte ich als Grundlage die Konzeption von anwendungsklassenspezifischen Designumgebungen anhand der Arbeiten von Fischer et al. schildern.

2.2.5 Anwendungsklassenspezifische Designumgebungen

Designprobleme sind i.a. schwer formalisierbar. Auch die XKL-Anwendung muß so konzipiert werden, daß wesentliche Designentscheidungen am Bildschirm von dem Systembenutzer (also dem Kundenberater) getroffen werden können. Für die Entwicklung solcher Designumgebungen wurden verschiedene Vorschläge gemacht. Zu beachten ist, daß die Designdomäne für die im Kontext von HAMVIS durchgeführten Arbeiten nur als Anwendungs- und Leitdomäne dient. Es geht in dieser Arbeit *nicht* darum, eine ausgefeilte Entwicklungsumgebung für die *manuelle* Programmierung von Designsystemen zu erstellen. Trotzdem möchte ich an dieser Stelle einige Arbeiten zu diesem Thema diskutieren. Wir betrachten die Ansätze von Fischer und seiner Arbeitsgruppe zur Entwicklung von domänenspezifischen Designumgebungen.

Fischer et al. konzipieren in [85] eine Designumgebung für die Einrichtung von Küchen. Sie argumentieren dafür, daß Designer domänenspezifische Einrichtungsgegenstände wie z.B. Herde, Kühlschränke, Aufwäschen usw. frei positionieren wollen. Einrichtungsgegenstände werden durch Piktrogramme dargestellt und können aus einer Palette in den „Designraum“ gezogen und dort an beliebiger Stelle abgelegt werden. Das System ist mit Absicht „konservativ“ konzipiert, so daß auch die offensichtlich ungeschickte Plazierung eines Objekts nicht unterbunden wird. Zur Illustration der Ideen ist hier ein Beispiel, in dem eine Aufwäsche in einer Küche plaziert wird, als Abbildung 23 wiedergegeben.

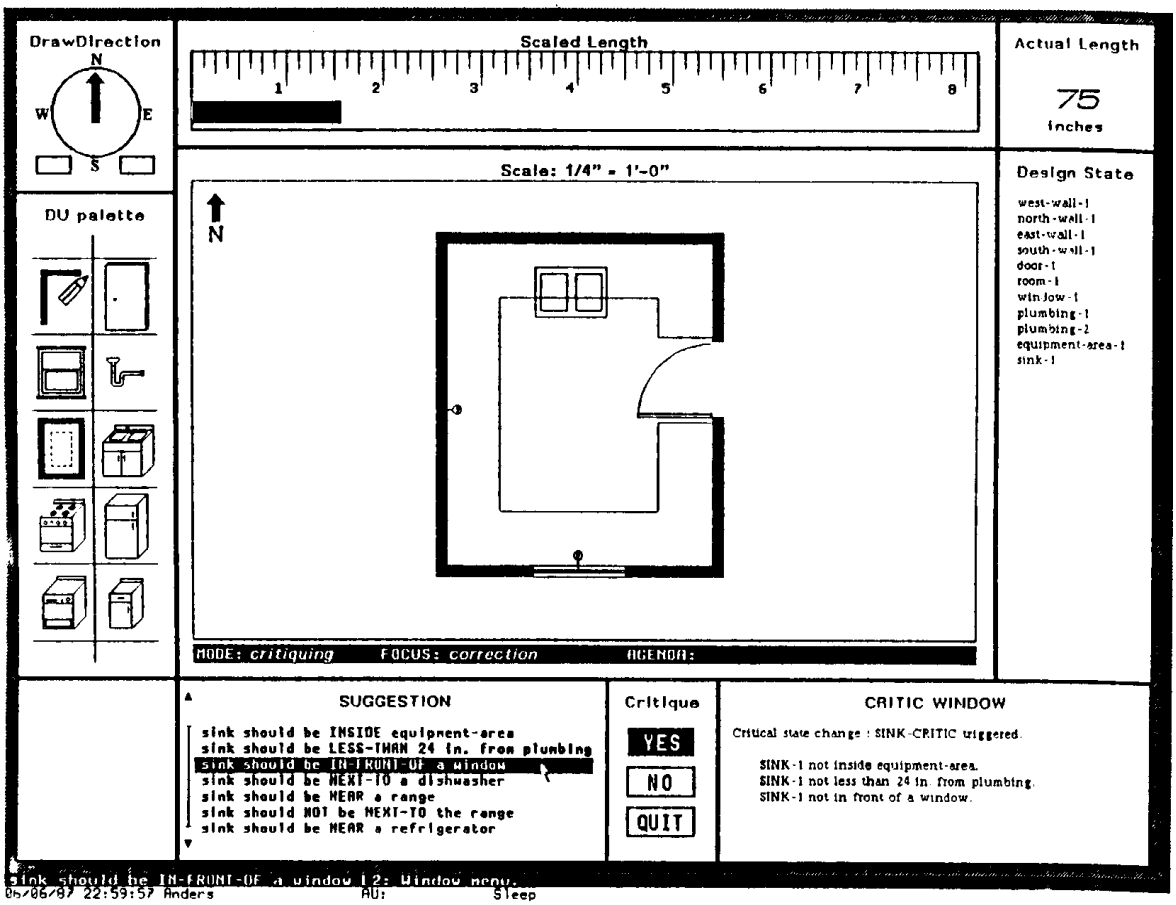


Abbildung 23. Domänenspezifische Designumgebung mit freier Plazierung einer Aufwäsche (nach [85]).

Das Beispiel illustriert die Verwendung von automatisch im Hintergrund ablaufenden Prozessen zur Bewertung der aktuellen Konstruktion. Verbesserungsvorschläge der „Kritikers“ werden in einem kleinen Teilfenster ausgedruckt und können auf Bedarf genau inspiziert und ggf. durch Designänderungen berücksichtigt werden. Als Erklärung wird die „Argumentationsstruktur“ in einem speziellen Darstellungsfenster ausgegeben ([85] p. 273).

In neueren Arbeiten wird dieser Gedanke erweitert und in einem Tutor-Kontext auch zur Unterweisung von Designern eingesetzt [87]. In Abbildung 24 wird ein Beispiel gezeigt, in dem einem Designer ein weniger offensichtlicher Designfehler unterläuft.

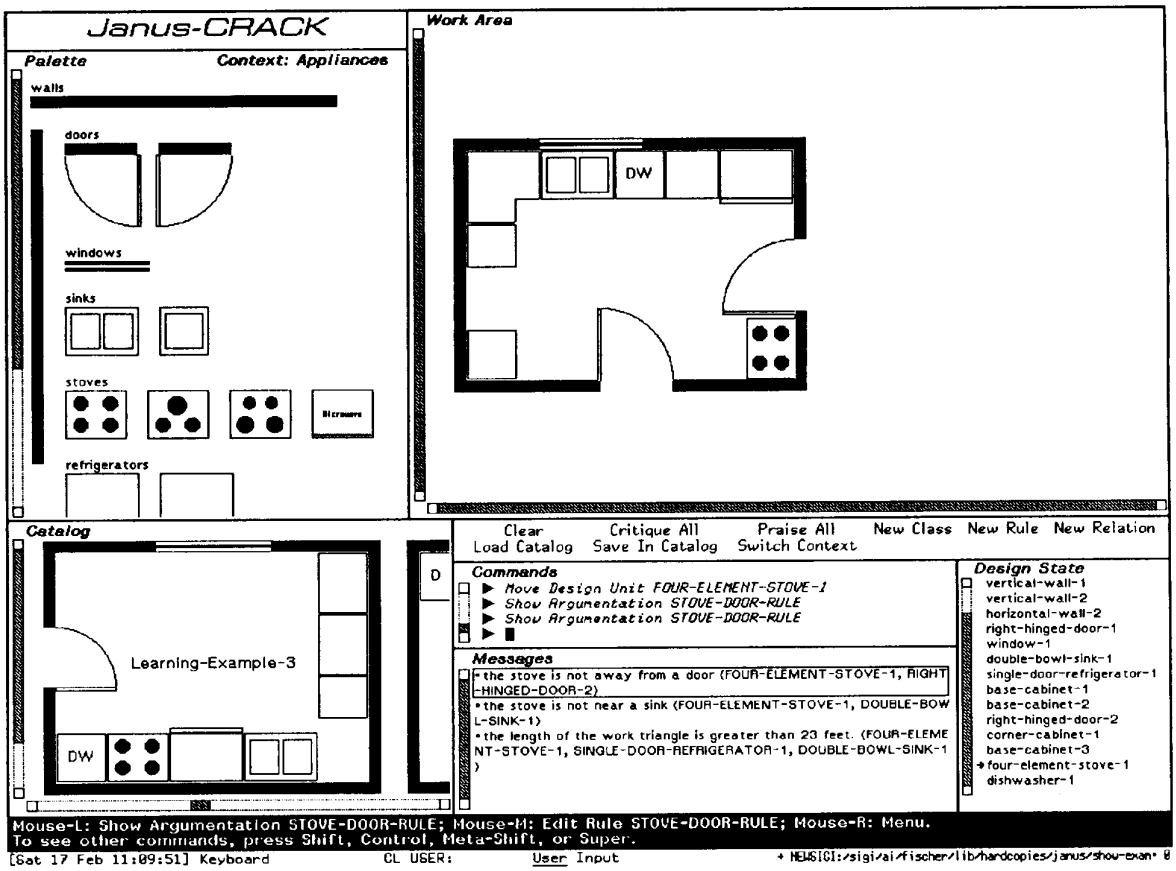


Abbildung 24. Eine Anwendung zur Vermittlung von Design-Kenntnissen (siehe Text).

In diesem Beispiel wird ein Herd zu nahe bei einer Tür plaziert. Der Designer kann sich durch Mausklick ein detaillierte Erklärung anzeigen lassen (Abbildung 25).

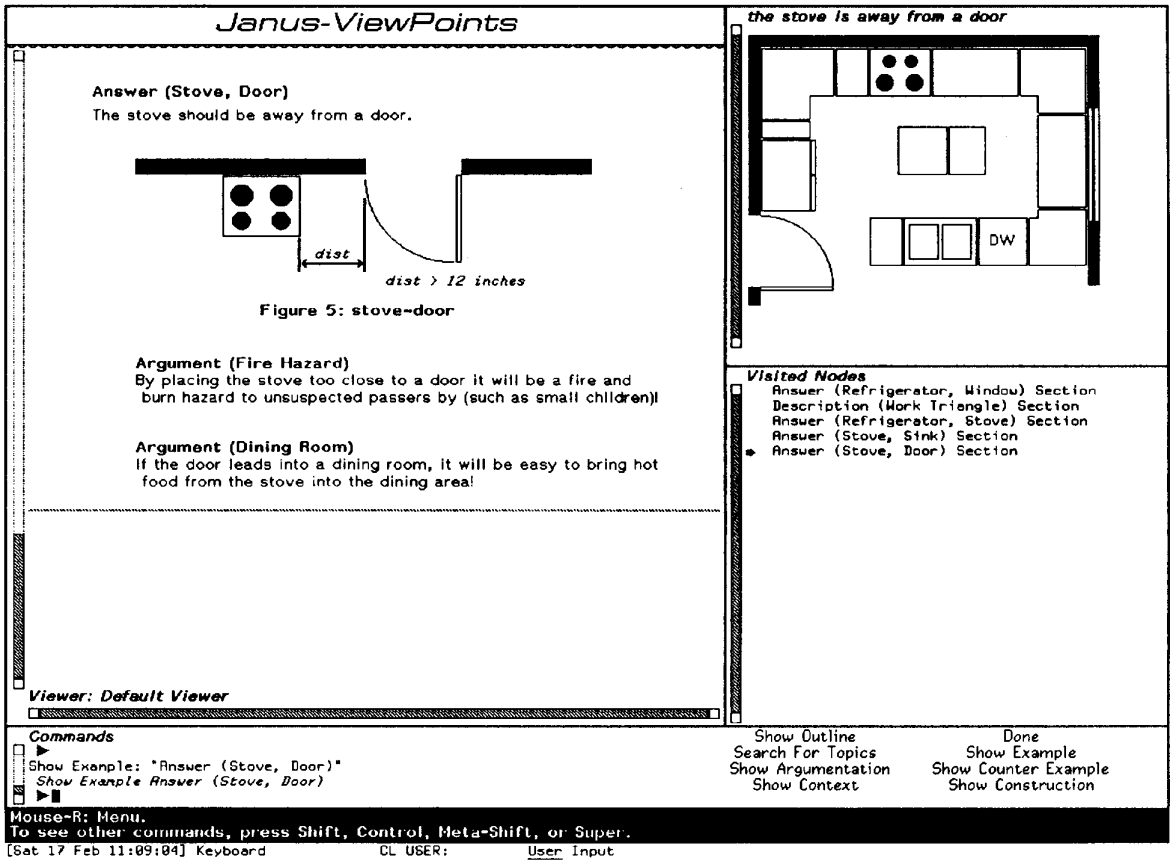


Abbildung 25. Fenster zur Präsentation von Erklärungen zu domänenspezifischen Designregeln.

Erklärungen werden dem Designer mit Hilfe eines Hypertext-Systems präsentiert. In einer weiteren Arbeit wird auch die Spezifikation der Aufteilung von Teilfenstern in einem Gesamtrahmen als Designaufgabe aufgefaßt (Lemke und Fischer [172]).

Weiterhin wurde untersucht, inwieweit sich der Designprozeß am Computer durch „Einblenden“ von vorher konstruierten Fällen unterstützen läßt. Die Bestimmung von relevanten Fällen wird z.B. in einem iterativen Verfahren (retrieval by reformulation) durchgeführt (HELGON: Fischer und Nieper-Lemke [86]). Zur Beschreibung von Designartefakten wird das beschreibungslogische Repräsentationssystem KANDOR [242] verwendet. HELGON verwendet die KANDOR-Klassifikationsalgorithmen, um ausgehend von einer Beschreibung (mit Konzepte, Rollen und Rolleneinschränkungen) „passende Fälle“ zu bestimmen (siehe auch die Arbeiten von Patel-Schneider, Brachman und Levesque zur Anwendung von Beschreibungslogiken im Bereich des Information Retrieval [243] sowie die Arbeiten von Köhler [156], [157]). Beschreibungslogische „Anfrageformeln“ können in HELGON iterativ verfeinert oder verändert werden.

In neueren Arbeiten wird versucht, Anfragen an eine Falldatenbasis automatisch aus dem aktuellen Designartefakt zu bestimmen (Fischer und Nakakoji [88], [89]). Unter Hinweis auf die Probleme bei der Formulierung von formalen Spezifikationen werden nun auf der Grundlagen einer partiellen

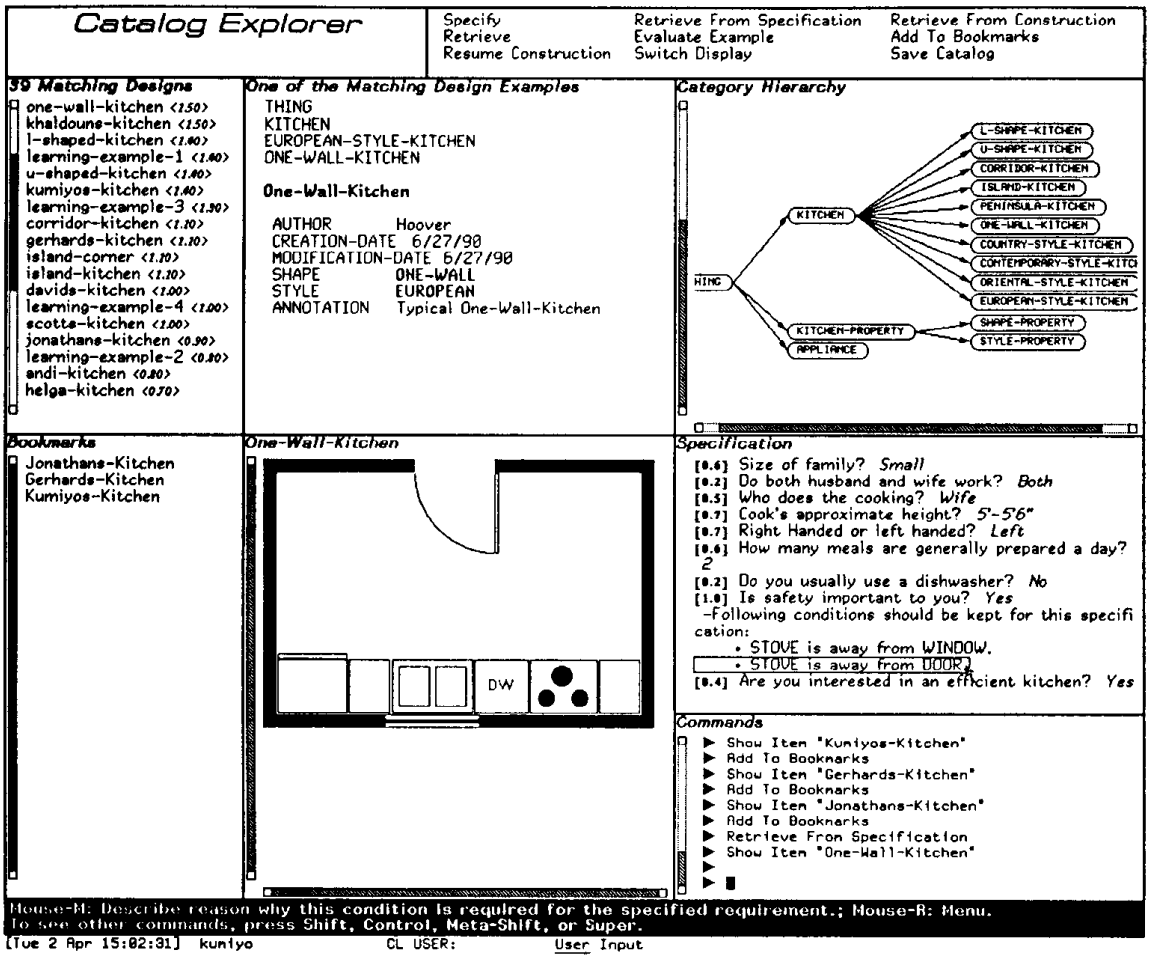


Abbildung 26. Darstellungsfenster zur Präsentation von automatisch berechneten Designbeispielen („Fällen“).

Spezifikation durch den aktuellen Konstruktionszustand mit Hilfe von Regeln und speziellen Ähnlichkeitsmaßen vorher schon einmal bearbeitete Fälle aus einem Katalog bestimmt (Abbildung 26). Für den Designprozeß wurden interaktive Oberflächen entwickelt, wobei der Programmieraufwand trotz der fortschrittlichen Symbolics-Entwicklungsumgebung beträchtlich gewesen sein dürfte.

In dieser Arbeit kann und soll keine vergleichbare Designumgebung entwickelt werden. Die Designdomäne dient im HAMVIS-Kontext zur Illustration der Arbeiten zur Formalisierung der Visualisierungsgenerierung für Oberflächen, die z.B. in das hier beschriebenen Rahmenwerk von Fischer integriert werden könnten.

Fischer et al. betonen, daß bei der Entwicklung von Designoberflächen gleich von Anfang an damit gerechnet werden muß, daß sich das notwendige Wissen ändern kann, daß neues Wissen erfaßt werden muß usw. (Fischer et al. [90]: seeding, evolutionary growth und reseeding). Unter Verweis auf eine Anwendungsdomäne, in der Netzwerke von Computersystemen konfiguriert werden sollen, wird darauf hingewiesen, daß Entwicklungsumgebungen so gestaltet werden müssen, daß Änderungen und Hinzufügungen leicht möglich sind.

Zusammenfassend läßt sich sagen, daß bei den hier aufgeführten Arbeiten von Fischer grundsätzlich versucht wird, den Designer möglichst zu unterstützen, ihm aber keine (oder nur wenige) Entscheidungsmöglichkeiten zu entziehen. Fischer zeigt auf, daß es vorteilhaft ist, einem Benutzer nicht vorzugeben, dieses oder jenes zu tun, sondern eine Menge von Handlungsalternativen zu unterstützen, aus denen er auswählen kann (s.o.).[44] Es wird z.B. nicht von dem System ausgedruckt: „Jetzt kann eine Küche plaziert oder verschoben werden“. Dieses ist bestenfalls in Oberflächen für absolute Anfänger geeignet, da schon nach kurzer Zeit die Ausgaben (und deren Betrachtung) als Störung empfunden werden. Die möglichen Handlungsalternativen in einer bestimmten Konfigurationssituation ergeben sich im Sinne der direkten Manipulation idealerweise dadurch, daß bestimmte graphische Objekte maussensitiv sind. Die Möglichkeit, ein neues Objekt in ein Küchendesign zu integrieren, wird beispielsweise dadurch unterstützt, daß entsprechende graphische Objekte in einer Palette dargestellt werden und z.B. durch Ziehen-und-Fallenlassen-Gesten auf einen Plazierungsbereich gezogen werden können (siehe Abbildung 23 und Abbildung 24). Wird die Maus auf ein Objekt in der Palette bewegt, so wird in der Mausdokumentationszeile eine textuelle Beschreibung der Aktion gegeben. Auch für HAMVIS soll ein ähnliches Vorgehen unterstützt werden.

Das hier kurz skizzierte Rahmensystem von Fischer unterstützt allerdings nicht die Erstellung der notwendigen Visualisierungen aus einem wissensbasierten Domänenmodell unter Berücksichtigung von geometrischen Daten und begrifflichen Strukturen. Die in den Abbildungen gezeigten Oberflächen sind „von Hand“ programmiert.

Der Ansatz von Fischer kann auch mit Ansätzen zur Konzeption eines intelligenten CAD-Systems verglichen werden. In CAD-Ansätzen standen eine lange Zeit die geometrischen Daten im Vordergrund, während begriffliche Informationen z.B. zur Modellierung des Produktionsprozesses erst mit Einführung des CIM-Gedanken in den Vordergrund rückten. Inzwischen sind auch erweiterte CAD-System zur integrierten Geometrie- und Produktdatenmodellierung kommerziell verfügbar. ICAD™ enthält z.B. eine Frame-orientierte Modellierungssprache auf der Basis von CLOS und bietet Schnittstellen zu gängigen CAD-Formaten (z.B. STEP [19] [20] [125]). Neben der Benutzungsschnittstelle für die Entwicklung von CAD-Modellen wird auch eine Umgebung für die manuelle Erstellung von speziellen Benutzungsoberflächen für Änderungen in der „Produktionsphase“ bereitgestellt.

2.2.6 Diskussion

In diesem Kapitel wurde ein Spektrum von Arbeiten aufgespannt, das von theoretischen Arbeiten aus dem Bereich Mensch-Maschine-Interaktion bis hin zu praktischen Designumgebungen und Designsystem-Entwicklungsumgebungen reicht. Es wurde deutlich, daß eine Generierung von Visualisierungen für Benutzungsschnittstellen nicht ohne Blick auf methodische Techniken zur Anwendungsentwicklung verfolgt werden kann. Am Anfang dieses Kapitels wurde die einflußreiche Theorie von Norman über konzeptuelle Modelle (Designmodell, System, Systembild) nebst Erweiterungen (Implementationsmodell) erläutert. Nach Betrachtung von Arbeiten zur Aufgaben- und Dialogmodellierung wird deutlich, daß der Begriff Designmodell als wesentliche Komponente des

44. Ein anderer Ansatz, bei dem mehr Konstruktionsschritte durch ein automatisches System übernommen werden und der Benutzer weniger „freizügig“ agieren kann, wurde im Projekt PROKON entwickelt (Günter [106]). Zentraler Bestandteil der wissensbasierten Konfiguration ist im PROKON-Ansatz (wie auch bei Fischer) die konzeptuelle Begriffshierarchie.

Systementwurfs etwas zu allgemein ist. Es kann kaum ausreichen, eine einzige „Modellierungssprache“ für die Designebene aufzustellen. Zur Entwicklungszeit müssen verschiedene Aspekte einer Anwendung durch unterschiedliche Modelle erfaßt werden.

Die Diskussion der konzeptuellen Modelle von Norman und der Erweiterungen von Holz ergab die Notwendigkeit für die abgestimmte Definition eines Designmodells, eines Implementationsmodells und eines Systembilds. Holz schlägt vor, für die Aufstellung von Entwurfsmodellen Konzepte der Arbeitspsychologie zu verwenden. Dieses erscheint mir sinnvoll, da sich hieraus eine Diskussionsgrundlage für die verschiedenen, am Entwurf beteiligten Personengruppen ergibt. Der Weg zu einer lauffähigen Version ist aber immer noch sehr lang. Auch auf der Ebene des Entwurfs muß m.E. nach Mitteln und Wegen gesucht werden, verschiedene Aspekte der Entwurfsmodelle (Designmodell, Implementationsmodell, Systembild) *formal* zu beschreiben und für die automatische Weiterverarbeitung zugängig zu machen. Die Dienste, die ein UIMS (inklusive Erweiterungsbibliotheken usw.) bietet, müssen dabei während der Entwurfsphase zugreifbar gemacht werden. Mit HAMVIS wurde daher ein Ansatz entwickelt, mit dem einem Oberflächendesigner diese Dienste zur Verfügung gestellt werden und weiterhin eine methodische, abgestimmte Entwicklung von Anwendung und Oberfläche unterstützt werden kann.

Nach meiner Einschätzung bilden Aufgaben- bzw. Aktionenmodelle die Basis für die Entwicklung von interaktiven Visualisierungen. Hierzu werden allerdings „tiefe Modelle“ benötigt, um die erforderlichen Einschränkungen für die Darstellung auch formal ableiten zu können. Daher ist es erforderlich, Aktionenmodelle nicht auf der Ebene der Manipulation von graphischen Elementen zu formulieren, sondern auf die Manipulation von *Anwendungsobjekten* zu beziehen, d.h. anstelle der syntaktischen Ebene (z.B. Klicken auf ein Rechteck) ist die *semantischen Modellierungsebene* bedeutsam (z.B. Lokalisierung eines räumlichen Objekts). In diesem Zusammenhang läßt sich ein Vergleich mit der Sprechakttheorie ziehen. Bei der Interpretation einer Äußerung ist neben der oberflächennahen Deutung (z.B. „Kannst Du mir sagen, wie spät es ist?“ interpretiert als Frage nach einer eigenen Einschätzung der Fähigkeiten des Gesprächspartners) auch die illokutionäre Ebene des Sprechaktes für die Interpretation relevant (ein Gesprächspartner will die aktuelle Zeit wissen und von dem anderen die Auskunft erlangen). In Übertragung dieser Terminologie läßt sich sagen, daß Aktionen des Benutzers (während der Entwurfsphase) zunächst einmal auf der illokutionären Ebene modelliert werden müssen, damit die zu präsentierenden Informationen samt Darstellungseinschränkungen in einem Handlungsmodell ausgedrückt werden können. Die Wahl der Ausführungsmöglichkeit dieser Handlungen auf der Graphikebene (lokutive Ebene) bedingt weitere Einschränkungen für die Gestaltung von Visualisierungen.

Bei der Konzeption des zur Aktionenmodellierung verwendeten Vokabulars herrscht recht große Verwirrung. Der Begriff „Aufgabe“ (task) wird meist synonym zu „Aktion“ oder „Ziel“ verwendet. Ein Ziel kann jedoch durch mehrere Aktionen erreicht werden, und eine Aufgabe kann mehrere Ziele umfassen. Auch wissensbasierte Ansätze, in denen jeder Teilschritt als Aufgabe (task) bezeichnet wird (entweder durch ein Programm oder durch einen Benutzer zu erfüllen), bringen hier keine Klarheit. Dieses wird auch von Chandrasekaran et al. hervorgehoben [43], die den Aufgabenbegriff aus der Perspektive der wissenbasierten Softwareentwicklung beleuchten.

Für HAMVIS möchte ich auf unterer Ebene zwischen Benutzeraktionen und automatischen Berechnungs- bzw. Speicherfunktionen unterscheiden. Berechnungs- und Speicherfunktionen (zusammenfassend auch Anwendungs- oder Applikationsfunktionen genannt) können durch wissenbasierte

Komponenten realisiert sein, müssen es aber nicht. Anwendungsfunktionen haben Eingangsparameter und liefern ein oder mehrere Werte. Sie sind durch eine Beschreibung der Parameter und Wertetypen definiert, werden aber ansonsten als Blackbox betrachtet. Benutzeraktionen können auf gegebene Objekte Bezug nehmen („Eingangsparameter“) und produzieren neue Werte („Ausgangsparameter“), modifizieren Anwendungsobjekte aber nicht direkt. Bei einer Verschiebung wird beispielsweise nur die neue Position bestimmt, das Objekt wird aber nicht modifiziert. Dieses erfolgt durch eine „nachgeschaltete“ Speicherfunktion. Die „Aufgaben“ von Benutzern definieren sich aus den zu liefernden Werten von Benutzeraktionen.

Im Gegensatz zu den vorgestellten Entwurfstechniken aus dem Bereich Mensch-Computer-Interaktion scheint es mir weiterhin notwendig, aus einer Aktionenmodellierung durch formale Verfahren automatisch Einschränkungen für die Darstellung ableitbar zu machen, wobei Aktionen auf einer hohen Granularitätsstufe modellierbar sein sollen.

Aus Sicht der direktmanipulativen Interaktion sind Anwendungsfunktionen transparent zu halten. Auf einer höheren Ebene sollte also von einer Benutzeraktion gesprochen werden (sog. zusammengesetzte Aktion). Mit HAMVIS erfolgt die Strukturierung der Anwendung durch eine hierarchische Aktionenmodellierung (hierfür wird eine interaktive Oberfläche verwendet). Die zentrale Idee von HAMVIS ist, vordefinierte Konzepte für Benutzeraktionen bereitzustellen, die für eine Klasse von Anwendungen automatisch auf UIMS-Dienste abgebildet werden können. Elementare Aktionenkonzepte legen die zur Ausführung der Handlungen nötigen Objekte fest und bestimmen dadurch, welche Objekte kommuniziert werden müssen. Ein Umweg über Informationsverarbeitungsziele erfolgt nicht.

Die Dialogstrukturierung einer Anwendung im groben ist durch die aufgestellte Aktionendekomposition festgelegt. Innerhalb der Aktionenmodellierung ist dadurch auch die Verteilung von Aufgaben für den Benutzer und für automatische Berechnungsfunktionen definiert. Eine dynamische Verteilung von Aufgaben (bzw. eine Verhandlung über die Aufteilung) zur Laufzeit möchte ich im Kontext von HAMVIS nicht betrachten. Die von einigen Autoren betonte Trennung zwischen exekutiven und deskriptiven Arbeitsformen läßt sich zumindest in der für HAMVIS betrachteten Anwendungsklasse nicht erkennen, so daß auch die Betrachtung einer Berechnungsfunktion als selbständig tätiger Agent im HAMVIS-Kontext nicht von entscheidender Bedeutung ist. Der Prozeß des Entwurfs von Visualisierungen für Benutzungsschnittstellen wird als Festlegung und Ausbalancierung von Einschränkungen betrachtet (vgl. die Argumentation von Holz). Es ergeben sich Einschränkungen für die Gestaltungs- und Kombinationsmöglichkeiten von Visualisierungen für verschiedene Aktionen sowie Anforderungen an die vom Anwendungsentwickler bereitzustellenden Modelle.

Meiner Ansicht nach sollten Einschränkungen formal repräsentiert und automatisch verwaltet werden. Durch HAMVIS werden daher darauf abgestimmte Modellierungsformen für den Anwendungsentwickler zur Aufstellung von Domänenmodellen sowie vordefinierte Modelle für Aktionenkonzepte und aufgabenangemessene Dialogstrukturen bereitgestellt. Mit Aktionenkonzepten für elementare Benutzerfunktionen sind in HAMVIS Anforderungen für die Darstellung verbunden, die in dem Dialogmodell, das für eine Anwendung erstellt wird, vermerkt werden. Es werden nicht nur die primären, manipulierten Objekte dargestellt (vgl. die Werkbankmetapher), sondern es wird gefordert, daß aus den domänenspezifischen Modellen z.B. geeignete Referenzsysteme mit Landmarken usw. ermittelt werden können. Neben diesen allgemeinen Darstellungsanforderungen müssen noch spezielle, sich aus der konkreten Applikation ergebende Informationsverarbeitungsprozesse unterstützt werden.

Bei der von HAMVIS unterstützten Anwendungsklasse ist es nicht ausreichend, nur bestimmte spezielle Objekte oder Prozeßvariablen (wie bei GRADIENT) darzustellen. Interessant sind im XKL-Kontext z.B. spezielle räumliche Beziehungen wie Überschneidungen von notwendigen Freiräumen (z.B. Arbeitsbereichen vor Küchen, Ausstiegszonen vor Türen usw.). Die notwendigen Funktionen müssen durch das Systementwicklungsteam spezifiziert und den Aktionen zugeordnet werden. Zur Entwicklungszeit kann allerdings schon die Darstellungsform für die zur Laufzeit bestimmten Objekte ermittelt werden.

Holz betont die Notwendigkeit, während der Entwurfsphase einer Anwendung verschiedene Personengruppen (Endbenutzer, Manager, Programmierer, Oberflächendesigner usw.) in die Diskussion einzubeziehen. Dieses wird durch den HAMVIS-Ansatz im Prinzip unterstützt. Jedoch wird nicht der Versuch gemacht, Repräsentationsformen und Denkweisen zu verwenden, die ohne Einarbeitung für jede Personengruppe intuitiv zugängig sind.

Weiterhin betont Holz die Wichtigkeit des Vergleichs von Entwurfsalternativen. Obwohl in der momentanen Konzeption von HAMVIS die Aktionenmodellierung nicht in mehreren Varianten erstellt werden kann, soll die Erstellung von Alternativen beim Dialogmodell unterstützt werden. Es gibt ggf. mehrere Möglichkeiten, die im Dialogmodell verwalteten Einschränkungen zu erfüllen. HAMVIS muß daher gestatten, verschiedene Alternativen zu verwalten und zu bewerten. Insbesondere zur Einbeziehung des Benutzers in den Erstellungsprozeß ist es notwendig, aus einer Dialogmodellalternative automatisch auch eine Oberfläche generieren zu können. Die von HAMVIS bereitgestellten Aktionen- und Dialogmodellkonzepte sind so zu gestalten, daß dieses möglich ist. Aus dem Dialogmodell muß automatisch ein Laufzeitsystem mit UIMS-Anschluß erzeugt werden können. Die Diskussion der hierfür benötigten UIMS-Dienste hat gezeigt, daß bei modernen UIMS-Konzepten der Schritt zur technischen Realisierung klar aufgezeigt werden kann.

Obwohl die Kommunikationsmetapher im HAMVIS-Kontext nicht auf eine Agentensicht ausgerichtet ist, macht eine Betrachtung von Kommunikationsaspekten im Dialogmodell auch bei „konventionellen“ Oberflächen Sinn, sofern man nicht einen reinen Werkbankansatz verfolgt – wie z.B. bei objektorientierten Zeichenprogrammen. Dem Benutzer werden nicht nur die zu manipulierenden Objekte präsentiert, sondern auch die für die Durchführung der jeweiligen Handlungen notwendigen Informationen. Diese wiederum müssen nach Kommunikationsgesichtspunkten strukturiert werden. Im HAMVIS-Kontext wird diese Struktur explizit repräsentiert. Sie dient der Verwaltung von Einschränkungen im Dialogmodell und bildet die Ausgangsbasis für die Bestimmung von vorgeschlagenen Werten für Zeichenparameter (Farben, Strichdicken, metagraphische Symbole). Im folgenden wird daher eine Übersicht über Forschungsarbeiten gegeben, die sich mit Aspekten der graphischen Kommunikation befassen. Es werden Konzepte von Generierungssystemen für Präsentationen auch aus historischer Sicht geschildert. Auch im Bereich der Präsentationssysteme belegen neuere Forschungsarbeiten die Notwendigkeit der Abstimmung einer Präsentation auf die damit zu erledigenden Aufgaben und zeigen auf, welche Gesichtspunkte bei einer Komposition von Visualisierungen zu berücksichtigen sind.

2.3 Graphische Kommunikation

In diesem Abschnitt wird die Verwendung von Graphiken aus der Kommunikationsperspektive erläutert. Graphische Elemente können als Sprachelemente angesehen werden, die kombiniert werden, um bestimmte kommunikative Ziele zu erreichen:

> *„While the text speaks with words, the graphic figure speaks with form. Although subject matter provides the substance for the figure, what the figure actually says as a visual statement depends more on the communicative aim which shapes this raw material into a purposeful visual idea [...]. The capabilities and limitations of visual languages are themselves decisive factors in determining the kinds of ideas the figure may show about a given subject.“ (Bowman 1968, [30])*

In seinem für Graphikdesigner gedachten Buch stellt Bowman verschiedene Möglichkeiten zur Lösung von praktischen Kommunikationsproblemen mit graphischen Ausdrucksformen und Konzepten vor. Wie alle Sprachen haben auch visuelle Sprachen ihre Stärken und Schwächen (siehe z.B. Wahlster [329]).

Ausgehend von frühen Arbeiten zur Definition eines graphischen Vokabulars sowie zur Formulierung von formalen Methoden zur Definition von visuellen Notationen werden in diesem Abschnitt die Hauptideen von Systemen zur automatischen Generierung von graphischen Präsentationen skizziert. Die Techniken, den kombinatorischen Suchraum bei der Präsentationskomposition zu definieren, werden mit den im HAMVIS-Kontext notwendigen Verfahren zur Verwaltung von Einschränkungen verglichen.

2.3.1 Definition eines graphischen Vokabulars

Zur gleichen Zeit wie Bowman (ca. 1967) arbeitete Bertin an seiner *Semiologie Graphique*, die als Buch 1981 erschienen ist [21]. Um die Erstellung von Graphiken zu formalisieren, untersuchte Bertin die Präsentation von Informationen mit Geschäftsgraphiken (Kuchendiagramme, Balkendiagramme usw.) sowie auch Karten. Er unterscheidet zwischen sechs Dimensionen von „visuellen Variablen“ (Abbildung 27).

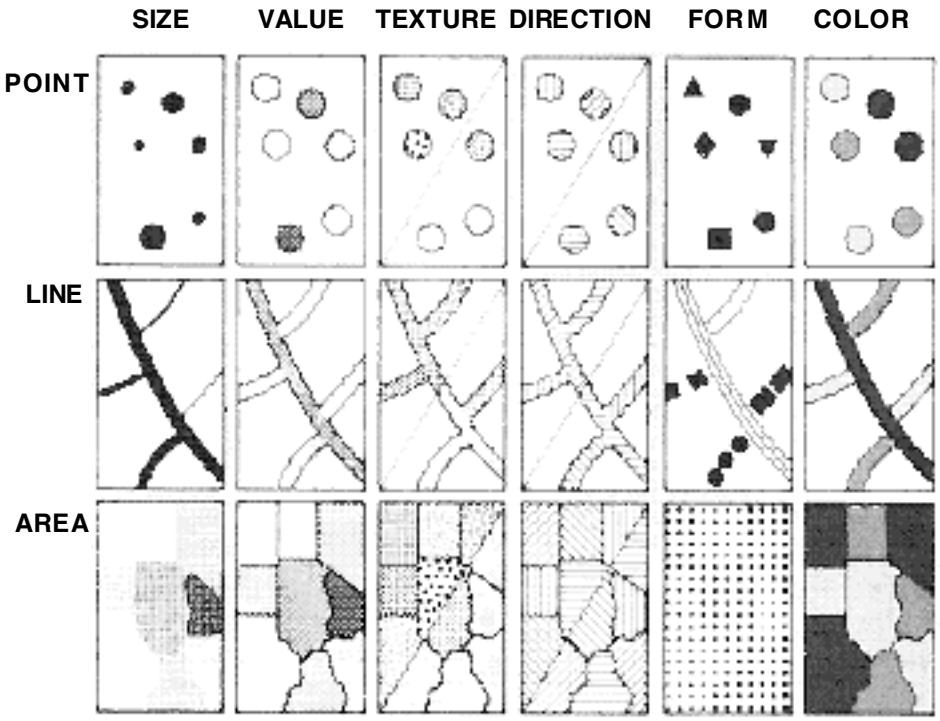


Abbildung 27. Visuelle Variablen (aus [205]).

Bei der Analyse von Eigenschaften der Ebene für das Zeichnen von Punkten, Linien und Regionen entdeckte er zum Beispiel, daß entlang einiger Dimensionen eine Ordnung von Elementen erfolgt (Größe, Helligkeitswert), während entlang anderer Dimensionen keine Ordnung induziert wird (Texture, Richtung, Form, Farbe). Weiterhin stellte er heraus, daß die Variablen voneinander abhängen. Wenn die Größe eines Objektes zu klein wird, können unterschiedliche Formen nicht mehr unterschieden werden usw. Andere Einsichten seiner Arbeit sind ebenfalls sehr interessant, aber da die Forschungsarbeiten zu HAMVIS nicht auf die Produktion von Karten und Geschäftsgraphiken ausgelegt sind, möchte ich hier keine Details diskutieren. Leider ist auch die Formalisierung der Prinzipien und Theorien von Bertin, um sie in automatische Generierungsprozesse für Graphiken integrieren zu können, außerordentlich schwierig.

2.3.2 Visuelle Notationen

Zur Formalisierung der Definition des Begriffs „visuelle Notation“ wurden verschiedene Ansätze vorgestellt. Selker und Koved [281] definieren die Menge von räumlichen, oberflächenorientierten und zeitlichen[45] Darstellungstechniken, die in einer visuellen Notation verwendet werden können, als

45. In Zeichentrickfilmen werden verschiedene Techniken zur Übertreibung eingesetzt. Zum Beispiel werden häufig auch starre Körper gestreckt, wenn sie beginnen, sich zu bewegen. Wenn sie zum Stehen kommen, werden sie wieder gestaucht [48]. Eine weiteres wichtiges Thema ist Konditionierung. Unter Konditionierung wird in Zeichentrickfilmen das Prinzip der Vorbereitung des Zuschauers auf die Haupthandlung durch vorgeschaltete Nebenhandlungen verstanden ([48] p. 3). Der intendierte Effekt ist, daß die Haupthandlung nicht überraschend kommt, bzw. unbemerkt bleibt.

visuelles Alphabet (vgl. den Begriff der visuellen Variablen von Bertin). Visuelle Ausdrücke werden durch eine Syntax generiert, die zwischen den folgenden syntaktischen Kategorien unterscheidet:

- Position
  - relativ (sequentiell, metrisch, die Orientierung betreffend)
  - interagierend (eingebettet, überschneidend)
  - denotiert (verbunden, beschriftet)
- (Relative) Größe
- Zeitlicher Verlauf (z.B. Animation, Geschwindigkeiten)
- andere Techniken (z.B. Einschränkungen)

Eine Vielzahl von Techniken zur Spezifikation der Syntax von visuellen Notationen wurde veröffentlicht. Siehe hierzu den Überblick von Chang et al. über Visuelle Programmiersprachen ([44], [45], [46], [47]). Lohse et al. geben eine Klassifikation von visuellen Repräsentationen [181]. Helm und Marriott spezifizieren visuelle Notationen mit CLP(*R*) [137] und stellen eine deklarative Semantik vor, die auf der Semantik von logischen Programmen mit Beschränkungen basiert.[46]

Durch visuelle Notationen lassen sich Beschreibungsformen für Objekte deklarativ definieren. Dieses ist insbesondere dann wichtig, wenn die visuelle Notation nicht nur vom Objektkonzept, sondern auch von den Objekteigenschaften abhängt. In der XKL-Anwendung läßt sich beispielsweise eine Küche, dem „Cabin Configuration Guide“ des Flugzeugherstellers folgend, mit Strichen zur Markierung der Trolley-Stellplätze kennzeichnen usw. (Abbildung 28).

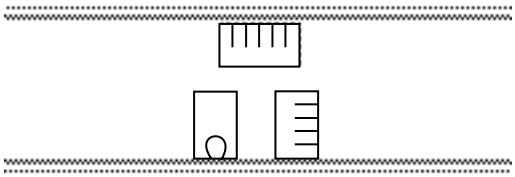


Abbildung 28. Unterschiedliche Darstellung von Küchen und Waschräumen (siehe Text).

Bei Einführung von domänenspezifischen visuellen Notationen für spezielle Konzepte brauchen z.B. Küchen und Waschräume in einer Darstellung nicht mehr durch Zeichenattribute wie z.B. Farbe oder Füllmuster unterscheidbar gemacht zu werden.[47] Eine visuelle Notation kann demnach als besondere

46. Deklarative Spezifikationen von visuellen Notationen wie sie von Helm und Marriott vorgestellt wurden, können – im Prinzip – sowohl für die Generierung als auch zum Parsing verwendet werden. Allerdings darf nicht vergessen werden, daß wegen des großen Suchraumes in realen Anwendungen zumindest für den Parsing-Prozeß spezielle Verfahren bereitgestellt werden müssen, die auch prozedurale Kontrollaspekte berücksichtigen (siehe z.B. die Arbeiten von Pasternak zur Zeichnungsinterpretation [239] [240] [241]). Deklarative visuelle Grammatiken wurden auch von Wittenburg [342] untersucht. Haarslev, Möller und Schröder schildern, wie Beschreibungslogiken zur Spezifikation zum Parsing von visuellen Strukturen eingesetzt werden können [112].

47. Piktogramme würde ich in diesem Zusammenhang als statische visuelle Notation bezeichnen, die auch als spezielles Zeichenattribut behandelt werden kann.

Art von „Zeichenattribut“ aufgefaßt werden, das nur für bestimmte Konzepte verwendbar ist und auf die konkreten Objektattribute einer Instanz ausgerichtet wird (z.B. Anzahl der Trolleys sowie deren Einbauposition an der Küchenrückseite).

Um die Beziehungen zwischen Anwendungsdaten bzw. Objektattributen und deren Präsentation in einer visuellen Oberfläche zu beschreiben, haben Takahashi et al. ein detailliertes bidirektionales Modell vorgestellt, das Anwendungsdaten schrittweise in visuelle Strukturen überführt. Abbildung 29 gibt einen Überblick über die Architektur ihres Systems mit Namen TRIP2 ([312], [200]).

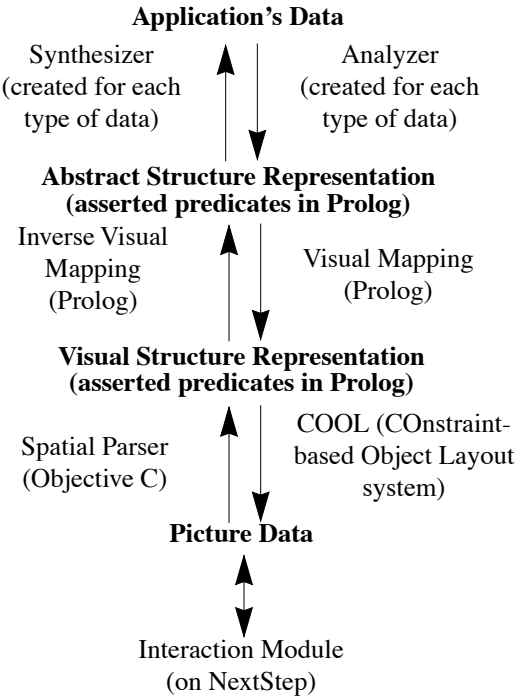


Abbildung 29. Skizze der Architektur des Systems TRIP2 (nach [312]).

Zur Gewährleistung der Unabhängigkeit der Systemarchitektur von verschiedenen Repräsentationsformalismen für Anwendungsdaten führt TRIP2 eine kanonische Zwischenrepräsentation ein (genannt ASR, abstract structure representation). Anwendungsdatenstrukturen werden in Prolog-Strukturen der ASR-Ebene transformiert und umgekehrt (siehe Abbildung 30 für ein Beispiel). Synthese- und Analysefunktionen müssen durch den Programmierer vorgegeben werden. Das TRIP2-System enthält Modelle, die geeignet sind, um Strukturen der ASR-Ebene in visuelle Strukturen abzubilden. Die Ebene der visuellen Strukturrepräsentation (VSR) verwendet ebenfalls Prolog-Strukturen, um die für die Generierung einer graphischen Figur notwendigen Informationen zu kodieren. Auf dieser Ebene werden qualitative Beziehungen verwendet, die anschließend in konkrete (quantitative) Koordinaten abgebildet werden müssen. Dieses wird durch ein Einschränkungs-basiertes Layoutsystem erreicht (genannt COOL, für eine detaillierte Einführung siehe [143]). Die produzierten Ausgaben können mit der Maus manipuliert werden. Durch die bidirektionalen Abbildungen von TRIP2 können Änderungen der graphischen Darstellung durch einen „räumliche Parser“ erkannt und in Strukturen der VSR-Repräsentation ausgedrückt werden usw. Abbildung 30 zeigt ein Beispiel, in dem eine „besteht-aus“-Relation auf der Ebene der Anwendungsdaten durch eine Graphstruktur ausge-

drückt wird, die auf der VSR-Ebene qualitativ repräsentiert ist und durch einen Render-Prozeß auf die Bildebene abgebildet wird.

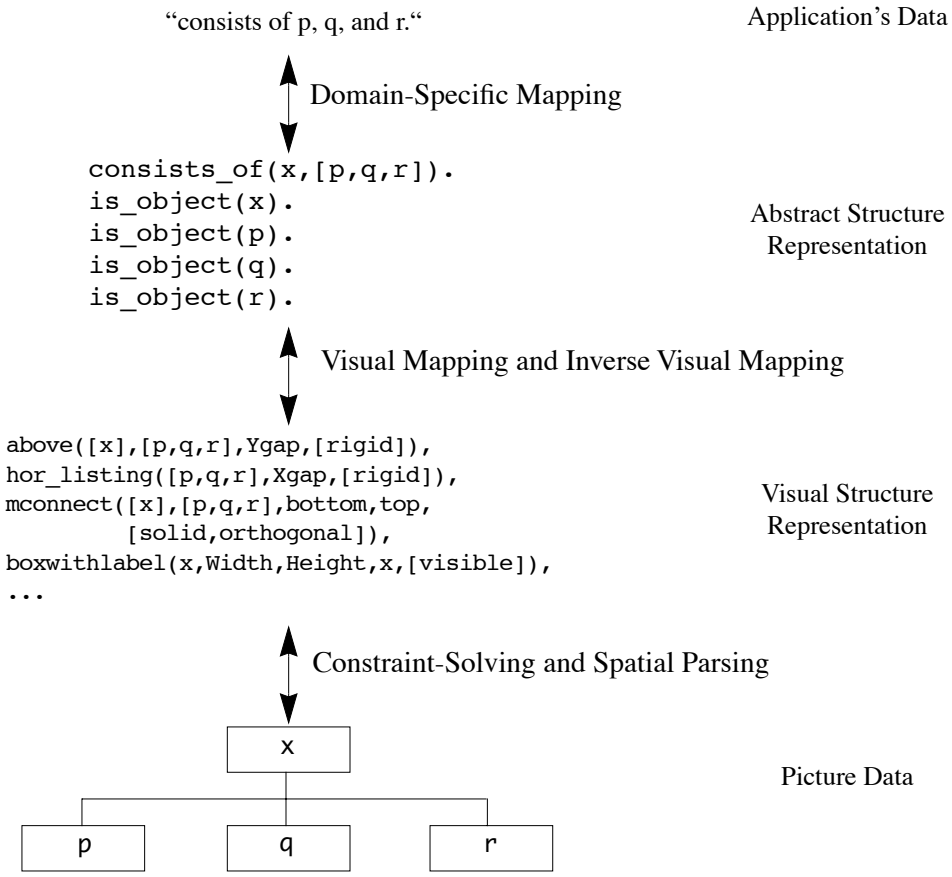


Abbildung 30. Beispiel zur Illustration der Repräsentation des TRIP2-Systems (nach [312]).

Der visuelle Abbildungsprozeß von ASR-Strukturen in VSR-Strukturen ist durch eine Menge von Abbildungsregeln definiert (`objectmap`, `inv_objectmap`, `relationmap`, `inv_relationmap`). Die Abbildungsregeln von TRIP2 sind in Prolog geschrieben. Neue Entwicklungen der TRIP-Systems (TRIP3) sehen die Berechnung von Abbildungsregeln aus Beispielen vor. Bei vorgegebenen Anwendungsdaten und entsprechenden Visualisierungen versucht TRIP3, generalisierte Abbildungsregeln durch Generalisierung der visuellen Beispiele zu „lernen“ [209].

Das TRIP-System erlaubt mit seinen Abbildungsregeln die Transformation von Anwendungsdaten in graphischen Präsentationen. Allerdings ist es unklar, wie unterschiedliche graphische Darstellungen kombiniert werden können. Weiterhin sieht die Konzeption von TRIP nicht die Betrachtung von Abbildungsalternativen mit einer Abwägung von Vor- und Nachteilen vor. Die Manipulationsmöglichkeiten für visuelle Strukturen gestatten nicht die Einbeziehung von semantischen Rückkopplungen (semantic feedback). Die gewählten visuellen Abbildungen und die erlaubten Editieroperationen werden nicht auf die Aufgaben des Benutzers abgestimmt. Ansätze zur Generierung von Visualisierungen und Präsentationen, die diese wichtigen Punkte berücksichtigen, werden in den nächsten Abschnitten betrachtet.

Visuelle Sprachen betrachten im allgemeinen nur die syntaktische Ebene von visuellen Notationen. Ähnlich wie bei natürlichsprachlichen Systemen, ist der kombinatorische Raum, der durch die Syntax definiert wird, noch zu groß. Semantische und pragmatische Kriterien müssen für die Zusammensetzung von Präsentationen ebenfalls in Betracht gezogen werden.

Geller und Shapiro [96] definieren den Begriff des tiefen graphischen Wissens („tief“ wird in Analogie zu den Tiefenstrukturen der Computerlinguistik benutzt). Eine Wissensbasis repräsentiert tiefes graphisches Wissen, wenn die Modelle zum einen deduktive Adäquatheit im Bereich der Graphik zeigen, d.h. die Wissensbasis modelliert Schlußfolgerungsprozesse über Formen und Positionen von Objekten, und zum anderen auch projektiv adäquat sind, also Bilder durch Inferenzprozesse tatsächlich erzeugt werden (und nicht schon vorliegen). Positionen von Objekten können als Einzelwerte gegeben oder mit Freiheitsgeraden versehen sein. Sie können auch relativ zueinander definiert werden und zwar mit impliziten oder expliziten Referenzsystemen. Weiterhin sollte das Wissensrepräsentationssystem Schlußfolgerungen über Teile, Gruppierungen (clusters) und Baugruppen unterstützen.

In ihrem Ansatz zum „intelligenten maschinellen Zeichnen“ (intelligent machine drafting) streichen Geller und Shapiro den Unterschied zwischen Objekten und ihren graphischen Präsentationen sowie den Unterschied zwischen Attributen von Objekten und Attributen der graphischen Darstellung heraus. Sie schlagen ein recht einfaches Abbildungsschema vor, um Domänenobjekte und ihre Attribute auf visuelle Notationen abzubilden. Mächtigere Beschreibungsformen für Abbildungen sind jedoch für viele Anwendungen notwendig. Die nächsten Abschnitte beschreiben verschiedene Ansätze zur Kommunikation von Informationen mit visuellen Notationsformen.

2.3.3 Automatisches Design von graphischen Präsentationen

Es wurde inzwischen ein Vielzahl von Ansätzen zur automatischen Generierung von graphischen Präsentationen veröffentlicht. Mackinlay unterscheidet zwischen den folgenden Schwerpunkten von existierenden Arbeiten ([189] p. 112f.): *Inhaltsaspekte* und *Aspekte des graphischen Designs* sowie *Aspekte der Designvariation* in einem Syntheseraum. Die bestehenden Arbeiten werden hier diskutiert und in diese Kategorien eingeordnet.

Eine der ersten Arbeiten zur Automatisierung der graphischen Informationspräsentation war die Arbeit von Zdybel et al. [347] (ca. 1980). Es wurde das AIPS (Advanced Information Presentation System) vorgestellt . Die Idee von AIPS war es, Domänenwissen explizit zu repräsentieren und Wissen über Kommunikationsmodalitäten (Tabellen, Karten, Graphen) sowie auch graphische Ausdruckselemente (Text, Rechtecke, Kartensymbole, usw.) durch KL-ONE-artige Strukturen zu modellieren. Der Kodierungsprozeß wird in AIPS mit Prozeduren implementiert, die an Instanzen dieser Konzepte „angeheftet“ werden. Wenn Eingabedaten verfügbar sind, werden diese Prozeduren automatisch ausgeführt und neue Objekte werden erzeugt. Daraufhin werden wiederum andere Prozeduren ausgeführt usw. Das VIEW-System von Friedell (ca. 1983) war ein Versuch, graphische Figuren, die in einer Frame-Repräsentation definiert waren (mit Eigenschaften und Teil-Ganzes-Beziehungen), algorithmisch zusammenzusetzen ([94] [95]). Allerdings ist die prozedural definierte Inferenzmaschine überaus schwer zu verstehen und zu bewerten, da die Wissensbasen keine deklarativen Repräsentationsformen verwenden (außer vielleicht Frame-Systeme zur Datenmodellierung).

Für spezielle Präsentationen wurden im Projekt „Integrated Interfaces“ deklarative Techniken entwikkelt (siehe [9], [10] und auch die Arbeiten zum CUBRICON-System in [231]). Beides, das Anwen-

dungsmodell und das Oberflächenmodell wird hier als KL-TWO-Wissensbasis formuliert. Zur Konstruktion von visuellen Präsentationen für Anwendungsobjekte werden Präsentationsregeln verwendet. Ein Beispiel wird in Abbildung 31 gezeigt.

```
1.  IF (Operational-Ship x) or (NonDeployed-Ship x)
        THEN (Coloration Image(x) Green)
2.  IF (Disabled-Ship x)
        THEN (Coloration Image(x) Red)
3.  IF (Ship x) and (Course x y)
        THEN (Orientation Image(x) y)
4.  IF (Ship x) and (Mission x y) and (Mobile y)
        THEN (Icon-Type Image(x) Arrow)
5.  IF (Ship x) and (Schedule x y)
        THEN (Tag Image(x) Textual-Description(y))
```

Abbildung 31. Einige Präsentationsregeln aus dem Projekt „Integrated Interfaces (aus [9], p. 810).

Ein interessanter Punkt dieser Arbeit ist, daß spezielle räumliche Situationen (z.B. eine Konstellation von Schiffen, die in einer Pazifikregion verteilt sind) speziell zur Visualisierungsgestaltung erkannt werden. Auch hierfür werden Regeln verwendet. Unglücklicherweise müssen bei diesem Ansatz die Präsentationsregeln für jede Anwendungsdomäne neu erstellt werden, da die Abbildungen nicht für generische Konzepte definiert sind. Mit HAMVIS verfolge ich daher den Weg, zumindest für eine Klasse von Anwendungen ein gemeinsames Grundmodell bereitzustellen, um die Repräsentation von domänenübergreifendem Wissen zu ermöglichen. Weiterhin werden in dem frühen Ansatz von Arens et al. Einschränkungen von graphischen Attributen (bzw. Abhängigkeiten zwischen ihnen) noch nicht repräsentiert, d.h. die Regeln können nicht auf Kohärenz überprüft werden (wird ein Objekt auf schwarzem Hintergrund schwarz dargestellt?).

APEX

Das System APEX (Automated Pictorial EXplanations) wurde von Feiner entwickelt ([81], [82]) und generiert Präsentationen mit physikalischen Objekten. Zur Darstellung von durchzuführenden Handlungen werden auch Folgen von Bildern generiert. Die Reihenfolge der von APEX generierten Bilder ist vorbestimmt durch die Reihenfolge der Unteraktionen einer zusammengesetzten Handlung, d.h. diesbezüglich gibt es keinen Variantenraum. Die Handlungen, die der Betrachter durchführen soll, betreffen Wartungs- oder Reparaturarbeiten an militärischen Geräten. Der Schwerpunkt der Arbeit liegt hier in der automatischen Inhaltsbestimmung für ein bestimmtes Bild zur Beschreibung einer Teilhandlung. Zur Bestimmung der in ein Bild einzufügenden Objekte, zusammen mit dem Darstellungsstil sowie dem notwendigen Detaillierungsgrad, verwendet APEX Regeln. APEX verwaltet auch ein einfaches Modell des Betrachters. Dieses Modell speichert die Menge von Objekten, von denen angenommen wird, daß sie ihm bekannt sind. Objekte, die in einem Bild gezeigt werden, können die folgenden kommunikativen Funktionen übernehmen:

- Frame-Objekte

  Die Aktionenmodelle von APEX definieren die Objekte, die durch die Aktionen direkt betroffen sind. Diese Hauptobjekte werden Frame-Objekte genannt.

- Kontextobjekte

Ausgehend von den Frame-Objekten durchläuft APEX die Teil-Ganzes-Hierarchie und fügt Objekte zur Darstellung hinzu, bis ein Objekt gefunden wird, das dem Betrachter laut Benutzermodell bekannt ist.

- Landmarkenobjekte

  APEX versucht, Landmarkenobjekte zu bestimmen, um eine Referenz für die Lokation der bisher in die Darstellung übernommen Objekte (Frame-Objekte und Kontextobjekte) angeben zu können. Hierzu betrachtet APEX die Umgebung von Frame-Objekten und bestimmt diejenigen Objekte, die sich in Form, Größe oder Material wesentlich von den Frame-Objekten unterscheiden.

- Ähnliche Objekte

  Ein weiterer Schritt bei der Inhaltszusammenstellung besteht bei APEX in der Suche nach Objekten, die ähnlich zu denen sind, die schon in der Darstellung auftreten. Die Begründung hierfür ist, daß es dadurch vermieden werden kann, daß der Betrachter die gezeigten Objekte mit den ähnlichen Objekten verwechseln könnte.

- Stützende Objekte

  In einem nächsten Schritt fügt APEX noch zusätzliche Objekte hinzu, die z.B. sicherstellen, daß die bis dahin gezeigten Objekte nicht in der Luft schweben.

- Meta-Objekte

  Objekte, die eine Aktion visualisieren (und damit die Objekte markieren, die durch die Aktion beeinflußt werden) werden als metagraphische Objekte als letztes hinzugefügt. Im Prototypsystem wurden z.B. Pfeile verwendet.

Durch jede dieser Mengen werden den jeweils enthaltenen Objekten verschiedene kommunikative Rollen oder Funktionen zugewiesen. Neben der Zusammenstellung der darzustellenden Objekte werden also noch Eigenschaften definiert, die die Darstellungsform im Bild beeinflussen. Folgende „Darstellungsparameter“ sind betroffen.

- Kamera

  Außer für Stützobjekte, die auch teilweise unsichtbar oder verdeckt sein können, sichert APEX die volle Sichtbarkeit aller Objekte der anderen Mengen zu und berechnet passende Kameraparameter.

- Techniken zum Rendering

  APEX unterstützt zwei Rendering-Stile: normal und gedämpft. Der normale Rendering-Stil wird für Frame-Objekte verwendet, d.h. Objekte, die durch Aktionen modifiziert werden. Der gedämpfte Stil wird für alle weiteren Darstellungsobjekte verwendet, um anzudeuten, daß diese Objekte weniger wichtig sind.

- Detaillierungsgrad

  In der Teil-Ganzes-Hierarchie von APEX entsprechen nur die Blätter tatsächlichen physikalischen Objekten mit Eigenschaften wie Größe, Form und Material. Falls ein Objekt, das in der Zerlegungshierarchie nicht als Blattknoten vorkommt, in die Graphik aufgenommen werden soll, traversiert APEX die Zerlegungshierarchie und bestimmt, welche der Blätter im Bild erscheinen sollen. Das System ist so konzipiert, daß alle für die Aktionen notwendigen Teile zur Graphik hinzugefügt werden. Weiterhin werden diejenigen Objekte hinzugefügt, die erforderlich sind, um Verwechslungen mit ähnlichen Objekten auszuschließen. Obwohl nicht ganz klar ist, welche Maße hier konkret verwendet werden, scheint die Idee einleuchtend, bei ähnlichen Objekten lieber alle zu präsentieren. Der Betrachter hat dann noch die Zusatzinformation über relative Positionen, so daß die Verwechslungsgefahr herabgesetzt werden kann.

Wie APEX betrachtet auch HAMVIS physikalische Objekte. Wie schon in der Einleitung diskutiert, unterstützt HAMVIS jedoch direktmanipulative Aktionen mit graphischen Objekten (und nicht Aktionen mit den Referenten in der realen Welt im Sinne einer Bedienungsanleitung). Im HAMVIS-Kontext ist die Unterscheidung von Entwurfszeit und Laufzeit (bzw. Benutzungszeit) wichtig. Berechnungen zur Sichtbarkeit von Objekten können zur Entwurfszeit nicht in gleichem Umfang durchgeführt werden, wie bei APEX, da konkrete Objekte erst zur Laufzeit als Instanzen berechnet werden. Wissen über die Sichtbarkeit bzw. Überlappungsfreiheit muß also auf konzeptueller Ebene (deklarativ) repräsentiert werden und kann nicht in jedem Fall über geometrische Berechnungsvorgänge abgeleitet werden.

Wie in APEX sind für die Bestimmung der Umgebung von Objekten und für die jeweiligen Referenzsysteme auch im HAMVIS-Kontext Teil-Ganzes-Zerlegungen notwendig. Wegen der Verwendung der Zerlegungshierarchien zur Entwicklungszeit sind wiederum nur konzeptuelle Informationen verfügbar (siehe Kapitel 3.1).

Vom Grundansatz her verwendet APEX Techniken der realitätsangenäherten 3D-Computergraphik, repräsentiert jedoch zusätzlich noch den rhetorischen Status eines Objektes. Der Status eines Objekts wird definiert durch den Grund, warum ein Objekt in die Darstellung aufgenommen wurde (Mitgliedschaft in einer der oben aufgeführten Mengen). APEX macht jedoch wenig Gebrauch von graphischen Techniken, um Informationen über den rhetorischen Status von Objekten zu vermitteln. Zur Hervorhebung der durch die Aktionen manipulierten (Haupt-)Objekte werden die anderen Objekte gedämpft dargestellt. Andere Techniken werden nicht in Betracht gezogen.

In APEX wird noch kein Variantenraum für die Komposition von Darstellungen verwaltet. Im Gegensatz zu APEX ist es für HAMVIS wichtig, daß die generierten Visualisierungen mehrere Aktionsarten gleichzeitig unterstützen. Es muß für alle Aktionsarten ein „gemeinsamer Nenner“ gefunden werden (hierbei spielt wiederum das in einer Darstellung verwendete Modell ein Rolle, s.u.).

Die Betrachtung verschiedener Varianten führt auf ein anderes Thema der automatischen Präsentationsgenerierung: Aspekte des graphischen Designs und insbesondere Aspekte der Designvariation.

APT

Das Ziel von Mackinlays APT-System war es, die Generierung von zweidimensionalen statischen Präsentationen zu automatisieren [188] [189] [190].[48] Er betrachtete als Ausdrucksmittel folgende visuelle Sprachen:

- horizontale und vertikale Achsen,
- Punktwolken-Darstellungen (scatter plots),
- (geographische) Karten und
- Graphen.

Zur Spezifikation von graphischen Notationen verwendet APT formale Definitionen, die syntaktische und semantische Eigenschaften von graphischen Präsentationen mit prädikatenlogischen Formeln beschreiben.

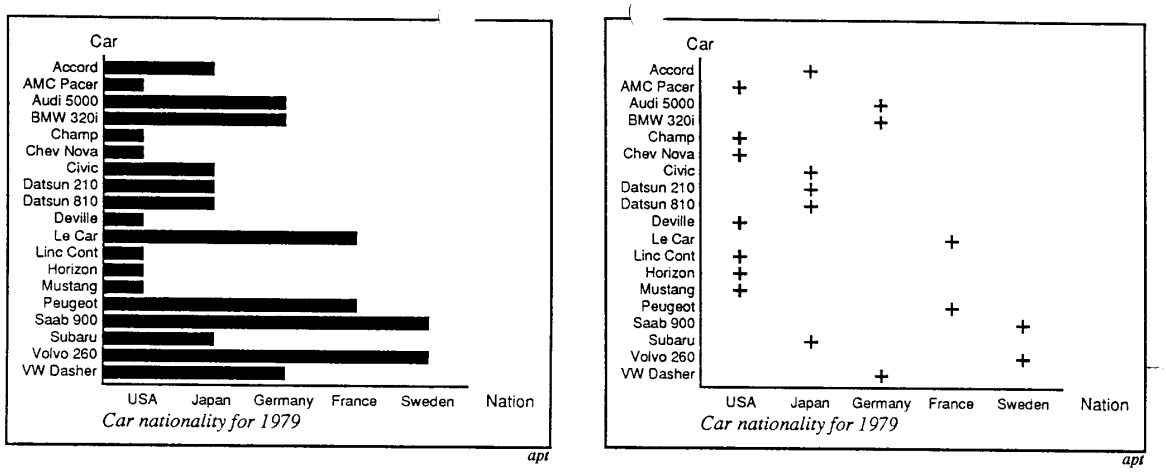


Abbildung 32. Zwei Möglichkeiten, eine Nation-Auto-Relation zu präsentieren (aus [190], siehe Text).

Mackinlay unterscheidet zwischen zwei Auswahlkriterien für visuelle Notationen: *Ausdrucksfähigkeit* und *Ausdruckseffektivität*. Bestimmte graphische Notationen sind dazu geeignet, jeweils spezielle Relationen zu kommunizieren. Nicht alle Relationen lassen sich jedoch mit allen Sprachen ausdrükken. Mackinlay definiert, daß eine Relation in einer visuellen Notation *ausdrückbar* ist, falls ein Element einer Relation (ein Faktum) in der graphischen Notation kodiert werden kann, und ein Tupel, das nicht in einer Relation enthalten ist, durch diese Notation nicht kodiert wird. Implizite, ungewollte Aussagen dürfen demnach nicht durch eine Graphik vermittelt werden. In dem linken Teil aus Abbildung 32 wird ein Balkendiagramm zur Präsentation von Informationen über eine Nation-Automarke-Relation verwendet. Bei dieser Darstellungsart wird implizit durch die Länge der Balken eine Ordnung entlang der vertikalen Achse ausgedrückt (vgl. die Arbeiten von Bertin). In diesem Falle

48. Eine frühe Arbeit im Bereich des Designs von Geschäftsgraphiken war das BHARAT-System [98].

erscheinen beispielsweise schwedische Autos besser als andere. Dieses ist hier nicht beabsichtigt und daher ist die Darstellung auf der rechten Seite von Abbildung 32 eine bessere Wahl. Implizite Ordnungsinformationen werden hier nicht kommuniziert.

Als Beispiel für Ausdruckskriterien und implizite Informationen diskutieren Mackinlay und Genesereth ein weiteres Beispiel, das hier als Abbildung 33 wiedergegeben ist.

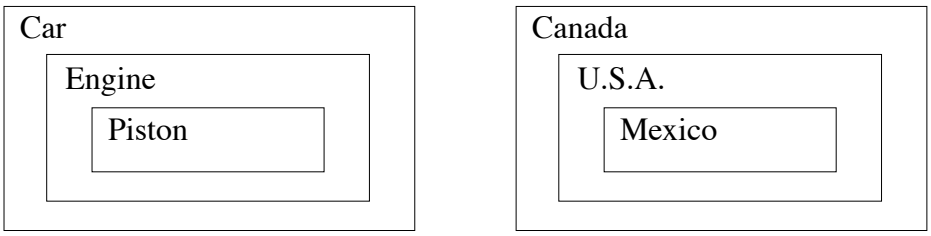


Abbildung 33. Richtige Verwendung der Geschachtelte-Boxen-Notation für eine Teil-von-Relation und fehlerhafte Verwendung für eine (transitive) In-der-Nähe-von-Relation (nach [188] S. 2ff).

Die Geschachtelte-Boxen-Notation kann zur Präsentation von Elementen der transitiven Teil-von-Relation verwendet werden, verursacht aber falsche Informationen für die nicht-transitive In-der-Nähe-von-Relation.

Auch wenn kein Auto-Kolben-Tupel explizit in der Teil-von-Relation eingetragen ist, kann die zusätzliche Information, die durch die Geschachtelte-Boxen-Notation vermittelt wird, toleriert werden, da die Domänenrelation und die visuelle Notation transitiv sind. Ein Syntheseverfahren für Präsentation muß daher Eigenschaften von Relationen auf der Domänenseite und Eigenschaften von visuellen Notationen auf der Graphikseite betrachten. Die Aufgabe besteht darin, eine Abbildungsfunktion zu finden, die ungewollte graphische Implikationen vermeidet.

Um nun die Auswahlkriterien bei mehreren Kandidaten für visuelle Notationen zu formalisieren, hat Mackinlay den Begriff der *Ausdruckseffektivität* zusammen mit dem Begriff der *Ordnung nach Wichtigkeit* eingeführt. Wichtige Informationen sollten mit effektiveren Techniken kommuniziert werden. Im Gegensatz zur Ausdruckfähigkeit, hängt die Ausdruckseffektivität vom perzeptuellen System des Menschen ab. Unter Bezugnahme auf Cleveland und McGill, die in ihren Untersuchungen belegen konnten, daß Menschen verschiedene perzeptuelle Aufgaben mit unterschiedlicher Genauigkeit ausführen können [50] [51], verwendet Mackinlay eine Genauigkeitsrangfolge für quantitative perzeptuelle Aufgaben wie z.B. die (relative) Bestimmung von Positionen (siehe Abbildung 34 auf der nächsten Seite).

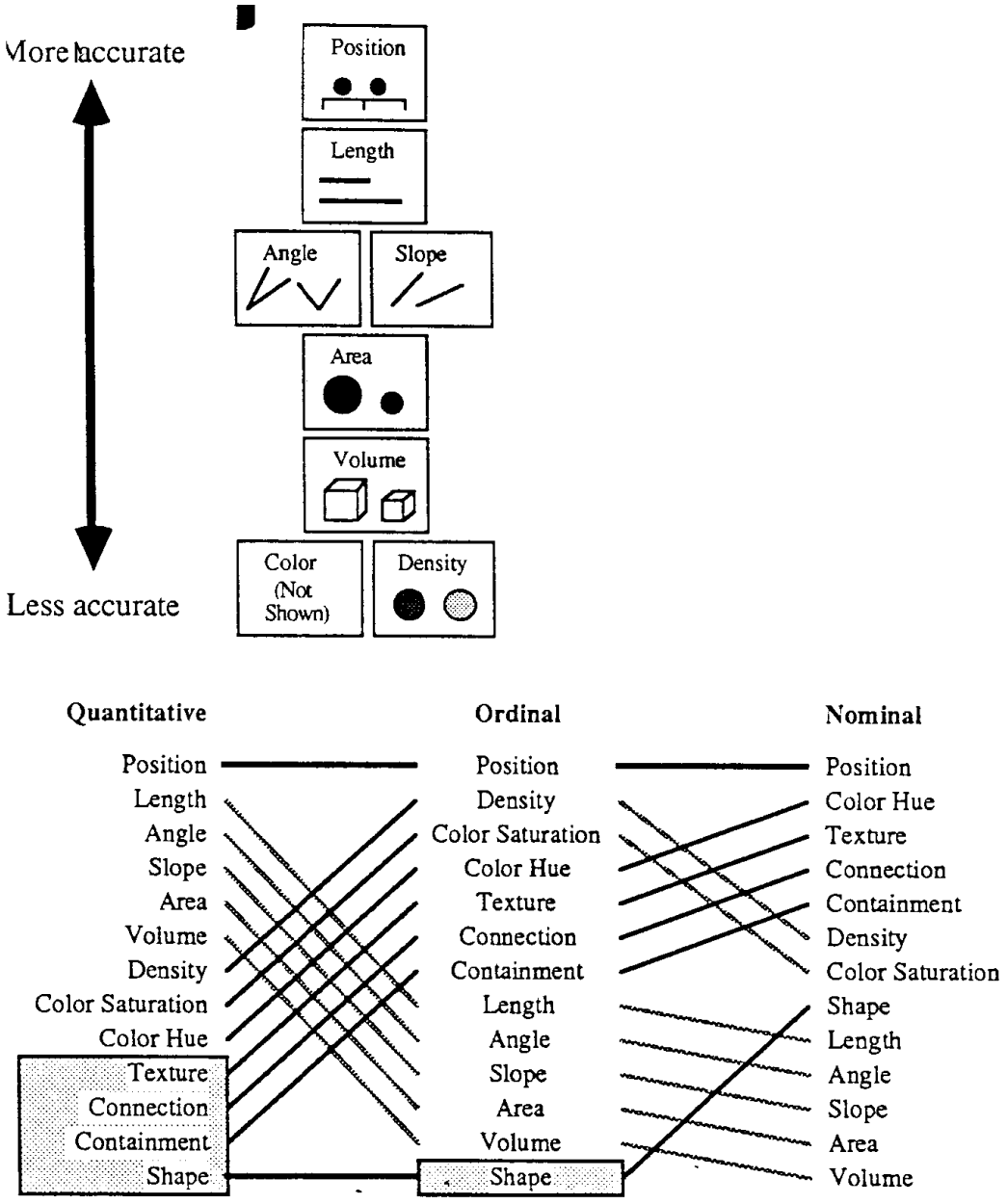


Abbildung 34. Rangordnung für die Effektivität von Perzeptionsaufgaben (Aufgaben in grauen Boxen sind für den betreffenden Datentyp nicht relevant (aus [189] S. 125).

Die obere Darstellung vermittelt eine Abschätzung der Genauigkeit von unterschiedlichen perzeptuellen Aufgaben für quantitative Daten (kontinuierlicher Wertebereich, z.B. das Intervall [0, 17]). Der untere Teil zeigt eine Rangordnung für ordinale (z.B. Aufzählungen wie <Montag, Dienstag, ...> und nominale Domänen (Mengen ungeordneter Elemente wie z.B. {Kreis, Rechteck, Dreieck, ...}).

Der Synthesealgorithmus von APT arbeitet in drei Schritten:

- Partitionierung

  Die Menge der darzustellenden Relationen wird so in Partitionen zerlegt, daß für jede Partition eine primitive graphische Sprache existiert, deren Ausdrucksfähigkeitskriterien durch jede der Relationen der Partition erfüllt wird. Bei der Partitionierung wird die wichtigste Relation zuerst betrachtet und so weiter. Nach diesem Schritt gibt es für jede Partition eine Liste von visuellen Notationen als Kandidaten.

- Auswahl

  Die Menge von Kandidaten für eine Partition wird nach den Effektivitätskriterien geordnet.

- Komposition

  Nachdem die Kandidaten geordnet worden sind, versucht APT, die Sprachen von verschiedenen Partitionen zu kombinieren. Abbildung 35 zeigt ein Beispiel, in dem zwei individuelle Präsentationen für unterschiedliche Relationen sehr effektiv kombiniert werden können.

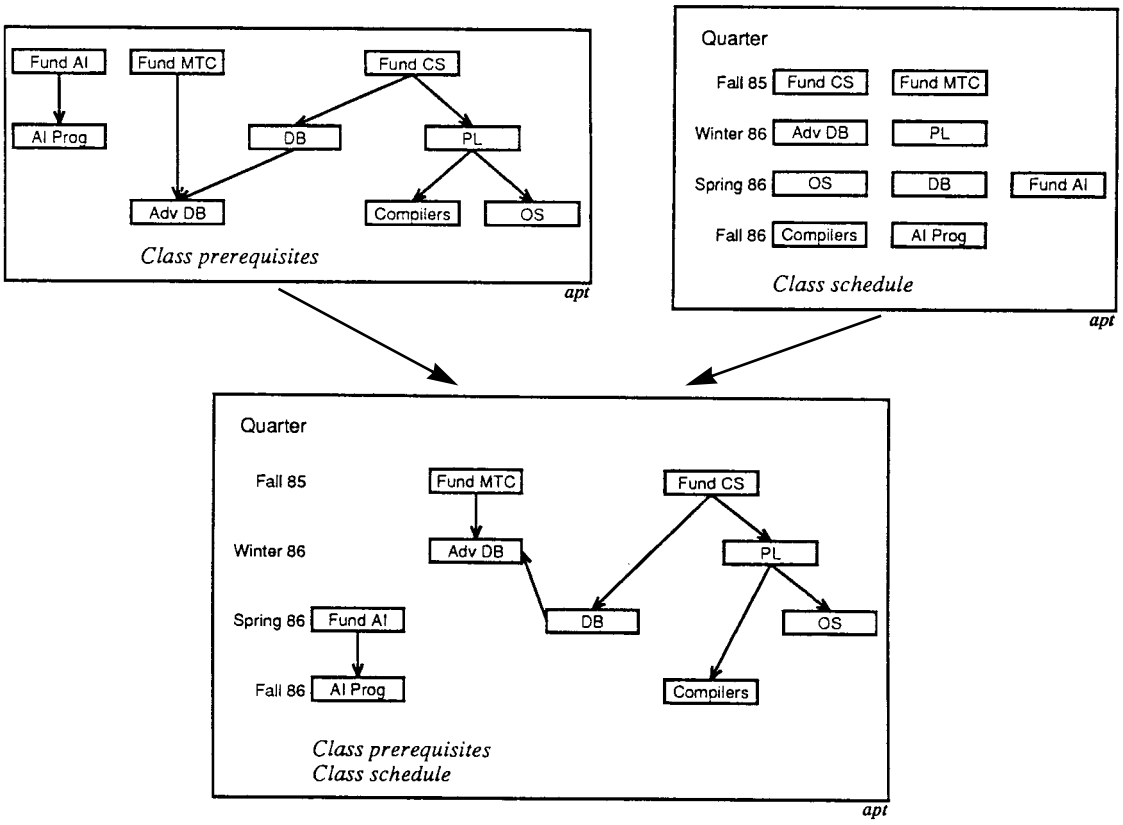


Abbildung 35. Komposition zweier graphischer Präsentationssprachen (aus [190] S. 128).

Ein anderes Kompositionsbeispiel ist die Überlagerung zweier Diagramme. Diese Kombinationsform ist anwendbar, wenn zwei Achsen die gleichen Domänenmengen kodieren. Mackinlay definiert drei Kompositionsoperatoren (mit einer groben Effektivitätsanordnung) und gibt Anwendbarkeitsbedin-

gungen an. Fragen nach der Größe von Darstellungen (graphische Elemente nicht zu klein dargestellt werden) werden von APT auf der Rendering-Ebene behandelt.[49]

In jedem der drei Schritte des Synthesealgorithmus gibt es Wahlmöglichkeiten, die jeweils in Sackgassen führen können. Falls dieses passiert, verwendet APT chronologisches Wiederaufsetzen, um andere Alternativen zu finden.[50] Wiederaufsetzpunkte sind gegeben durch andere Partitionierungen, durch Verwendung einer weniger effektiven graphischen Sprache für eine (weniger wichtige) Relation und weiterhin durch verschiedene Kompositionstechniken.

In beschränktem Maße berücksichtigt APT auch Aspekte des zu verwendenden Mediums. Falls ein Farbbildschirm als Ausgabemedium zur Verfügung steht, generiert APT während der Partitionierung und Auswahlphase weitere Alternativen, die wiederum in zusätzlichen Kompositionsmöglichkeiten resultieren.

APT hat ein Rahmenwerk für automatische Präsentationsgenerierungssysteme definiert, und im Gegensatz zu vagen Ratschlägen oder Richtlinien stellt APT einen wichtigen Versuch dar, graphisches Designwissen formal zu repräsentieren. In APT wird eine Komposition von Präsentationen angestrebt. Dieses wurde auch als wichtiger Schritt für die Generierung von Visualisierungen für Benutzungsschnittstellen erkannt. Der Konstruktionsraum (Variantenraum) für graphische Präsentationen wird mit Hilfe von Techniken der logischen Programmierung aufgespannt. Allerdings präsentiert APT eine Menge von vorgegebenen Daten nur aufgrund von Datencharakteristiken, d.h. die Darstellungen werden nicht auf eine damit zu bearbeitende Aufgabe abgestimmt. Als Einschränkungen für den Konstruktionsraum gelten nur die Eigenschaften der Daten und deren Relationen sowie die Eigenschaften der durch APT bereitgestellten visuellen Notationen. Gleiches gilt auch für die Komposition von Präsentationen.

BOZ

Ein Nachfolger von APT ist das von Casner entwickelte System BOZ [39] [40]. BOZ generiert ebenfalls graphische Darstellungen für relationale Daten, stimmt aber die Präsentationen auf die Erfordernisse einer vorgegebenen Aufgabe ab:

> *„Since the usefulness of a graphic presentation is a function of the task that the graphic will be used to support, graphic design should focus on designing efficient perceptual procedures to be performed by human users. Decisions made about how to encode and structure information in an accompanying graphic should be based primarily on supporting efficient and accurate performance of the perceptual procedure.“ ([39], S. 115)*

Inspiriert durch Forschungsergebnisse aus den Kognitionswissenschaften und insbesondere durch die Arbeiten von Larkin und Simon [171], unterstützt Casner die menschlichen Fähigkeiten durch:

---

49. Falls ein Objekt zu klein ist, kann z.B. Form und Farbe nur schwer wahrgenommen werden (vgl. [190] S. 133).

50. APT wurde auf einer Symbolics Lisp Machine mit MRS, einem logischen Programmiersystem, implementiert ([190] p. 136).

1. Ersetzung „langsamer“ logischer Inferenzen durch „schnelle“ perzeptuelle Inferenzen, wobei davon ausgegangen wird, daß logische Inferenzen außerdem einen höheren geistigen Aufwand (mental effort) erfordern.[51]
2. Reduktion der Suche nach aufgabenrelevanter Information.

Eingaben für BOZ sind:

- eine sogenannte „logische Prozedur“ (mit Kontrollstrukturen wie `if`, `while`, `repeat`),
- die Charakterisierungen der Domänenmenge: Name, Art (quantitativ, ordinal, nominal)
- der Wertebereich der Domänenmengen.

Die Operationen (statements) in einer logischen Prozedur sind als sog. logische Operatoren gegeben. Casner unterscheidet zwischen Suchoperatoren (`ASK`, `TELL`, `RETRACT`) und Berechnungsoperatoren (`+`, `-`, `*`, `/`, `AND`, `OR`, `NOT`). Um einen Eindruck vom Modellierungsstil in BOZ zu vermitteln, betrachte ich einige seiner Beispiele.

In einem Beispiel zur Präsentation von Daten im Rahmen einer Flugreservierung werden die folgenden logischen Operatoren definiert:[52]

```
(LOP determineDeparture (<flight> <DEPARTURE>)
   (ASK (Departure <flight> <DEPARTURE>))

(LOP computeLayover (<departure> <arrival> <LAYOVER>)
   (- <departure> <arrival> <LAYOVER>))
```

Mit diesen Deklarationen werden zwei logische Operatoren `determineDeparture` und `computeLayover` definiert. Der erste Operator bekommt als Parameter einen Flug und liefert eine Abflugzeit. Die Abflugzeit wird direkt aus der Menge der Attribute des Fluges erfragt. Für diesen elementaren Zugriff wird ein bestimmter Aufwand zugrundgelegt. Bei dem zweiten Operator werden zwei Parameter zur Beschreibung der Abflug- und der Ankunftszeit übergeben. Es wird eine Überlappungsdauer geliefert. Die Operation `computeLayover` wird auf eine Subtraktion zurückgeführt. Mit der elementaren Subtraktion ist wiederum ein bestimmter mentaler Aufwand assoziiert.

Die Eingabe für BOZ besteht aus einer Datenbasis von logischen Fakten. Das Repertoire zur graphischen Kodierung dieser Informationen ist eine Menge von primitiven graphischen Notationen (z.B. horizontale Position, vertikale Position, Höhe, Breite, Linienlänge, Form, Fläche, Schattierung, Verbundenheit, Farbe, Beschriftungen, Liniendicke, usw.). Für jede dieser „Sprachen“ gibt es eine Menge von perzeptuellen Operatoren, die graphische Attribute ermitteln oder Beziehungen zwischen graphischen Attributen berechnen, z.B.:

```
(POP search-object-with-shade (<OBJECT> <shade>)
   (ASK (Shading <OBJECT>) <shade>))

(POP determine-horz-distance (<horzpos1> <horzpos2> <DISTANCE>)
   (DIFFERENCE <horzpos1> <horzpos2> <DISTANCE>))
```

51. Siehe die Diskussion über GOMS und CCT als Vorschläge für ein Maß für die Schätzung der mentalen Anstrengung.

52. Variablen sind in spitzen Klammern gedruckt. Ausgabevariablen sind in Großbuchstaben gesetzt.

Verschiedene perzeptuelle Operatoren (aus verschiedenen primitiven graphischen Sprachen) berechnen die gleiche Relation (`ASK`, `DIFFERENCE`, ...), realisieren also den gleichen logischen Operator.

Die Hauptidee des Synthesealgorithmus von BOZ besteht darin, logische Operatoren durch perzeptuelle Operatoren zu substituieren, z.B. kann `determine-horz-distance` für „–“ eingesetzt werden. Allerdings ist `determine-horz-distance` nicht die einzige Möglichkeit. Auf den ersten Blick mögen einige Alternativen besser erscheinen als andere, klar ist jedoch, daß eine bestimmte Wahl eines perzeptuellen Operators in eine Sackgasse führen kann, weil z.B. zwei perzeptuelle Operatoren nicht effektiv kombiniert werden können. BOZ muß also in ähnlicher Weise wie APT einen Konstruktions- bzw. Suchraum verwalten.

Bevor ein perzeptueller Operator tatsächlich aus der Menge der Möglichkeiten ausgewählt und als Substitution für einen logischen Operator verwendet wird, betrachtet BOZ die Informationen, die durch jeden logischen Operator manipuliert werden. BOZ bestimmt dadurch, wie Informationen kombiniert werden können, die von verschiedenen Operatoren referenziert werden. BOZ verwendet hierzu einen Partitionierungsansatz („perzeptuelle Datenstrukturierung“), der dem von APT ähnelt. Weiterhin setzt BOZ ein Rangordnungsschema für visuelle Notationen ein, das an das Schema von APT angelehnt ist.

Aus der Menge der möglichen perzeptuellen Operatoren wird eine Teilmenge ausgewählt, deren Elemente geeignet sind, die Menge von logischen Operatoren einer Partition zu ersetzen. Das Ergebnis der Substitutionsphase ist eine perzeptuelle Prozedur, die die logische Prozedur aus der Eingabe ersetzt. Die Komposition der perzeptuellen Operatoren bestimmt, wie die Darstellungskomponente von BOZ logische Fakten aus der Eingabedatenbasis in graphische Fakten übersetzt. Graphische Fakten sind durch die graphischen Objekte und ihre Attribute definiert.

BOZ unterstützt auch die Manipulation von graphischen Objekten. Änderungen der graphischen Fakten werden in Änderungen von logischen Fakten übersetzt. Die Interaktionsformen sind durch die primitiven graphischen Notationen definiert, die durch die perzeptuellen Operatoren bestimmt wurden.

APT und BOZ sind die ersten Systeme, die demonstrieren, daß graphische Präsentationen algorithmisch zusammengesetzt werden können.[53] BOZ fügt zu APT den Aspekt der Ausrichtung der Präsentationen auf bestimmte Aufgaben des Benutzers hinzu. Aufgaben werden in BOZ in Form von logischen Prozeduren und logischen Operatoren repräsentiert. Sie schränken den Kompositionsraum für primitive graphische Notationen weiter ein. Casner erwähnt folgende Aspekte, die von BOZ nicht behandelt werden:

- Domänenspezifische Darstellungskonventionen können nicht auf einfache Weise repräsentiert werden (siehe [321] für ein Beispiel).
- Räumliche Informationen wie z.B. bestimmte Formen oder räumliche Konstellationen, die aus Sicht von BOZ nicht essentiell für die Übermittlung der Informationen sind, aber viele Merkmale der Realweltobjekte widerspiegeln, werden nicht repräsentiert und können nicht in den Synthese-

53. Weitere Arbeiten zum Design und zur Komposition von Geschäftsgraphiken wurden im TASSO-Projekt durchgeführt ([66], [67], [158], [159]). TASSO verwendet Techniken des nicht-monotonen Schließens, um Diagramme aus Standardnotationen (Balkendiagrammen, Kuchendiagramme, usw.) zusammenzusetzen. Da die Generierung von Geschäftsgraphiken nicht der Schwerpunkt von HAMVIS ist, möchte ich hier nicht ins Detail gehen.

algorithmus eingehen. Casner erwähnt ein Beispiel, in dem Übersichten über die Sitzverteilung in Flugzeugen präsentiert werden sollen. In diesem Beispiel ist es vorteilhaft, eine räumliche Skizze der Sitze mit ihren Positionen im Flugzeugrumpf zu verwenden (zumindest sollte die Skizze topologisch mit der Realwelt übereinstimmen).

Nach meiner Einschätzung ist die Definition von Aufgaben mit logischen Prozeduren und logischen Operatoren ein wenig künstlich. Die Definitionen sind mehr oder weniger direkt auf den Ersetzungsmechanismus von logischen Operatoren durch perzeptuelle Operatoren ausgelegt. Weiterhin sind logische Operatoren nicht auf interaktive, direktmanipulative Aktionen des Benutzers ausgelegt. Interaktionsmöglichkeiten in BOZ sind ein Nebenprodukt der Eins-zu-eins-Abbildung von logischen Fakten auf graphische Fakten. BOZ unterstützt keine semantischen Rückkopplungen für interaktive Manipulationen.

SAGE

In dem automatischen Präsentationsgenerierungssystem SAGE (System for Automatic and Graphical Explanation, Roth et al. [268]) wird die Aufgabe, die eine Präsentation unterstützen soll, in Form von Informationsverarbeitungszielen definiert. SAGE bildet quantitative und relationale Daten (und ihre Charakteristiken [267]) auf graphische Notationen ab (Diagramme, Netzwerke, Karten, Tabellen, etc.). SAGE unterstützt dabei folgende Arten von Informationsverarbeitungszielen: Ermittlung von konkreten Werten, Vergleich von Werten (innerhalb einer Relation oder zwischen mehreren Relationen, paarweise, n-tupelweise), Verteilungen von Werten für eine Relation, Korrelationen zwischen Attributen usw.

Die Generierung einer Präsentation durch SAGE ist in folgende Schritte gegliedert:

- Auswahlphase

  Unter Betrachtung von Datencharakteristiken wird eine Kandidatenmenge von anwendbaren Präsentationstechniken generiert, die bestimmte Ausdrucksfähigkeitskriterien erfüllt. Anschließend werden die Elemente dieser Kandidatenmenge nach Ausdruckseffektivitätskriterien angeordnet. Die Ausdruckseffektivitätskriterien beziehen sich darauf, wie gut die Informationsverarbeitungsziele erfüllt werden. Im Gegensatz zu APT basieren die Ordnungskriterien nicht nur auf Datencharakteristiken (die durch das Datenmodell definiert sind), sondern beziehen sich auch auf die Informationsverarbeitungsziele[54].

- Verfeinerung der Darstellungskandidaten

  Im nächsten Schritt werden die Darstellungstechniken, die in der Auswahlphase als Kandidaten bestimmt worden sind, gefiltert und verfeinert. In Abhängigkeit von den Informationsverarbeitungszielen werden Details wie Achsen, Beschriftungen usw. zu jedem Kandidaten hinzugefügt. SAGE bestimmt dabei z.B., ob die Beschriftungen an den Achsen sortiert werden sollen (ordinale Domänen) oder nicht (nominale Domänen). Roth et al. stellen heraus, daß neue Darstellungstechniken zu SAGE hinzugefügt werden können, ohne daß die hier geschilderten Techniken geändert werden müssen ([268] p. 93), obwohl nicht klar dargestellt wird, wie dieses erreicht werden kann.

54. Für weitere Informationen siehe [268] S. 92.

- Synthesephase

  SAGE versucht, die Darstellungen für verschiedene Relationen in eine einzige Präsentation zu integrieren. Hierzu wird eine Menge von Syntheseregeln verwendet. Die Zusammensetzung von Komponenten zweier Basispräsentationen darf jedoch nicht den zuvor aufgestellten Einschränkungen widersprechen. Zwei Teildarstellungen können zusammengesetzt werden, wenn sie die gleiche kommunikative Funktion bzgl. mindestens einer gemeinsamen Datenmenge erfüllen (z.B. Kodieren von gleichen Mengen, Ausdrücken einer Korrespondenz zwischen Elementen von ein oder mehreren gleichen Mengen, usw.).

  Zu beachten ist, daß die gleichen kommunikativen Funktionen durch verschiedene graphische Primitive erfüllt werden können. SAGE betrachtet nur solche zusammengesetzten Darstellungen, deren Konstituenten die kommunikativen Funktionen erfüllen, die für die Informationsverarbeitungsziele benötigt werden. Roth et al. betonen, daß die Kompositionen nicht willkürlich vorgenommen werden, sondern auf die Informationsverarbeitungsziele abgestimmt werden (vgl. das Beispiel in Abbildung 35).

In jüngster Zeit wurde SAGE um die Möglichkeit erweitert, weitere Designeinschränkungen (Designdirektive) zu berücksichtigen, die von einem Benutzer mit Hilfe einer interaktiven Oberfläche vorgegeben werden [270]. Die Idee ist, partielle Kodierungshinweise, die der Benutzer gibt, durch SAGE vervollständigen zu lassen. Kodierungseinschränkungen können auf der Auswahlebene und auf der Verfeinerungsebene gegeben werden. Eigenschaften von graphischen Objekten (Linien, Marker, etc.), die laut Designdirektiven zu verwenden sind, können auf Eigenschaften von Domänenobjekten abgebildet werden. Vervollständigungen dieser Vorgaben aus den Designdirektiven werden von SAGE aufgrund der oben angesprochenen Datencharakteristiken bestimmt. Zur Vermeidung einer abstrakten Spezifikationssprache sieht SAGE vor, Skizzen des Benutzers zu interpretieren. Zur Anfertigung von Skizzen steht ein Zeicheneditor zur Verfügung, der es gestattet, Abbildungen durch direktmanipulative Interaktionsformen zu definieren. Vorher generierte Graphiken können hierzu mit speziellen Anzeigeprogrammen (Browser) durchgesehen und wiederverwendet werden. Bestehende Abbildungsfunktionen können für ein spezielles Darstellungsproblem angepaßt werden. Informationsverarbeitungsziele werden jedoch nicht durch graphische Techniken spezifiziert, sondern über textuelle Sprachen.

Das System VISTA von Senay und Ignatius [282] ist eine neuere Arbeit zur Komposition von Visualisierungen (scientific visualization) mit wissensbasierten Methoden. Es werden auch hier verschiedene Techniken zur Komposition von Teilvisualisierungen formalisiert.

Die neueren Entwicklungen zeigen, daß insgesamt der Trend weg von vollautomatischen Systemen und hin zu Systemen mit Interaktionsmöglichkeitnen für den Benutzer bzw. Entwickler geht. Der HAMVIS-Ansatz mit einer Trennung von Entwicklungszeit und Laufzeit (Benutzungszeit) liegt auf der gleichen Linie.

Das Problem bei vollautomatischen Systemen ist, daß Modelle für menschliche Perzeptionsprozesse zur Zeit kaum so detailliert ausgearbeitet sind, daß sie für Präsentationsgenerierungsprozesse nutzbar gemacht werden können (außer in Heuristiken z.B. für Rangordnungen wie in Abbildung 34). Eigenschaften des menschlichen perzeptuellen Systems können zwar im Prinzip zur graphischen Informati-

onsvermittlung ausgenutzt werden (z.B. Kenntnisse über sinnesphysiologische Zusammenhänge, siehe Livingstone [177]).

Die praktische Realisierung dieser Ideen ist aber außerordentlich schwierig. Die Vernachlässigung bestimmter Wahrnehmungsprinzipien resultiert allerdings leicht in einer unintendierten Interpretation einer Präsentation. Die nachfolgenden Abschnitte illustrieren die Wirkung bestimmter Perzeptionsprinzipien und zeigen die Schwierigkeiten auf, die bei der automatischen Generierung von visuellen Darstellungen entstehen.

2.3.4 Perzeptuelle Effekte

Als Teil ihrer Arbeit zu ANDD (Automated Network-Diagram Designer), einen System für die Generierung von Netzwerkdiagrammen, betrachten Marks und Reiter [198] ungewollte „konversationelle Implikaturen“, die in visuellen Notationen auftreten können. Um den Einfluß von graphischen Effekten auf die Bildinterpretation zu verdeutlichen, diskutieren wir ein Beispiel von Marks und Reiter, das hier als Abbildung 36 dargestellt wird. Der linke Teil der Abbildung enthält die intendierte Visualisierung. Im rechten Teil sind leichte Modifikationen vorgenommen worden.

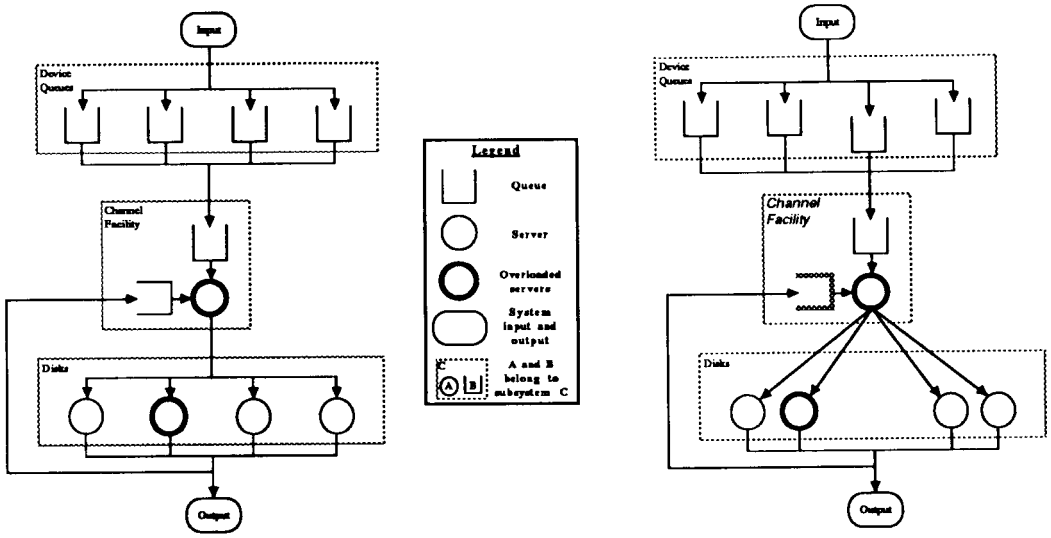


Abbildung 36. Ungewollte konversationelle Implikaturen in Graphiken (aus [198]).

Die Zeichenstiftbreite, die zum Zeichnen der linken Warteschlange der „Channel Facility“-Box verwendet wird, unterscheidet sich von der anderer Warteschlangen. Dadurch wird suggeriert, daß diese Schlange einen besonderen Status hat, was jedoch hier nicht der Fall ist. Der Betrachter nimmt an, daß Objekte mit gleichen Attributen und rhetorischem Status auch mit den gleichen Zeichenattributen präsentiert werden. Die zweite Warteschlange auf der rechten Seite der „Device Queues“ wird ebenfalls als herausgestellt wahrgenommen, da der Designer das Gestalt-Prinzip der guten Fortsetzung sicher eingehalten hätte, falls die Betonung der Warteschlange nicht beabsichtigt wäre. Andere Gruppierungsphänomene spielen ebenfalls eine große Rolle. In Abbildung 36 scheinen zwei Gruppen von Dienstrechnern (servers) zu existieren (Gruppierungsphänomen der „Nähe“). Dieses ist ebenfalls ein ungewollter Effekt. Falls das Layout aber von uninformierten Graphlayoutprogrammen berechnet

worden wäre, gäbe es keine Garantie, daß solche Effekte nicht auftreten könnten. Algorithmen zur Simulation der hier angesprochenen Effekte sind zur Zeit nur für sehr eingeschränkte Anwendungsbereiche und Randbedingungen bekannt (siehe z.B. die Arbeiten von Lowe [182], Rome [265], Wanger et al. [332]).

Die Effekte der perzeptuellen Organisation können auch in konstruktiver Form ausgenutzt werden. Marks und Reiter schlagen sogar vor, perzeptuelle Organisationsphänomene als Ausdrucksmöglichkeiten in die Syntax einer visuellen Notation zu übernehmen:

> *„The most novel aspect of the syntax used by ANDD is the inclusion of relations that describe the perceptual organization of symbols, such as sequential layout (top-to-bottom or left-to-right), proximity grouping, alignment, symmetry, similarity, and ordering. The reasoning for including perceptual grouping and ordering relations in the syntax is that perceptual organization is a property of the human visual system that we cannot disable." ([198], p. 454)*

Was muß also getan werden, um die Graphik im linken Teil aus Abbildung 36 zu erzeugen? Zuerst einmal muß die Anordnung der Knoten berechnet werden. Neben den gegebenen Beziehungen zwischen Knoten (z.B. Geschwister-Relationen) beeinflußt die Topologie der Verbindungen das Layout.[55]

Im Prinzip kann nach Marks und Reiter die Berechnung eines Diagrammlayouts als Einschränkungserfüllungsproblem formalisiert werden, in dem Einschränkungen benötigte syntaktische Relationen modellieren. Nach der Einschätzung von Marks und Reiter scheiden aber Einschränkungserfüllungsansätze aufgrund der hohen Berechnungskomplexität der benötigten Algorithmen aus. Daher schlagen sie einen regelbasierten Ansatz vor, in dem sie Prolog-Regeln verwenden, um Bedingungen zu formulieren, unter denen Netzwerkattribute und Relationen auf Eigenschaften graphischer Objekte und syntaktischer Relationen abgebildet werden [197]. Graphische Eigenschaften werden durch eine weitere Regelmenge unter Berücksichtigung von Prinzipien der perzeptuellen Organisation (Ähnlichkeit und Ordnung) ausgewählt. Abbildung 37 auf der nächsten Seite zeigt ein Beispiel, in dem die visuelle Variable „Größe" dazu verwendet wird, unterschiedliche Objekte zu diskriminieren (und nicht „Form" wie in Abbildung 36). Der Effekt ist, daß Gerätewarteschlangen (device queues), die in diesem Fall aufgrund der Datenflußrelation im oberen Teil der Darstellung gezeigt werden, viel wichtiger erscheinen als Laufwerke (unten gezeigt).

---

55. Für eine Übersicht über Algorithmen zur Berechnung des Layouts von Graphen siehe [33], [36], [64], [72], [69], [78], [126], [262], [297], [313], [340]. Karp et al. beschreiben ein Graph Management System [145]. Im Gegensatz zum VLSI-Design, bei dem kürzeste Pfade berechnet werden müssen, benötigen interaktive Oberflächen schnelle Algorithmen, die ein ästhetisch ansprechendes Layout erzeugen (siehe z.B. [127]).

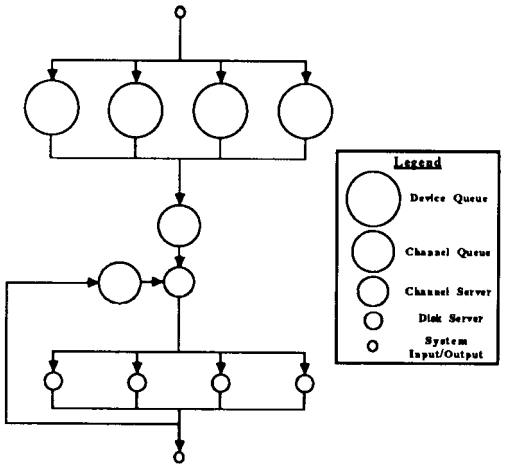


Abbildung 37. Die Verwendung der visuellen Variable „Größe“ als Kennzeichnung des Typs einer Warteschlange resultiert in ungewollten Ordnungsrelation (aus [198]).

Die dritte Regelmenge von ANDD wählt unter Verwendung von Heuristiken die Positionen für Beschriftungen aus. Obwohl Marks und Reiter berichten, daß ihr System in der Praxis gut funktioniert hat, ist es doch schwer, die Abhängigkeiten zwischen den Regeln nachzuvollziehen und – von einer externen Warte – deren Leistungsfähigkeit zu beschreiben. Wie die Autoren selbst zugeben, kann nicht garantiert werden, daß in einem von ANDD generierten Diagramm keine ungewollten Effekte auftreten. Es gibt zur Zeit noch keine allgemeine formale Berechnungstheorie zur Simulation von Effekten der perzeptuellen Organisation (siehe z.B. [264] für einen Überblick über einige relevante Arbeiten aus der kognitiven Psychologie).

Obwohl die Beispiele von Marks und Reiter vielleicht etwas konstruiert und überzogen erscheinen, machen sie doch wesentliche Prinzipien der Wirkung von Perzeptionsphänomenen deutlich. Es wird klar, daß die automatische Produktion von graphischen Darstellungen für kommunikative Zwecke unter Berücksichtigung der in diesem Kapitel aufgezeigten Gruppierungsphänomene eine sehr schwierige Aufgabe ist. Auch für Menschen ist graphisches Design eine außerordentlich anspruchsvolle Tätigkeit. Zur Erstellung bzw. Verbesserung eines Designs gibt es vielfältige Empfehlungen: Kosslyn [164], Schmid [274] und Tufte [321] [322] [323]. Die als Richtlinien formulierten Empfehlungen sind allerdings teilweise schwer nachzuvollziehen, und es ist noch schwerer, sie zu befolgen, da sie oft sogar widersprüchlich sind [315]. Weiterhin werden Aufgaben oder Ziele des Betrachters durch die aufgestellten Richtlinien kaum berücksichtigt, so daß Empfehlungen kaum in ein konstruktives Verfahren umgesetzt werden können.

2.3.5 Antizipationsrückkopplung für Präsentationen

Die heuristischen Rangordnungen der perzeptuellen Aufgaben, die von APT, BOZ und SAGE verwendet werden, um die Kandidatenmenge zu ordnen, greifen zur Schätzung der Aufwandkosten nicht auf Modelle zurück, die auf Basisprinzipien aufbauen (first principles models). Wenn Sprachen kombiniert werden, ist nicht klar, wie der mentale Aufwand bei der Interpretation geschätzt werden kann. Daher ist es auch schwierig, Designalternativen bzgl. der Komposition automatisch zu vergleichen.

In der Tradition der Mensch-Computer-Interaktion stellen Conati und Slack ein Rahmenwerk zur kognitiv-perzeptuellen Kodierung und Verarbeitung von graphischen Präsentationen vor, das Basisprinzipien-Modelle zur Abschätzung des mentalen Aufwandes verwendet ([56], [288]). Der Ansatz sieht vor, gegebene Daten durch Standard-Präsentationstechniken (Geschäftsgraphiken) zu präsentieren und anschließend für Interpretationsaufgaben einzusetzen. Der Präsentationsgenerierungsprozeß wird von Conati und Slack mit Hilfe von Regeln formalisiert.[56] Durch Regeln wird eine Menge von alternativen Präsentationen für eine Menge von Anwendungsdaten bestimmt.

Das Hauptziel der Arbeit von Conati und Slack ist es, eine Basis für die Abschätzung der „Kosten" eines Informationsextrahierungsprozesses zu bieten, und damit den Begriff einer „effektiven Graphik" zu quantifizieren. In ihrem Modell wird der kognitive Aufwand, auf Informationen zuzugreifen, auf der Basis der Verarbeitungszeit von elementaren Verarbeitungsschritten gemessen.

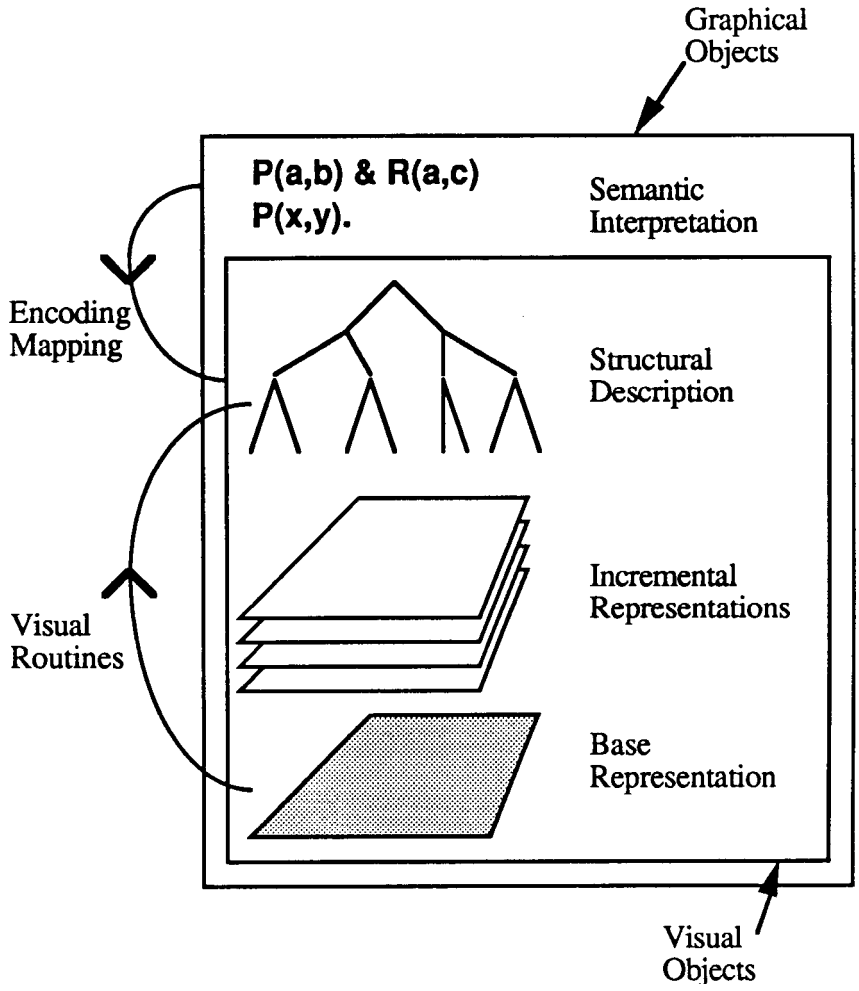


Abbildung 38. Ebene der Repräsentation (nach [56], S. 760). Die Architektur von Conati und Slack verwendet das von Ullman eingeführte Vokabular (cf., [325] S. 110).

56. Die Präsenationsgenerierungskomponente von Conati und Slack ist mit dem Produktionensystem Soar realisiert [168].

Die Architektur von Conati und Slack (siehe Abbildung 38) ist durch das Bildinterpretationsmodell von Ullman [325] inspiriert. Ullmans Modell betrachtet die Verarbeitung von visuellen Informationen über die Erzeugung von frühen Repräsentationen hinaus (cf. [199]). Der erste Schritt ist die Erzeugung von Basisrepräsentationen über den visuellen Eingabedaten. Im zweiten Schritt werden spezielle Verarbeitungseinheiten zur Extrahierung von abstrakten Formmerkmalen und räumlichen Relationen verwendet (in Abbildung 38 als inkrementelle Repräsentationen bezeichnet). Ullman nennt die Operationen visuelle Routinen (visual routines) und argumentiert, daß ein festes Repertoire an Basisoperationen ausreichend ist, um eine unbegrenzte Vielfalt an verschiedenen Formmerkmalen und räumlichen Relationen (wie innen/außen [325], S. 110) herzuleiten. Inkrementelle Repräsentationen werden dazu verwendet, abstrakte strukturelle Beschreibungen von Bildinhalten zu berechnen. Diese Beschreibungen sind die visuelle Kodierung der semantischen Beschreibungen, die wiederum Informationen über die Domäne kodieren (siehe Abbildung 38). Sie explizieren jeweils einen Teil der Informationen, die in der inkrementellen Repräsentation und in der Basisrepräsentation enthalten sind. In Systemen wie APT oder BOZ werden Repräsentationsstrukturen auf der Ebene der strukturellen Beschreibungen durch den speziellen Kodierungsprozeß im Synthesealgorithmus hergeleitet (siehe die Diskussion von oben, vgl. auch das System TRIP2).

Conati und Slack gehen von einem bestimmten Dekodierungsprozeß aus, der das Inverse zur Kodierungsrelation berechnet. Zur Extraktion von Informationen aus Graphiken während des Dekodierungsprozesses definieren sie folgende Elementaroperationen:

- Auswahl und Fokusverschiebung
- Referenzrahmen- und Markierungsoperationen
- Aktivierungsoperationen (Kurvenverfolgung, Regionenaktivierung).

Zur Bearbeitung von Interpretationsaufgaben ist ein Zugriff auf die strukturellen Beschreibungen notwendig. In der Konzeption von Conati und Slack bilden sie die einzige Schnittstelle zwischen den visuellen Informationen und den Zielen höherer Ordnung. Der Zugriff auf strukturelle Beschreibungen bzw. deren Erzeugung ist mit Berechnungskosten verbunden. Eine effektive graphische Darstellung minimiert diese Kosten. Um nun verschiedene graphische Kodierungen bewerten zu können, müssen die Kosten auf eine gemeinsame Einheit umgerechnet werden. Hierzu wird die „Verarbeitungszeit“ gewählt. Verarbeitungszeiten werden dazu verwendet, den „Interpretationsaufwand“ für verschiedene Präsentationen zu vergleichen.

Zusammengesetzte Verarbeitungsziele werden durch Kombination von primitiven Zielen wie `LESEN`, `VERGLEICHEN` und `TENDENZ` formuliert. Als ein Beispiel möchte ich hier die Definition eines `LESEN`-Ziels für die y-Koordinate einer Marke bei gegebener x-Koordinate besprechen. LESE-MARKE kann durch folgende Elementaraktionen realisiert werden:

- Identifikation des x-Werts als visuelles Objekt;
- Bestimmung der visuellen Form der Marke, die andeutet, wie der y-Wert erkannt werden kann;
- Identifikation der Verbundenheitsstruktur von x- und y-Wert;
- Verfolgung der Verbindung, beginnend beim x-Wert, bis der y-Wert erreicht ist;
- Dekodieren oder Ablesen der identifizierten y-Werts

Für jede der möglichen Notationen, die für die graphische Kodierung einer gegebenen Menge von Anwendungsdaten verwendbar sind, können für die obigen Schritte unterschiedliche Verarbeitungszeiten ermittelt werden. Nach der Addition der Verarbeitungszeiten für die Teilschritte wird die Sprache mit den geringsten Kosten ausgewählt.

Der erste Teilschritt, die oben dargestellte LESE-Operation, wird Indizierung genannt. Man versteht darunter den initialen Zugriff auf visuelle Daten, der als Ausgangspunkt für weitere visuelle Informationsverarbeitungsschritte dient. Nach Lohse, der ebenfalls ein kognitives Modell für die Perzeption von Graphiken entwickelt hat, ist das Problem des initialen Zugriffs verantwortlich für die große Varianz bei der menschlichen Performanz bei Informationsextrahierungsaufgaben [179] [180]. In seinem Modell UCIE (Understanding Cognitive Information Engineering) verwendet er ähnliche Basisverarbeitungsschritte für visuelle Informationsverarbeitungsaufgaben. In Abbildung 39 wird ein Beispiel mit einer Folge von elementaren Augenfixierungen gezeigt.

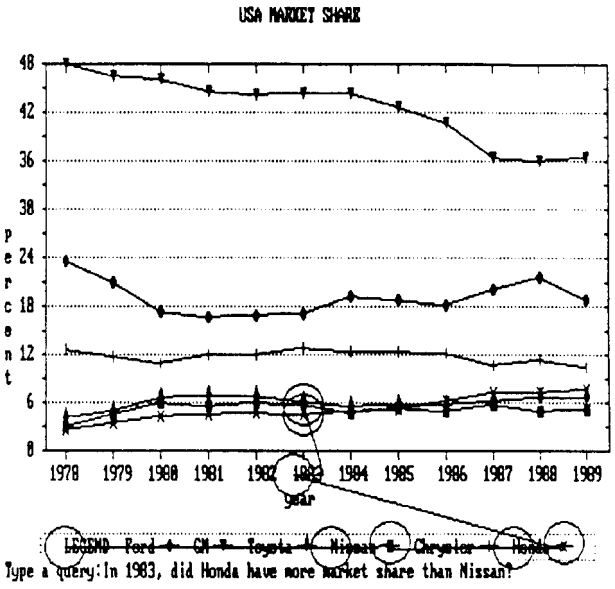


Abbildung 39. Sequenz von Augenfixierungspunkten für eine Graphinterpretationsaufgabe (aus [179], S. 139).

UCIE schätzt die Verarbeitungszeit, die notwendig ist, um eine Frage mit Hilfe eines Graphen zu beantworten. Die elementaren Aufgaben (Anzahl der Augenfixierungen, Verfolgungsdistanzen, Winkel zwischen verschiedenen Augenfixierungen, Anzahl der Objekte bei einem Fixierungspunkt, Diskriminierung zwischen verschiedenen Objekten, Interpolation beim Ablesen eines Wertes von einer Achse) hängen vom Typ des Graphen und vom Fragetyp ab. Kosten für jede Teilaufgabe werden akkumuliert. UCIE berücksichtigt hier detailliertere Informationen über Teilaufgaben der visuellen Perzeption als z.B. GOMS.

UCIE und das Modell von Conati und Slack versuchen, Kosten für elementare Verarbeitungsschritte abzuschätzen (als Einheit wird eine Verarbeitungszeit gewählt). Kosten für elementare Verarbeitungsschritte einer zusammengesetzten Aktion werden akkumuliert. UCIE betrachtet weiterhin die zu verwaltenden Informationen (information load) im Kurzzeit- und Langzeitgedächtnis. Wenn mehrere

graphische Kodierungen möglich sind, wird diejenige mit der laut Modell geringsten Verarbeitungszeit ausgewählt.

Beide Ansätze werfen Probleme auf. Obwohl Conati und Slack generellere Elementaroperationen als Lohse einsetzen, sind die Evaluierungsmethoden doch sehr stark an Graphrepräsentationen ausgerichtet. Es kann nur eine feste Menge an vorgegebenen Alternativdarstellungstechniken verglichen werden. Weiterhin ist nicht klar, wie Bewertungen bei kombinierten Graphiken, wie sie z.B. von APT, BOZ und SAGE generiert werden, verrechnet werden sollen. Die in diesem Abschnitt diskutierten Methoden können angewendet werden, wenn eine Präsentation vollständig fertiggestellt wurde. Es ist aber unklar, wie Antizipations-Rückkopplungstechniken dieser Art während der Konstruktionsphase auf halbfertige Graphiken angewendet werden können. Weiterhin werden perzeptuelle Effekte nicht berücksichtigt. Ein weiteres Problem ist, daß die Fähigkeit, Graphiken zu interpretieren, erst erworben werden muß (siehe [246]). Es tritt hier also das Problem auf, daß auch noch ein Modell des Betrachters berücksichtigt werden muß, um brauchbare Abschätzungen zu erhalten.[57] Für die Integration von verschiedenen Teilmodellen bei der Benutzermodellierung wurde von van Mulken ein Vorschlag ausgearbeitet, der Bayessche Netzwerke zur Modellierung von Schlußfolgerungen des Rezipienten vorsieht [219]. Durch die Verwendung von Bayes-Netzen zur Modellierung von unsicherem Wissen wird auf bekannte, gut erforschte mathematische Mechanismen zurückgegriffen. Weiterhin wurden heuristische Schätzfunktionen zur Modellierung von Benutzerpräferenzen bei der Auswahl von verschiedenen Präsentationsmodalitäten vorgestellt [5].

Die von Conati/Slack und Lohse verwendeten Schätzwerte für den Aufwand, bestimmte Informationen zu extrahieren, basieren auf der Annahme, daß die Teilschritte seriell durchgeführt werden. Forschungsergebnisse im Bereich der menschlichen Wahrnehmung belegen jedoch eindeutig, daß bestimmte Prozesse nebenläufig ablaufen. Schon die vielzitierte Arbeit von Treisman [320] zeigt dieses. Bei einer speziellen Gestaltung des visuellen Stimulus hängt die Zeit, die benötigt wird, um z.B. ein Objekt mit bestimmten Merkmalen zu finden, nicht unbedingt linear von der Gesamtanzahl der Elemente ab. Der in diesem Zusammenhang vielfach verwendete Begriff der präattentiven Wahrnehmung ist allerdings inzwischen wieder umstritten. Die von Treisman verwendeten Stimuli waren künstlich erzeugt. Es ist völlig unklar, wie die Ergebnisse dieser Arbeiten auf Skizzen von Realweltobjekten, wie wir sie in der Einleitung betrachtet haben, übertragen werden können.

2.3.6 Diskussion

Obwohl generelle visuelle Notationen formal definiert werden können (siehe die Arbeiten zur formalen Spezifikation von visuellen Sprachen), ist die Herleitung einer visuellen Sprache für einen bestimmten Einsatzzweck extrem schwierig. Die Arbeiten von Mackinlay haben gezeigt, daß es jedoch für ein gegebenes Repertoire von graphischen Notationen (im Bereich der Geschäftsgraphiken) möglich ist, Präsentationen algorithmisch zusammenzusetzen und Informationen über relationale Daten automatisch zu visualisieren. Eigenschaften der zugrundeliegenden Relationen haben einen Einfluß auf die graphische Kodierung. Mackinlays System verwendet einen Prolog-ähnlichen Wiederaufsetz-Mechanismus, um den Konstruktionsraum aufzuspannen und nach Lösungen zu durchsuchen. Für den Einsatz im Gesamtkonzept von HAMVIS ist jedoch ein (reiner) Wiederaufsetz-Ansatz weni-

57. Ein Simulationsmodell für die erlernte Nutzung einer graphischen Oberfläche wird von Kitajima and Polson [149] vorgeschlagen.

ger gut geeignet. Während der Erstellung ist notwendig, mehrere mögliche Varianten zu betrachten, daher ist für HAMVIS ein Breitensuchansatz bei der Visualisierungskomposition anzustreben. Ähnlich wie bei APT sollen aber Visualisierungen kombiniert werden. HAMVIS muß erstens Kombinationsmöglichkeiten entdecken und zweitens bei einer möglichen Kompositionsvariante die sich daraus ergebenden Einschränkungen verwalten.

Die Arbeit von Casner war der erste Ansatz, der versuchte, den Begriff einer „Aufgabe“ bei der Visualisierungsgenerierung auszunutzen. Die Definition einer Aufgabe (formuliert als „logische Prozedur“ mit „logischen Operatoren“) bestimmt den Präsentationssyntheseprozeß. Logische Operatoren werden durch perzeptuelle ersetzt. Wenn eine Menge von Alternativen für die Wahl eines perzeptuellen Operators zur Verfügung steht, verwendet BOZ ein heuristisches Rangordnungsschema für die Abschätzung des „mentalen Aufwands“ eines perzeptuellen Operators. BOZ wählt den Operator mit dem geringsten Aufwand. In SAGE wird die Anordnung der visuellen Operatoren noch durch Informationsverarbeitungsziele beeinflußt.

Die Diskussion der Ansätze durch Modellierung von visuellen Perzeptionsvorgängen endeten, was die Integrierbarkeit in einen Antizipationsrückkopplungsmechanismus anbetrifft, mit einem Negativresultat. Die enorme Anfälligkeit des menschlichen Perzeptionssystems gegenüber geringsten Störungen (siehe z.B. die Arbeiten von Marks und Reiter) läßt starke Zweifel aufkommen, ob es in absehbarer Zeit gelingen kann, z.B. Gruppierungsphänomene aktiv und in einer stabilen Art und Weise für die Visualisierungsgenerierung auszunutzen. Die Arbeiten von Conati und Slack sowie auch die Modelle von Lohse verdeutlichen den enormen Aufwand, der selbst für simple Geschäftsgraphiken nötig ist, um einen Formalismus zu entwickeln, der es gestattet, Graphiken detailliert auf Effektivität zu vergleichen. Die zur Beschreibung von Perzeptionsvorgängen verwendeten Elementarschritte sind keinesfalls unumstritten. Es erscheint mir wenig sinnvoll, eine Architektur zur Visualisierungsgenerierung zu konzipieren, die darauf aufbaut, daß in nächster Zeit im Bereich der Perzeptionspsychologie stabile Modelle entwickelt werden können, deren Eignung zur Modellierung einer Antizipationsrückkopplung weit über Kriterien wie „Sichtbarkeit“ oder – m.E. schon erheblich problematischer – „Identifizierbarkeit“ hinausgeht.

Arbeiten aus der Perzeptionspsychologie zeigen, daß das menschliche perzeptuelle System sehr sensitiv bezüglich kleiner Störungen reagiert. Dieses wurde schon von Mackinlay und Genesereth in ihren Diskussionen über Ausdrucksfähigkeit und Ausdruckseffektivität erwähnt. Marks und Reiter stellen heraus, daß ungewollte Implikaturen durch perzeptuelle Effekte zu vermeiden sind. Anders betrachtet, können derartige Effekte aber auch zur Kommunikation aktiv ausgenutzt werden. Zu beachten ist, daß nicht nur der visuelle Kanal für die Interaktion und für die Gestaltung von graphischen Darstellungen bedeutsam ist. Barnard und May stellen in [17] eine kognitive Informationsverarbeitungstheorie vor, die die Verarbeitung von Eingaben über mehrere Kanäle berücksichtigt. Beim heutigen Stand der Kunst sind allgemeine, für die Generierung verwendbare Modelle für Perzeptionsprozesse nicht verfügbar.

Mit Einschränkungen können Modelle für die Interpretation von Geschäftsgraphiken aufgestellt werden, die einige Effekte modellieren, so daß Abschätzungen für den mentalen Aufwand bestimmter Interpretationsoperationen möglich werden (siehe die Arbeiten von Lohse, Conati und Slack sowie auch Rome [265]). Es ist allerdings vollständig unklar, wie die Ergebnisse auf andere Anwendungsklassen übertragen werden können, so daß automatische Generierungsansätze auf heuristische Verfahren angewiesen sind.

Die Arbeiten zu APEX sind für HAMVIS für die Inhaltsbestimmung von Visualisierungen relevant. Wie bei APEX, werden auch in der für HAMVIS betrachteten Anwendungsklasse Visualisierungen für physikalische Objekte benötigt. Wir haben gesehen, daß die Synthese von Visualisierungen und deren Komposition von Wissen über Konzepte und Relationen der Anwendungsdomäne abhängt. Es wurde klar, daß aufgrund der Nichtverfügbarkeit von konkreten Objekten zur Entwurfszeit von Visualisierungen für HAMVIS besondere Wissensrepräsentationstechniken entwickelt und eingesetzt werden müssen (perspektivenspezifische Modelle).

Der im Projekt „Integrated Interfaces“ gewählte Ansatz zur Bestimmung von graphischen Kodierungen einschließlich entsprechender Zeichenattribute ist ein Ansatz, wissensbasierte Methoden auch im Bereich der Generierung von Benutzungsschnittstellen anzuwenden (Arens et al. [9]). Durch die explizite Repräsentation von Domänenobjekten und graphischen Objekten kann mit Hilfe von Regeln für eine spezielle Anwendung definiert werden, wie die graphische Kodierung aussehen soll. Obwohl die hier verwendeten Regeln eine bessere Repräsentationsform darstellen als die ersten Ansätze von Zdybel (AIPS) und Friedell, müssen doch die Regeln für jede Anwendung neu definiert werden. Um dieses zu vermeiden, wird in HAMVIS ein Grundmodell für eine Klasse von Anwendungen verwendet. Visualisierungswissen ist bezüglich dieses Grundmodells definiert. Anwendungsspezifische Modelle werden durch das Systementwicklungsteam unter Bezugnahme auf dieses Grundmodell definiert.

Die Betrachtung von automatischen Synthesesystemen, die Präsentationen für gegebene Datenmengen generieren (APT, BOZ, SAGE), haben gezeigt, daß für eine Menge von speziellen, vorgegebenen Darstellungstechniken Präsentationen zusammengesetzt werden können. Präsentationsgenerierungssysteme verwenden Heuristiken, um eine graphische Kodierung zu bestimmen und beziehen sich auf die von Mackinlay aufgestellten Kriterien „Ausdrucksfähigkeit“ und „Ausdrucksstärke“. Die Arbeiten zeigen, daß mit diesen Techniken für statistische Datenmengen teilweise gute Resultate erreicht werden können. Auch Modelle für Aufgaben, die durch die Präsentationen unterstützt werden sollen, können berücksichtigt werden. Durch den abstrakten Kompositionsraum sind aber Abwägungen von Vor- und Nachteilen von verschiedenen Varianten (siehe die Forderungen von Holz für die Gestaltung von Entwurfsmodellen) nicht durchführbar. Obwohl in den frühen Systemen zur Präsentationsgenerierung schon die wichtigen Aspekte der Aufgabenorientiertheit und der Komposition von Visualisierungen behandelt wurden, sind die Architekturen nicht darauf ausgelegt, Varianten zu erzeugen und Informationen für den Designer in adäquater Form innerhalb einer Entwurfs-Oberfläche aufzubereiten. Weiterhin werden Interaktionshandlungen nicht oder nur sehr rudimentär unterstützt.

Im Bereich der Präsentationsgenerierungssysteme verstärkt sich der Trend, einen Designer oder den Benutzer einer Präsentation in die Generierungsarchitektur mit einzubeziehen (siehe insbesondere die Arbeiten von Roth et al. zu SAGE). Das Ziel von HAMVIS ist es ebenfalls, bei der Generierung von direktmanipulativen Visualisierungen für Benutzungsschnittstellen, den Designer zu unterstützen und nicht zu ersetzen. Daher ist es notwendig, zwischen Entwicklungszeit und Laufzeit (Benutzungszeit) zu trennen. Die zur Benutzungszeit verfügbaren Informationen müssen für die Visualisierungsgenerierung verfügbar gemacht werden. Aus der Kommunikationsperspektive betrachtet, müssen die Visualisierungen so generiert werden, daß der Benutzer die benötigten graphischen Interaktionsmöglichkeiten für eine „Antwort“ bzw. einen weiteren „Dialogbeitrag“ hat. Dabei ist zu beachten, daß möglichst eine Visualisierung mehrere Benutzeraktionen unterstützt. Auch in HAMVIS muß also ein

Koordinationsproblem behandelt werden. Dieses Kernproblem bei der Visualisierungsgenerierung wurde schon von APT, BOZ and SAGE betont.

Zur Entwicklungszeit müssen die für die Konzeption der Visualisierungen notwendigen Modelle deklariert werden. Wenn also während des Entwurfs von Visualisierungen und deren Komposition deutlich wird, daß die bisher deklarierten Modelle nicht genügend Informationen enthalten, so kann der Designer weitere Modelle deklarieren oder bestehende Modelle aktualisieren. Im Gegensatz zu den hier besprochenen Systemen zur Präsentationsgenerierung ist also auch die Eingabe für HAMVIS nicht eine feste Menge von Relationen bzw. Tupeln. Die erforderlichen „Eingabedaten“ können auch noch von den Erfordernissen der Visualisierungsgestaltung abhängen. Die Beziehungen zwischen den Wissensquellen müssen daher für den Entwickler transparent gemacht werden. Es kann für HAMVIS kein Verfahren verwendet werden, dessen Ergebnisse und Zwischenergebnisse nicht in einer Benutzeroberfläche für den Designer adäquat präsentiert werden können.

Im vorigen Kapitel wurde deutlich gemacht, daß diese Aspekte Teile des Designmodells bei der Anwendungsentwicklung betreffen. Bei interaktiven Oberflächen, die nicht einen reinen Werkbank-Charakter haben, ist es m.E. sinnvoll, die Gestaltung von Visualisierungen während der Entwicklungszeit aus der Kommunikationsperspektive zu betrachten, um ein Vokabular für die Entwicklung von Teilen des Designmodells zur Verfügung zu haben. Im nächsten Abschnitt werden hierfür relevante Forschungsarbeiten aus dem Bereich Multimedia-Präsentationsplanung geschildert. Die Betrachtung dieser Arbeiten zeigt, wie Kommunikationswissen formal repräsentiert werden kann.

2.4 Intelligente Multimedia-Präsentationsplanung

Nicht in jedem Fall ist es möglich, die graphische Oberfläche eines interaktiven Computersystems zur Entwicklungszeit vollständig zu konzipieren, da u.U. nicht alle Informationen zur Verfügung stehen oder zu viele potentielle Einflüsse berücksichtigt werden müssen. Ein vielfach betrachtetes Beispiel sind Bedienungs-, Wartungs- oder Reparaturanleitungen für technische Geräte. Die Idee ist, anstelle von langwierigen Einführungen, ein interaktives „Handbuch“ automatisch zu generieren, in dem Informationen für die durchzuführenden Tätigkeiten durch Kombination verschiedener Medien und Modalitäten präsentiert werden. Man denke nur an eine Anwendung im Bereich WWW. Präsentationen auf einer HTML-Seite können auch hier automatisch generiert werden. Ggf. ist eine Integration von neuen Präsentationen in schon gezeigte Informationsdarstellungen notwendig usw.

In letzter Zeit wurden neuere Arbeiten zur automatischen Präsentationsgenerierung vorgestellt, wobei der Schwerpunkt auf eine multimediale Informationspräsentation verlegt wurde. Arens et al. sprechen dabei von einem Information-zu-Medium-Zuweisungsprozeß [12]. In diesem Zusammenhang werden die Begriffe „Medium“ und „Modalität“ häufig synonym verwendet. Der Begriff „Modalität“ bezieht sich dabei primär auf den verwendeten Kommunikationskanal und die damit verbundenen Rezeptoren (Auge, Ohr, Tastsinn usw.). Mit „Medium“ ist eine spezielle Kommunikationsform über einen jeweiligen Kommunikationskanal gemeint. Betrachten wir kurz, welche Ressourcen einen Einfluß auf diesen Information-zu-Medium-Zuweisungsprozeß haben. Als Erweiterung zu den Arbeiten im „Integrated

Interfaces“-Projekt verwenden Arens et al. nun generalisierte Regeln und bilden Charakteristiken von Daten auf Charakteristiken von Medien ab. Ein Beispiel:

> *„Zweiertupel werden mit Hilfe von planaren Medien (wie Graphen, Tabellen oder Karten) präsentiert.“*

> *„Daten mit räumlichen Denotationen werden auf Medien mit räumlichen Denotationen präsentiert.“*

Es kommt hier die Idee eines Grundmodells zur Anwendung. Spezielle Zweitupel können z.B. Positionen sein. Für eine konkrete Anwendung werden beispielsweise „Termine“ als spezielle Positionen deklariert und können über die generellen Regeln dann auch auf ein Medium abgebildet werden. Positionen für Einrichtungsgegenstände sind spezielle räumliche Positionen. Räumliche Positionen werden durch die zweite Regel in einer besonderen Weise behandelt usw.

Es ist noch ein weiterer Generalisierungsschritt möglich:

> *„Falls mehr als ein Medium verwendet werden kann, und es schon eine Präsentation existiert, bevorzuge das Medium, das schon für die bestehende Präsentation verwendet wird.“*

> *„Falls mehr als ein Medium verwendet werden kann und noch Zusatzinformationen dargestellt werden müssen, bevorzuge das Medium, das auch für die weiteren Informationen geeignet ist.“ ([11], S. 283f., sinngemäße Übersetzung: RM)*

Die Zuweisung von Medien wird auf einer groben Granularitätsstufe betrachtet. Arens et al. charakterisieren Medien mit den Attributen aus Abbildung 40.

Generic Medium	Carrier Dimension	Int. Semantic Dim.	Temporal Endurance	Granularity	Medium Type	Default Detectability	Baggage
Beep	0D		transient	N/A	aural	high	
Icon	0D		permanent	N/A	visual	low	
Map	2D	>2D	permanent	continuous	visual	low	high
Picture	2D	∞D	permanent	continuous	visual	low	high
Table	2D	2D	permanent	discrete	visual	low	high
Form	2D	>2D	permanent	discrete	visual	low	high
Graph	2D	1D	permanent	continuous	visual	low	high
Ordered list	1D	#D	permanent	discrete	visual	low	low
Unordered list	0D	#D	permanent	N/A	visual	low	low
Written sentence	1D	∞D	permanent	discrete	visual	low	low
Spoken sentence	1D	∞D	transient	discrete	aural	medhigh	low
Animated material	2D	∞D	transient	continuous	visual	high	high
Music	1D	∞D	transient	continuous	aural	med	low

Abbildung 40. Überblick über die von Arens et al. betrachteten Mediencharakteristiken (aus [11], S. 291).

Präsentationsgenerierungssysteme für Geschäftsgraphiken betrachten vom Prinzip her die visuelle Kommunikation als Einweg-Prozeß, bei dem ein Sender (Kodierer) mit einem Empfänger (Dekodierer) über einen Kanal mit Hilfe von kodierten Nachrichten in Verbindung tritt. Im Gegensatz zu dieser simplifizierten Betrachtungsweise von Kommunikation als Übertragung von Informationen über einen Kanal argumentierte z.B. Appelt dafür [7], Sprache als eine Menge von Aktionen zu betrachten, die ein Agent zur Verfügung hat, um den mentalen Zustand eines anderen Agenten zu beeinflussen. Durch einen Planungsprozeß wird eine Sequenz von elementaren (Kommunikations-)Aktionen aneinandergereiht, so daß bestimmte Ziele erfüllt werden, die in Bezug auf ein Modell des mentalen Zustands des Rezipienten definiert sind.

Auf der Wissensebene identifizieren Arens et al. verschiedene, grob umrissene Wissensarten, die für die Organisation der Kommunikation von Informationen relevant sind:

- Charakteristiken der verwendeten Medien,
- Art der Information, die vermittelt werden soll,
- Präsentationsziele und Merkmale des Produzenten,
- Charakteristiken des Rezipienten (Kenntnisse, Interessen, Fähigkeiten),
- die kommunikative Situation,
- Wissen über die Welt.

Modelle für Inferenzprozesse über die Gestaltung der Kommunikation unter Verwendung dieser Wissensarten müssen vielfache Abhängigkeiten handhaben können. Abbildung 41 skizziert ein Netzwerk, das die Abhängigkeiten zwischen den Wissenquellen charakterisiert. Beim heutigen Stand der Kunst ist allerdings weder klar, wie die einzelnen Wissensquellen formal modelliert werden können, noch wie sie kombiniert werden können, um die Abhängigkeiten zu modellieren.[58]

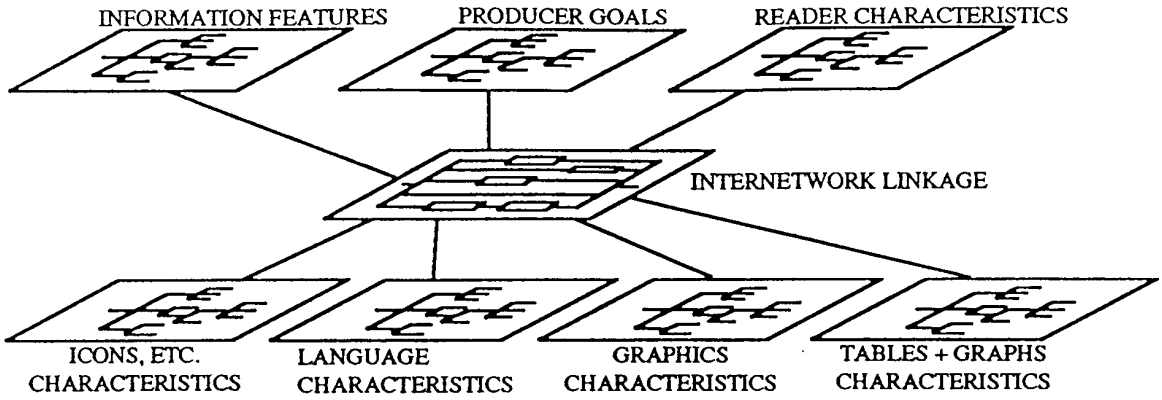


Abbildung 41. Schematische Darstellung der Wissensquellen für die Generierung von Multimedia-Präsentationen (aus [11], S. 287).

Im Gegensatz zu Hypertext- oder, etwas moderner, Hypermedia-Systemen, bei denen zum großen Teil der Benutzer oder der Autor (z.B. von *vorgefertigten* Bildsequenzen) verantwortlich für die Informationsauswahl ist, beschäftigen sich Intelligente Multimedia-Präsentationssysteme (IMMPS) damit, Informationen für vorgegebene Präsentationsziele automatisch durch wissensbasierte Techniken zusammenzustellen und adäquat zu präsentieren (hierzu gehört z.B. die koordinierte Generierung von graphischen Darstellungen und natürlicher Sprache).

Präsentationsziele können in einem IMMPS-Kontext durch das Präsentationssystem auch dynamisch generiert werden, indem das Verhalten des Benutzer in einem Rückkopplungsschritt analysiert wird. Denkbar ist z.B. bei technischen Geräten der Einbau von Sensoren, mit denen überprüft werden kann, ob der Benutzer die in einer bestimmten Anwendungssituation „richtigen“ Tasten gedrückt hat. Falls nicht, kann in der multimodalen Präsentation ein entsprechender Hinweis gegeben werden usw.

In ihrem Überblick über IMMPS [266] verwenden Roth und Hefley in diesem Zusammenhang den Begriff „intelligent“, weil die Informationsauswahl und die Kombination der Informationen, die durch ein IMMPS in einer speziellen Benutzungssituation zusammen dargestellt werden, nicht vorhersehbar ist.[59] Insbesondere sind die zu berücksichtigenden Querbezüge bei Verwendung verschiedener

58. Arens et al. schlagen vor, UND-ODER-Entscheidungsnetzwerke mit Merkmalsbeschriftungen an den Kanten zu verwenden (sog. systemic networks), um die oben verbal widergegebenen Regeln formal zu repräsentieren. Ähnliche Techniken wurden erfolgreich für das natürlichsprachliche Generierungssystem Penman angewendet [131].

59. Die Arbeit von Roth und Hefley [266] enthält einen ausführlichen Literaturüberblick über verschiedene IMMPS-Anwendungsbereiche, Generierungsarchitekturen, Informationstypen und Modellierungsansätze. Eine Zusammenfassung der Literatur über die wissensbasierte Generierung von multimedialen Präsentationen findet man in [204]. Eine Klassifikation von IMMPS sowie ein umfangreicher Literaturüberblick wurde auch von Wahlster et al. [331] publiziert.

Medien nicht im Vorwege antizipierbar. Daher wird ein generativer Ansatz benötigt, bei dem Wissen über Kommunikationstechniken eine große Rolle spielt. In einem IMMPS soll also das Wissen der Autoren eines Hypermedia-Präsentationssystems durch Programmkomponenten verfügbar gemacht werden. Die Begriffe „Entwicklungszeit“ und „Benutzungszeit“ haben hier einen anderen Charakter. Nach der Entwicklung (bzw. hier: Generierung) einer *konkreten* multimodalen Präsentation ist eine Evaluierung durch einen menschlichen Betrachter nicht vorgesehen. Hier ist also die agentenorientierte Sichtweise der Mensch-Computer-Interaktion vorherrschend.

IMMPS-Architekturen sind durch verschiedene Ansätze der Generierung natürlicher Sprache beeinflußt. Ein grobes Szenario mit verschiedenen Instantiierungsmöglichkeiten zeigt Abbildung 42.

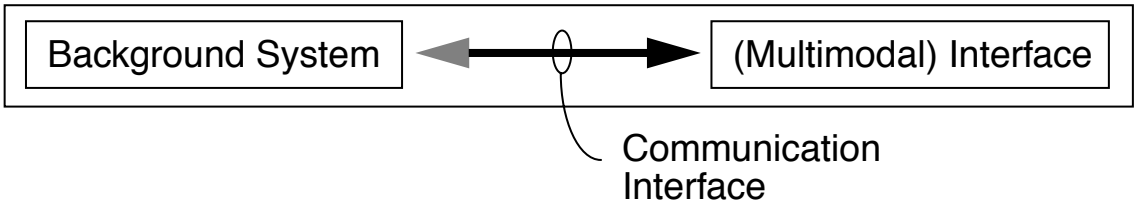


Abbildung 42. Kommunikationssicht der multimodale Interaktion zwischen Menschen und Computern.

Kölln diskutiert verschiedene Generierungsansätze für natürliche Sprachen zusammen mit den jeweiligen Anwendungen ([160], [161]) und zeigt zentrale Fragestellungen auf, die auch für IMMPS relevant sind. Die Architektur von Generierungssystemen hängt von der Anwendungs- und Leitdomäne und deren Anbindung an das IMMPS ab. Bei *Reportgeneratoren* (z.B. für Börsenberichte, Wettervorhersagen, Arbeitsmarktberichte usw.) besteht das Hintergrundsystem aus einem Faktengenerator, durch den Fakten aus einer Datenbasis ausgewählt und gefiltert werden. Die Oberflächenkomponente ihrerseits bereitet die Informationen zur Kommunikation auf und reichert sie ggf. so an, daß beispielsweise eine verbale Kommunikationsform adäquat unterstützt werden kann. Andere Anwendungsbereiche sind z.B. natürlichsprachliche Oberflächen für *Expertensysteme*. Das Hintergrundsystem aus Abbildung 42 wird also mit ein oder mehreren Problemlösungskomponenten instantiiert (siehe auch [1], [41], [216], [217], [303]). Die natürlichsprachliche oder multimodale Oberfläche wird in dieser Konzeption zur „Betreibung“ des Expertensystems und für Erklärungsmöglichkeiten verwendet. Eine spezielle Instanz ist auch ein Tutorsystem, in dem der Computer nicht nur als Dialogpartner verstanden wird, sondern in einigen Fällen sogar als „Aufseher“ fungiert [337].

Die strikte Trennung von Hintergrundsystem und Oberfläche kann in einigen Anwendungen wie Reportgeneratoren noch aufrecht erhalten werden. Allerdings werden die Abhängigkeiten von Kommunikations- und Domänenwissen bei Tutorsystemen immer stärker. Schlußfolgerungen über die Gestaltung der Kommunikation sind hier unvermeidbar mit Schlußfolgerungen über Domänenkonzepte verbunden. Die Unterscheidung zwischen einem Hintergrundsystem und einer Oberfläche wird zunehmend unklarer.

Dieses hat dazu geführt, daß sich die agentenorientierte Sicht der Kommunikationsplanung durchgesetzt hat. Kommunizieren wird als zielgerichtetes Handeln interpretiert. Insbesondere für natürlichsprachliche Kommunikationssysteme wird für eine Sequenz von Kommunikationshandlungen verschiedener Agenten häufig der Begriff *Diskurs* verwendet. Mit dem Begriff des Diskurses sind nicht nur die Konstituenten eines Textes und ihre Bedeutung, sondern auch die kommunikative Situa-

tion (u.a. beschrieben durch die Diskurshistorie), die Charakteristiken der Dialogteilnehmer und auch das relevante Weltwissen erfaßt (siehe [284], S. 233-245). Die Gesamtheit der für die Planung einer Kommunikationshandlung verwendeten Repräsentationsstrukturen für die verschiedenen Wissensquellen wird als *Diskursstruktur* bezeichnet. Die Ziele des Präsentierers und die daraus resultierenden Einflüsse auf den Diskurs werden als *intentionale Aspekte* des Diskurses bezeichnet. Relevant sind auch *attentionale Aspekte*: Fokusverschiebung, Aufmerksamkeitssteuerung, usw. Dem Präsentierer stehen „*rhetorische Mittel*“ bereit, um den Diskurs zu strukturieren und damit seine Ziele zu erreichen. Eine *rhetorische Relation* charakterisiert das Zusammenwirken von Teilkonstituenten und beschreibt deren Gesamtwirkung in Bezug auf den mentalen Zustand des Rezipienten und die Ziele des Präsentierers (Mann et al.: „Rhetorical Structure Theory“, siehe [192] [193]). Innerhalb einer rhetorischen Relation übernehmen Konstituenten bestimmte *kommunikative Funktionen*. Die Diskursstruktur erfaßt also primär, „*was*“ kommuniziert werden soll. Durch rhetorische Relationen und kommunikative Funktionen sind aber Einschränkungen für das „*Wie*“, d.h. für die „oberflächennahe“ Repräsentation, definiert. Sofern die Diskursstruktur beim Übergang auf „oberflächennahe“ Repräsentationsformen wesentlich umstrukturiert wird, spricht man für die resultierende Repräsentationsform von einer *Präsentationsstruktur.*

Es ist noch zwischen einer möglichen (vorausgeplanten) Diskursstruktur und einer konkreten, in einer bestehenden Kommunikationssituation instantiierten Diskursstruktur zu unterscheiden. Für die vorausgeplante Diskursstruktur wird auch der Begriff *Diskursstrukturmodell* verwendet, obwohl auch hierfür der Begriff Diskursstruktur gebräuchlich ist. Für die Repräsentation der Konstituenten einer Präsentation mit ihren Bedeutungen und Beziehungen möchte ich im HAMVIS-Kontext den eingeschränkteren Begriff der *Dialogstruktur* verwenden. Eine Dialogstruktur umfaßt nicht die Modelle für die Ziele des Präsentierers, die mentalen Zustände des Rezipienten und die Modelle für das Weltwissen. Bei nicht-interaktiven Präsentationen ist für diese Teilstruktur auch der Begriff *Dokumentstruktur* gebräuchlich (vgl. die Terminologie von WIP [331]). Ich möchte aber nicht jedesmal zwischen interaktiven und passiven, nicht-interaktiven Präsentationen unterscheiden, so daß ich den Begriff Dialogstruktur auch für passive Präsentationen verwende.

Die für den Übergang zur UIMS-Ebene notwendigen Repräsentationsformen werden im HAMVIS-Kontext ebenfalls als Präsentationsstruktur bezeichnet. Zur Unterstützung von Benutzeraktionen durch graphische, direktmanipulative Interaktionstechniken müssen die jeweils benötigten Interaktionswerkzeuge bereitgestellt werden. Letztendlich müssen Graphiken erzeugt werden, die wiederum manipulierbar sind.

Neben den „*Termen für den Interaktionsrahmen*“ (Fenster, Menüs, Graphikfenster, Standard-Interaktionsobjekte, Kommando-Eingabefenster etc.) müssen in der Präsentationsstruktur auch die „*Terme zur Konstruktion einer Benutzerantwort*“ (Hovy 1993, pers. Komm.) bereitgestellt werden.

Für IMMPS ergeben sich folgende Forschungsthemen, die auch im HAMVIS-Kontext relevant und schon in vorigen Abschnitten angeklungen sind (vgl. [160], [266]):

- Modellierung der Beziehungen zwischen Kommunikationsplanung und dem Verhalten des Gesamtsystems, insbesondere die Modellierung von Konzepten zur der Beschreibung der Kommunikationsschnittstelle (siehe Abbildung 42),
- Bewertungskriterien für generierte Präsentationen,

- Gesichtspunkte der Diskursgestaltung: intentionale, rhetorische und mediale Aspekte
- Unterstützung für interaktive Eingaben

In den nachfolgenden Abschnitten werden die aufgeführten Punkte detaillierter besprochen.

2.4.1 Einbettung der Präsentationsplanung in ein Gesamtsystem

Im vorigen Abschnitt wurden verschiedene Klassen von Anwendungssystemen aufgeführt. Um nun die Modularität des Systemdesigns zu erhöhen und eine Trennung von Hintergrundsystem und Oberflächen zu erzielen, müssen für die Kommunikationsschnittstelle Konzepte zur Charakterisierung der zu präsentierenden Informationseinheiten definiert werden. In Abhängigkeit von der Applikation gibt es hierzu verschiedene Möglichkeiten:

- *Domäneninformationen* sind in Form von „Fakten“ als ein Ausschnitt der Wissensbasis vorgegeben. Fakten können sowohl Konzepte oder Relationen als auch Assertionen beschreiben. Das Wissensrepräsentationssystem definiert die Charakteristiken dieser Fakten (siehe Abschnitt „Automatisches Design von graphischen Präsentationen“ and Abschnitt „Diskussion“). Es ist möglich, daß das Präsentationsgenerierungssystem die Menge der „initialen Fakten“ selbständig erweitert (dieses kann aufgrund von rhetorischen Erwägungen erfolgen oder nur weil eine Darstellung dann aus z.B. aus optischen Gründen „abgerundet“ werden kann).
- *Informationsverarbeitungsziele* des Rezipienten können explizit modelliert und dem IMMPS als Eingabe zur Verfügung gestellt werden, um das Präsentationsdesign auf diese Ziele abzustimmen. Bei interaktiven Systemen können Informationsverarbeitungsziele wiederum aus der Interaktion abgeleitet werden (siehe die Arbeiten zu automatischen Hilfesystemen [318] oder Tutorsystemen [217]).
- *Aktionen- oder Aufgabenmodelle* sind ebenfalls eine mögliche Beschreibung der zu präsentierenden Informationen, insbesondere wenn die zu generierenden Präsentationen gleichzeitig hierfür geeignete Interaktionsformen bereitstellen sollen.
- *Kommunikative Ziele* modellieren die Eingabe für ein IMMPS aus der Perspektive eines Präsentationsplanungsagenten. Üblicherweise werden kommunikative Ziele eines Agenten in Bezug auf ein (formales) Modell des Rezipienten modelliert.[60] Kommunikative Ziele reflektieren Informationsverarbeitungsziele des Benutzers. Aktionen bzw. Aufgaben können ebenfalls aus der Perspektive der Ziele eines Präsentierers modelliert werden. Bei Verwendung dieser Metapher steht die agentenorientierte Sicht der Mensch-Computer-Interaktion im Vordergrund.

Eine kanonische, anwendungsunabhängige Repräsentation, die als IMMPS-Eingabe für alle Anwendungsdomänen dienen kann, läßt sich jedoch kaum finden. Rambow [253] modelliert daher anwendungsspezifische Informationen als „Domänen-Kommunikationswissen“ (domain communication knowledge), das nicht direkt für automatische Problemlösungsprozesse verwendet wird, sondern extra für Kommunikationszwecke repräsentiert werden muß (siehe auch [333]). Zu beachten ist, daß für

60. Siehe [153] für einen Überblick über das Gebiet der „Benutzermodellierung“ und generelle Positionen über Einsatzmöglichkeiten und -grenzen dieser Modelle. Spezielle Themenbereiche wie die Behandlung von konversationellen Implikaturen findet man z.B. [42], [348] und [349].

Präsentationsgenerierungsprozesse auch Domänenwissen in entsprechende Schlußfolgerungsprozesse eingehen kann (siehe z.B. [37] für einen Ansatz, in dem Schlußfolgerungsprozesse auf der Domänenebene und auf der Kommunikationsebene kombiniert werden). Die Unterscheidung zwischen Hintergrundsystem und Kommunikationsoberfläche ist, ähnlich wie bei UIMS-Architekturen, nur auf der logischen Ebene vorhanden.

2.4.2 Bewertungskriterien für Präsentationen

Simulative Modelle für menschliche Perzeptionsprozesse, die von Basisprinzipien ausgehen (first principles models) und in einem IMMPS eingesetzt werden können, sind nicht verfügbar (siehe die Diskussion in Abschnitt „Antizipationsrückkopplung für Präsentationen“). Heuristische Modelle (second principles models) decken jeweils nur einen kleinen Teil der Effekte ab, die für die Bewertung von Präsentationen insgesamt relevant sind. Generelle *Konversationsmaximen* wie die von Grice [103] oder Normans Angemessenheitskriterien (*appropriateness criteria* [235]) sind viel zu generell, um direkt in einem IMMPS verwendet werden zu können. Richtlinien für interaktive Oberflächen sind sehr schwierig zu formalisieren (siehe z.B. [255], [194], [236]).

Als Erweiterung von Arbeiten zu natürlichsprachlichen Generierungssystemen wurden verschiedene Kohärenzkriterien für den internen Zusammenhang und die Abstimmung von (multimodalen) Präsentationsteilen aufgestellt [16]:

- Syntaktische Kohärenz und Kontinuität: oberflächenorientierte Phänomene

  Im Gegensatz zu Syntaxregeln der natürlichen Sprache, die Kombinationsmöglichkeiten zwischen Worten doch schon relativ stark einschränken (Grammatik, Subkategorisierung, Flexion, Kopfprinzip, X-Bar usw.), sind die Beziehungen zwischen visuellen Präsentationseinheiten auf syntaktischer Ebene ohne die Fokussierung auf spezielle visuelle Notationen kaum einschränkbar.

  Bandyopadhyay hat einige Kontinuitätsprinzipien angegeben, die folgende Darstellungsparameter betreffen: Perspektive, Fokus, räumliche Position, Zeichenattribute, unterstützte Aktion(en). Das Kriterium der Kontinuität charakterisiert den Effekt, daß sich in einer Darstellung nichts abrupt ändern sollte, zumindest nicht, wenn der Grund der Änderung (z.B. als Konsequenz einer durchgeführten Aktion) nicht deutlich wird (oder explizit gemacht wird).

- Semantische Kohärenz: Inhalt und globale Strukturierung eines Diskurses

  Präsentationseinheiten müssen zueinander in Beziehung gesetzt werden, so daß die Themaentwicklung aus einer globalen Perspektive wohlstrukturiert ist (z.B. stellt eine Einheit den Hintergrund für eine andere Präsentationseinheit bereit oder dient als Motivation).

  Präsentationseinheiten beeinflussen sich also gegenseitig auf spezielle Art und Weise. Diese Beziehungen müssen durch ein IMMPS koordiniert werden. Auch die Reihenfolge von Präsentationseinheiten kann auf semantischer Ebene bedeutsam sein.

- Pragmatische Kohärenz: Effektivität eines Diskurses in einer speziellen kommunikativen Situation

> Prinzipien für pragmatische Kohärenz betrachten die kommunikativen Rollen von Präsentationsteilen in Hinblick auf die Intentionen des Präsentierers und die Einflüsse auf den „mentalen Zustand“ des Rezipienten (Benutzermodell). Dieses Kohärenzkriterium beeinflußt z.B. die Auswahl von präsentierten Objekten in Hinblick auf die zu unterstützenden Handlungen des Rezipienten. Ein anderes Beispiel wäre, das Kaufverhalten des Rezipienten zu beeinflussen (Werbung).[61]

Ein anderer Aspekt, der durch den Begriff der Kohärenz berührt wird, ist die Vermeidung von Redundanz. Dieses ist besonders für linearen Text wichtig, aber auch für Bilder nicht irrelevant. Die natürliche Sprache hat verschiedene Techniken zur Redundanzvermeidung entwickelt (Pronominalisierung, Anaphern). In Präsentationen, die in einem auf Papier gedruckten Handbuch erscheinen (eventuell kombiniert mit Text), sollte ein Teilbild ebenfalls nicht ständig wiederholt werden. Im Kontext von interaktiven Oberflächen gelten für die Gestaltung eines Interaktionsfensters ähnliche Überlegungen. Wie in der Einleitung schon besprochen, ist es sinnvoll, z.B. das mehrfache Zeichnen eines Flugzeugrumpfes in verschiedenen Teilfenstern (soweit möglich) zu vermeiden. Diese Art von Redundanzvermeidung sollte durch Komposition von Visualisierungen ermöglicht werden.

Auch ein Kohärenzbegriff ermöglicht nur eine schwache Charakterisierung dessen, was als Ergebnis des Generierungsprozesses als Ausgabe tatsächlich abzuliefern ist. Meines Erachtens können aber hieraus Merkmale der Architektur zur Generierung von Präsentationen motiviert werden.

HAMVIS versucht, z.B. für eine Aktivität des Benutzers eine Dialogstruktur mit einem festen Layout der Teilfenster zu finden. Das Layout der Teilfenster sowie auch die in den Teilfenstern dargestellten Präsentationen der Anwendungsobjekte dürfen sich während der Interaktion nicht ohne Grund ändern. Auch wenn bestimmte Handlungsalternativen aufgrund des Zustands der Anwendung zu einem bestimmten Zeitpunkt nicht möglich sind, sollte die hierfür vorgesehene Darstellungsfläche nicht gelöscht werden. Notwendige Änderungen sollten sich aus der durchgeführten Handlung erklären. Redundanz hat bei interaktiven Systemen also einen völlig anderen Stellenwert als bei linearem Text. Redundanz ist aus dieser Sicht in einem gewissen Sinne sogar anzustreben. Änderungen sollten durch spezielle Darstellungstechniken „angekündigt“ werden (Techniken zur Konditionierung).

Die wichtigste Erkenntnis aus der Betrachtung der verschiedenen Kohärenzbegriffe scheint mir die Notwendigkeit einer kommunikationsorientierten Sicht auf die Gestaltung von Visualisierungen, die in einer interaktiven Oberfläche verwendet werden sollen, zu sein. Präsentationsteile stehen in Beziehungen zueinander und haben in Hinblick auf das Ganze jeweils eine bestimmte kommunikative Funktion. Formale Modelle zur Repräsentation von Kommunikationswissen wurden für IMMPS-Generierungsarchitekturen, die im nächsten Abschnitt näher vorgestellt werden, entwickelt.

---

61. Offensichtlich ist es nicht möglich, pragmatische Kohärenz absolut zu messen. In Dialogsystemen können allerdings Hinweise darauf, daß kommunikative Ziele nicht erreicht werden konnten, aus dem (unerwarteten) Verhalten des „Gesprächspartners“ abgeleitet werden. Moore schlägt folgerichtig einen sog. reaktiven Ansatz für die Kommunikationsgestaltung vor, in dem ein Benutzermodell des menschlichen Dialogpartners inkrementell aus der Interpretation seines Verhaltens aufgebaut wird [217]. In einigen Domänen können zu diesem Zweck auch physikalische Rückkopplungsmechanismen im Design eines Artefakts vorgesehen werden (z.B. ob ein Knopf gedrückt wurde oder nicht).

2.4.3 Modellierung und Anwendung von Präsentationswissen

Inspiriert durch Forschungsergebnisse zu Architekturen für die Generierung natürlicher Sprache wurden verschiedenartige Präsentationsgenerierungsarchitekturen konzipiert. Für die Koordination von verschiedenen Wissensquellen und internen Repräsentationsformen sowie für die Interaktion zwischen den Hauptverarbeitungseinheiten (vgl. Abbildung 41) wurden in der Literatur verschiedene Vorschläge gemacht.

Konzeptuelle Generierungsarchitektur

Um die Architekturen verschiedener IMMPS vergleichen zu können, wurde von Roth und Hefley eine generische (oder konzeptuelle) Architektur vorgestellt, die die wesentlichen Verarbeitungseinheiten umfaßt und hier als Abbildung 43 noch einmal graphisch widergegeben ist.

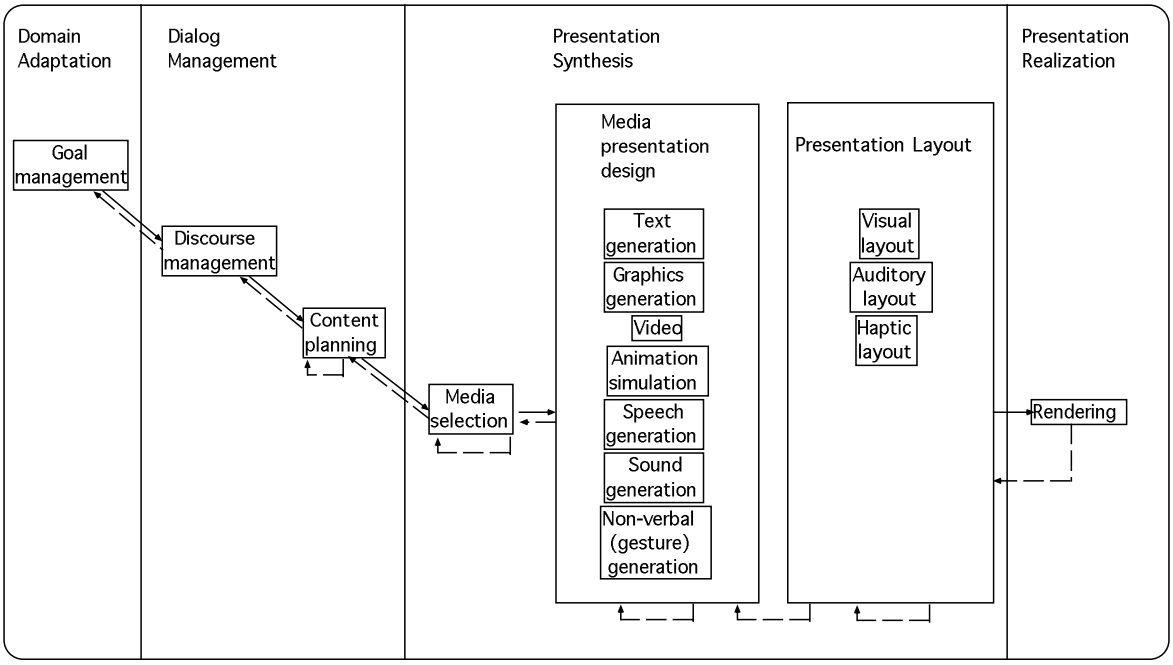


Abbildung 43. Konzeptuelle Architektur für multimodale Kommunikationssysteme (aus [266], S. 20).

Die komplexe Aufgabe der Präsentationsplanung wird (logisch) in mehrere Ebenen eingeteilt:

- Inhalts- und Diskursplanung („What-to-say-“ oder strategische Komponente)
- Auswahl eines speziellen Mediums oder einer bestimmten Modalität (zusammen mit einer Partitionierung der zu vermittelnden Informationen)
- Präsentationsdesign und -layout für eine Partition („How-to-say-“ oder taktische Komponente)
- Koordination von Medien und Modalitäten

Die Eingabe für ein IMMPS wird durch eine Verwaltungskomponente für Präsentationsziele (goal management unit) bereitgestellt. Ausgaben werden direkt an den Betrachter weitergeleitet (agentenorientierte Architektursicht), d.h. es gibt keine Unterscheidung zwischen Entwicklungszeit- und Benutzungszeitaktivitäten. Ausgaben werden (üblicherweise) nicht durch einen menschlichen

Systementwickler bewertet, nachbearbeitet oder direkt beeinflußt. Ein IMMPS kann zwar eine Inspektionsoberfläche (z.B. zur Fehlersuche) enthalten, eine für einen Designer adäquate Oberfläche wird i.a. jedoch nicht direkt die für die Fehlersuche notwendigen Darstellungen von „internen“ Strukturen enthalten. Die Konzeption einer Oberfläche, die für einen Designer geeignet ist, beeinflußt sicher auch die ganze Konzeption der Architektur und der internen Datenstrukturen eines IMMPS (vgl. die Ausführungen zu dem System SAGE).

Beträchtliche Meinungsunterschiede bestehen über die konkrete Instantiierung der konzeptuellen IMMPS-Architektur aus Abbildung 43 (cf. [12]). Um Aspekte der Architektur von HAMVIS zu motivieren, möchte ich kurz die Hauptansätze der konkreten Gestaltung von IMMPS-Architekturen diskutieren: COMET/IBIS ([83], [84], [279], [280]), SAGE ([266]), WIP ([330], [261], [3], [331]), die Arbeiten von Arens et al. [12], die Arbeiten von Maybury ([201], [202], [203]) und einige andere ([232], [29]).

Schichten und Datenstrukturen eines IMMPS

Aus der Systemmodellierungs- und Software-Engineering-Perspektive ist es wünschenswert, verschiedene Schichten (oder Ebenen) zu identifizieren, denen jeweils bestimmte Prozesse und Aufgaben zugeordnet werden können. Für eine IMMPS-Architektur müssen Kontrakte und Protokolle für die Kommunikation von Informationen zwischen den Ebenen vereinbart werden.

Bei fast allen Systemen wird neben den Schichten zur Ein- und Ausgabe eine medienunabhängige intentionale Schicht und eine präsentationsorientierte Schicht verwendet. Es müssen entsprechende Repräsentationsstrukturen für den (dynamischen) Planungsprozeß spezifiziert werden. Neben Wissensstrukturen wie Domänenmodellen, Benutzermodellen usw. werden für die Präsentationsplanungs- und Kompositionsprozesse üblicherweise hierarchische, baumartige Strukturen mit Aggregaten und Konstituenten verwendet (obwohl auch hier wieder die konkrete Ausprägung und Ausrichtung von System zu System stark variiert):

- COMET/IBIS verwendet eine Nachrichtentafel-Struktur (blackboard structure), die während des Planungsprozesses um neue Informationen ergänzt, jedoch niemals vollständig restrukturiert wird.
- Maybury verwendet ebenfalls eine einzige Datenstruktur, um Dialog- und Präsentationsplanungsinformationen zu verwalten.
- Das WIP-System erzeugt Planungsstrukturen dynamisch und inkrementell auf verschiedenen Ebenen und unterstützt dadurch ein nebenläufiges Vorgehen bei der Diskursplanung, bei der Medienzuweisung und bei der detaillierten Präsentationsplanung. Die Planungsstruktur von WIP kann lokal umstrukturiert werden (s.u.). Die Motivation für das inkrementelle Vorgehen ist durch die Problematik der Handhabung von Abhängigkeiten von strategischer und taktischer Planungsebene gegeben. Die inkrementelle Architektur von WIP wurde auch aus Erwägungen der Echtzeitgenerierung entworfen. In einigen Fällen stehen vielleicht nicht genügend Ressourcen bereit, um eine vollständige Diskursplanung durchzuführen.

- Arens et al. argumentieren, daß die Gesamtarchitektur wesentlich weniger kompliziert wird, wenn zunächst eine Planung auf der intentionalen Ebene zur Erstellung einer Dialogstruktur durchgeführt wird und diese anschließend in eine neue, oberflächenorientierte Präsentationsstruktur überführt wird. In [12] diskutieren sie ein Beispiel, das verdeutlicht, wie diese zwei Strukturen vom Konzept her aussehen können und was sie enthalten.

Für HAMVIS halte ich eine inkrementelle Generierungsarchitektur nicht für notwendig, da zwischen Entwurfszeit und Laufzeit getrennt wird. Zur Entwicklungszeit kann ein hoher Ressourcenverbrauch an Rechenzeit und Speicherbedarf toleriert werden. HAMVIS versucht durch Vorausplanung, das Laufzeitsystem effizient zu gestalten. Eine starke Verflechtung von Verarbeitungseinheiten bzw. Arbeitsschritten ist m.E. während der Entwurfszeit für den Designer nur schwer nachzuvollziehen.

Durch die bestehenden IMMPS-Architekturen werden Designvarianten nicht in einer Art und Weise verwaltet, so daß die Vor- und Nachteile einer Präsentationsform gegenüber Alternativen hinterher (durch einen menschlichen Designer) abgewogen werden können. Die von IMMPS für die Planung einer Diskurs- und Präsentationsstruktur verwendeten *Konzepte und Repräsentationstechniken* für Präsentationswissen sind aber dennoch für HAMVIS relevant. Sie werden im nachfolgenden Abschnitt genauer besprochen.

Strukturierung des Konstruktionsraumes für Präsentationen

In Abhängigkeit vom Kontext und vom Schwerpunkt der Autoren wurden verschiedene Ansätze zur Modellierung eines Diskurses und zur Diskursplanung vorgestellt. Hovy verwendet die Begriffe *funktionalistisch* und *formalistisch* zur groben Charakterisierung der jeweiligen Grundansätze ([134], S. 344f.). Bei beiden Betrachtungsweisen wird ein Diskurs in Teile zerlegt (sog. Diskurssegmente), die jeweils „semantisches Material“ enthalten, das durch Segmentbeziehungen (Diskurssegmentrelationen) zueinander in Beziehung gesetzt wird. Für Funktionalisten sind die Ziele des Präsentierers und ihrer Beziehungen zu einem Rezipientenmodell (Benutzermodell) besonders relevant. Die interne Struktur ist als Hierarchie (Baum oder gerichteter Graph) aufgebaut. Knoten höherer Ebene repräsentieren Beziehungen zwischen Segmenten. Beziehungen werden in Hinblick auf die Wirkung eines Segmentes auf den explizit modellierten mentalen Zustand des Benutzers repräsentiert. Über das Rezipientenmodell wird der Planungsprozeß strukturiert. Diskurssegmentrelationen werden dazu verwendet, die Darstellungsmöglichkeiten einzuschränken. Bei verbaler Kommunikation werden bestimmte Beziehungen z.B. durch Schlüsselworte ausgedrückt (siehe [134], p. 362). Nach Meinung von Hovy haben funktionalistische Ansätze Schwächen bei der Behandlung von formalen Eigenschaften der Diskursstruktur (und des Diskursstrukturierungsprozesses). Auf diesem Gebiet haben dagegen formalistische Ansätze ihre Stärken (Anwendungsgebiete sind z.B. Pronominalisierung und Quantorengültigkeitsbereiche).

Die Grundidee der Einführung von Diskurssegmentrelationen war, daß es eine Menge von domänenübergreifenden Segmentbeziehungen gibt, die einen Diskurs strukturieren, so daß eine adäquate Informationsaufnahme durch den Rezipienten gewährleistet ist. Allerdings wurde noch kein Basisvokabular hierfür gefunden und es ist nicht klar, ob Grundstrukturen dieser Art existieren, die einerseits flexibel sind, aber andererseits auch auf der Präsentationsebene noch eine einschränkende Wirkung haben. Hovy klassifiziert verschiedenen Kategorien, die jeweils bestimmte Diskurs- und Präsentationsaspekte beeinflussen: semantische, interpersonelle und präsentationsbezogene Relationen

(Hovy [134] S. 362, siehe auch die Diskussion über die Charakterisierungen „parsimonious“ und „profligate“ für die verschiedenen Ansätze auf Seite 359f.).

Für die Definition von Diskursstrukturen gibt es zwei Basisansätze, die einander bei der Diskursplanung ergänzen: schemabasierte und planbasierte Ansätze.[62]

Schemata können als Makro-Operatoren oder Extraktionen von kommunikativen Strukturen interpretiert werden, die in vielen Präsentationen Verwendung finden. Die Wirkung von Teilsegmenten auf den mentalen Zustand des Rezipienten wird mit Schemata nicht explizit repräsentiert. Die Anwendbarkeit eines Schemas für eine kommunikative Situation muß also extern repräsentiert werden. Schemata können rekursiv definiert werden. Vielfach werden hierzu kontextfreie Grammatiken verwendet. Der Konstruktionsraum für Diskursstrukturen ist dann durch die Grammatik definiert. Schemata auf höherer Ebene werden als Startsymbol vorgegeben und solange ersetzt bzw. verfeinert bis nur noch elementare Schemata in der Struktur vorkommen. Elementare Einheiten (Terminale) stellen die Beziehungen zu dem zu kommunizierenden Material her. Die initial gegebenen Informationen (z.B. die Kurse einzelner Aktien für Börsenberichte) müssen in den Terminalen vorkommen, d.h. die Diskursplanung kann als Suchprozeß aufgefaßt werden. Die Aufgabe der Diskursstruktur ist es, die Informationen adäquat anzuordnen, so daß der Rezipient sie leicht aufnehmen kann (vgl. die Diskussion um Kohärenzkriterien). Dieses muß durch die Definitionen der Grammatikproduktionen sichergestellt werden. Zu beachten ist auch, daß sich aus der Schemaexpansion ergeben kann, daß zur kohärenten Darstellung noch weitere Informationen hinzuzufügen sind, damit die Erwartungen der Rezipienten erfüllt werden (z.B. für die Einleitung bei Börsenberichten Informationen über allgemeine Trends usw.).

Der planbasierte Diskurskompositionsansatz ist dazu gedacht, Ausgaben stärker auf die kommunikative Situation abzustimmen und deckt damit andere Anwendungsbereiche (s.o.) ab. Insbesondere bei tutoriellen Dialogen muß ggf. eine Kommunikation über die Kommunikation geführt werden können. Anstatt also wie bei Schemata einfach die Existenz eines Segmentes zu fordern, muß bei diesen Anwendungen auch noch repräsentiert werden, warum ein Teilsegment notwendig ist und welche Aufgaben es im Gesamtdiskurs übernimmt. Kommunikationsaktivitäten werden als rationales Handeln aufgefaßt, wobei Handlungen aus einer bestimmten Intention heraus ausgeführt werden und bestimmte Effekte (auf den Rezipienten) haben und damit die (kommunikativen) Ziele des Präsentierers erfüllen. Die kommunikative Situation ist dabei ein Einflußfaktor für die Ziele (bzw. Intentionen) des Präsentierers.

Kommunikative Basisakte („Äußerungen“) sind Komponenten von Akten höherer Ebene die wiederum ebenfalls zu Handlungen auf einer noch höheren Ebene zusammengefaßt werden können. (siehe z.B. [55] für eine Einführung in die in diesem Kontext relevante Sprechakttheorie). Aus intentionaler Sicht werden die Aktionen auf höherer Ebene auch als Pläne bezeichnet. Neben der Dekomposition sind Pläne durch Anwendbarkeitsbedingungen (oder Vorbedingungen) und Effekte (Nachbedingungen) gekennzeichnet. Je nach Anwendung können Vorbedingungen und Nachbedingungen jeweils noch weiter in Partitionen eingeteilt werden. Vorbedingungen und Nachbedingungen beziehen sich auf globale Speicher (z.B. zur Repräsentation des angenommenen mentalen Status des Rezipienten). Die Dialogplanung kann also als Verkettung von Planoperatoren dieser Art modelliert

62. Ich ignoriere hier feste Textstücke (canned text) oder feste Sequenzen von Präsentationen in Multimedia-Anwendungen. Dieses fällt eher in den Bereich Hypertext oder Hypermedia (für einen Überblick siehe [283]).

werden, so daß von einem Ausgangszustand (bedingt durch die kommunikative Situation) ein Zielzustand erreicht wird (STRIPS-Planung).

Schemata und Planoperatoren können sich ergänzen. Während Schemata für die Planung auf grober Granularitätsstufe geeignet sind, können planbasierte Ansätze für eine situationsangepaßte Kommunikation auf „niedrigerer“ Ebene verwendet werden. Aus einer theoretischen Perspektive betrachtet, verwenden planbasierte Ansätze zusätzliche Speichereinheiten während schemabasierte Verfahren nur den Keller für die kontextfreie Grammatik benötigen. Aus Sicht der Wissensrepräsentation werden bei der planbasierten Kommunikationsgestaltung zusätzliche Wissensquellen berücksichtigt.

Schemata sind so ausgerichtet, daß der Diskurs (oder der Dialog) so strukturiert wird, daß erstens die notwendigen Informationen kommuniziert werden und zweitens dieses in einer Form erfolgt, so daß der Rezipient diese Informationen leicht aufnehmen kann. Die Grenze zwischen den beiden Ansätzen ist als fließend zu betrachten. Inzwischen wurden Schemata auch mit Anwendbarkeitsbedingungen versehen.

Für HAMVIS sind Kommunikationsschemata oder -pläne zur Gestaltung der Oberfläche und der darin gezeigten Visualisierungen relevant. Für die Konzeption eines Designmodells bilden die Dialogstrukturen und die möglichen Beziehungen zwischen den Konstituenten das Basisvokabular. Im folgenden werden die Konzepte zur Modellierung der Dialogstruktur und zur Formulierung von Diskurssegmentrelationen von IMMPS geschildert und anschließend in den HAMVIS-Kontext übertragen.

SAGE

Das Präsentationsgenerierungssystem SAGE (siehe oben) wurde mit einem schemabasierten natürlichsprachlichen Generierungssystem (siehe die Arbeiten zu TEXT [206]) gekoppelt [266]. Um Text- und Graphikmodalitäten zu koordinieren, beginnt die Planung auf der Diskursebene (outline stage). Auf dieser Ebene werden Designdirektive für die Graphikplanungskomponente generiert. Daten und ihre Charakteristiken sowie auch Informationsverarbeitungsziele werden an SAGE delegiert. Designdirektive leiten sich aus der Notwendigkeit her, Graphiken und Texte aufeinander abzustimmen. Die Ordnung von Elementen in Bildern sollte z.B. auf die lineare Ordnung von Referenzpassagen im Text abgestimmt sein. Designdirektive werden in Form von Einschränkungen formuliert (z.B. Layouteinschränkungen für Teildarstellungen, Abbildung von Datenobjekten auf graphische Objekte, visuelle Notationen usw.).

Nachdem die graphischen Präsentationen erstellt wurden, wird mit der Textplanung fortgefahren. Um Redundanzen zu vermeiden, werden Diskurseinheiten, die schon durch graphische Darstellungen vermittelt werden, aus der Diskursstruktur entfernt. Graphiken werden in die Diskursstruktur integriert, indem Referenzierung und Aufmerksamkeitssteuerungseinheiten in die Diskursstruktur eingefügt werden, die dann durch zu generierenden Text umzusetzen sind. Obwohl durch diesen Ansatz nur eine grobe Integration von Text und Graphik erreicht wird, kann er für die Generierung von Börsenberichten und statistischen Kommentaren sehr wohl ausreichend sein.

Mayburys Ansatz

Ein weiterer Ansatz für die Komposition von Graphiken und deren Koordination mit Text wurde von Maybury entwickelt ([201], [202], [203]). Er erweiterte den Ansatz von Hovy zur Generierung natürlicher Sprache ([132]). Abbildung 44 skizziert die Definition eines Planoperators. Als Eingabe für den Planer dient ein kommunikatives Ziel wie `Identify(SYSTEM, USER, #<TOWN-1>)`. Alle Planoperatoren, deren Kopf-Feld (HEADER) sich mit diesem Ziel unifizieren lassen und deren Typeinschränkungen (CONSTRAINTS) gelten, sind mögliche Kandidaten zur Realisierung des Zieles. Die in einem Planoperator aufgeführten Variablen werden bei der „Operatoranwendung“ entsprechend instantiiert. Falls mehr als ein Planoperator anwendbar ist, so wird die Kandidatenliste, den folgenden Heuristiken folgend, sortiert (die letzteren mit geringerer Präzedenz):

- Bevorzuge Operatoren, die Text und Graphik verwenden.
- Bevorzuge Operatoren mit weniger Vorbedingungen.
- Bevorzuge Operatoren mit weniger Unterzielen („kognitive Ökonomie“).
- Bevorzuge Operatoren mit weniger Variablen (Vermeidung der Einführung weiterer Entitäten).
- Bevorzuge Operatoren, deren Vorbedingungen schon erfüllt sind.
- Bevorzuge Operatoren, die häufig verwendete Präsentationstechniken verwenden.[63]

Jeder Operator der geordneten Kandidatenliste wird der Reihe nach angewendet. Der erste Schritt besteht in der Überprüfung der Vorbedingungen. Hierzu wird je nach Ausprägung entweder das System- oder das Benutzermodell daraufhin betrachtet, ob die Vorbedingungen schon eingetragen sind, d.h. ob sie „gelten“. Falls dieses nicht der Fall ist, werden diejenigen Operatoren betrachtet und der Reihe nach expandiert, deren Effekt-Feld die entsprechenden Formeln erzeugen. Letzteres kann wiederum bedeuten, daß neue Ziele erfüllt werden müssen usw. Nachdem die Vorbedingungen eines Operators erfüllt wurden, werden die Anweisungen zum Aufbau der Diskursstruktur in der Dekomposition des Operators „ausgeführt“. Für die „Ausführung“ wird wieder nach Operatoren gesucht, deren Kopf-Feld zur Unifikation gebracht werden kann.

63. Dieses ist durch die lexikalische Ordnung der Operatordefinition in einer Datei repräsentiert

NAME	Explain-route-linguistically-and-visually
HEADER	Explain-Route(*S*, *H*, *from-entity*, *to-entity*)
CONSTRAINTS	Cartographic-entity?(*from-entity*) ∧ Cartographic-entity?(*to-entity*) ∧ *path*
PRECONDITIONS	visible(*from-entity*) ∧ WANT(*S*, KNOW-HOW(*H*, Go(*from-entity*, *to-entity*)))
EFFECTS	KNOW-HOW(*H*, Go(*from-entity*, *to-entity*)) ∧ ∀*segment* ∈ *path* KNOW(*H*, Subpath(*segment*, *path*))
DECOMPOSITION	∀*segment* ∈ *path* Indicate-Deictically(*S*, *H*, source(*segment*)) Command(*S*, *H*, Do(*H*, Go(source(*segment*), link(*segment*), destination(*segment*)))) Indicate-Deictically(*S*, *H*, link(*segment*)) Indicate-Direction(*S*, *H*, source(*segment*), link(*segment*), destination(*segment*)) Identify(*S*, *H*, *to-entity*)
WHERE	*path* = cartographic-path(*from-entity*, *to-entity*)

Abbildung 44. Beispiel für einen Planoperator der höheren Ebene für die Erklärung einer Route auf einer Karte (S steht für System, H für Hörer).

Abbildung 45 zeigt eine Skizze des Diskursbaumes, der durch den Beispielplanoperator in Abbildung 44 abgeleitet wurde. Die Effekt-Felder eines Operators modifizieren nicht nur das System- oder das Benutzermodell, sondern definieren Einschränkungen für die möglichen Präsentationsformen für die an Variablen gebundenen Objekte.[64]

Die treibende Kraft hinter dem Diskursplanungsverfahren mit Planoperatoren ist das Rezipientenmodell (Benutzermodell). Der Effekt von kommunikativen Akten, die durch Planoperatoren modelliert werden, wird durch Hinzufügen von entsprechenden Propositionen in eine entsprechenden Partition des Benutzermodells ausgedrückt (für ein Beispiel für eine Benutzermodellierungsumgebung mit einem Wissensrepräsentation in der KL-ONE-Tradition siehe Kobsa [154]). Partitionen sind in einer

64. Maybury demonstriert die Notwendigkeit, mehrere Operatorinstantiierungen mit unterschiedlichen Bindungen für „lokale“ Variablen zu erzeugen (die WHERE-Klausel des Operators in Abbildung 44 beinhaltet eine Generatorfunktion für mehrere Pfadalternativen). Effekt- und Dekompositionsfelder enthalten Allquantoren, so daß Aussagen für alle Elemente eines Pfades gemacht werden können. Damit sind Planoperatoren auch auf Objektmengen (beliebiger Kardinalität) anwendbar.

Hierarchie angeordnet, die durch eine Sichtbarkeitsbeziehung definiert ist. Partitionen werden dazu verwendet, Informationen darüber zu kodieren, was ein Kommunikationspartner wissen will oder schon weiß bzw. glaubt. Eine Partition, die von mehreren Vorgängern aus sichtbar ist, wird dabei als „geteilt“ bezeichnet und modelliert wechselseitige Glaubenszustände (mutual belief). Die theoretische Fundierung ist durch Modallogiken definiert.[65]

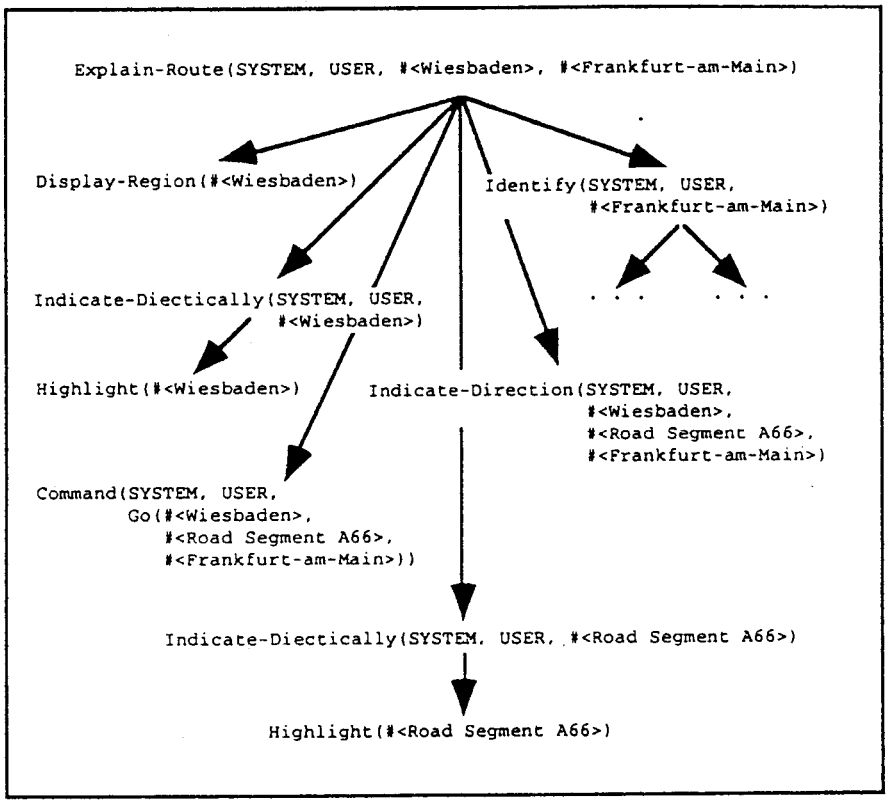


Abbildung 45. Instantiierte Dialogstruktur, aufgespannt durch die Dekomposition des Planoperators aus Abbildung 44.

Obwohl Benutzermodelle mit logischen Formalismen repräsentiert werden, wird in keinem der heute bekannten IMMPS eine Inferenzmaschine für logische Ableitungen eingesetzt, um z.B. die Präsentation von Informationen zu vermeiden, die der Rezipient ohne Mühe inferieren kann, oder um ungewollte Inferenzen (Implikaturen) durch Zusatzinformationen zu vermeiden. Propositionen werden bislang nur in den entsprechenden Partitionen „abgespeichert“.

In Mayburys Ansatz werden Planoperatoren höherer Ebene als anwendungsspezifisches Präsentationswissen formuliert. Basisoperatoren sind hingegen domänenübergreifend formuliert. Allerdings

65. Das Gebiet der Benutzermodellierung ist ein eigenes Forschungsfeld und kann hier nicht in voller Tiefe diskutiert werden. Für einen Überblick siehe [152], [153], [207]. Modelle des Glaubens, Wollens und Wissens und die Beziehungen zur Sprechakttheorie wird in dem klassischen Papier von Cohen und Perrault [55] aufgezeigt. Benutzermodelle für die Inhaltsplanung werden durch Zukerman et al. diskutiert (siehe [348] und [349]). Die Behandlung der intentionalen Operatoren von Maybury wird in [201] S. 62 detaillierter beschrieben (siehe auch die Definitionen von Intentionen nach Retz-Schmidt in [259] und [260]).

erlaubt es der Planungsansatz von Maybury bisher kaum, ein Modell für die Aufgabe zu berücksichtigen, die mit einer Präsentation unterstützt werden soll (siehe z.B. die Arbeiten an BOZ oder SAGE, die im Abschnitt „Automatisches Design von graphischen Präsentationen“ vorgestellt wurden). Die heuristischen Präzedenzregeln für die Anwendungsreihenfolge von Operatorenkandidaten nehmen keinen Bezug auf die tatsächlich generierten Präsentationen (Antizipationsrückkopplung). Wie sich aus den Diskussionen in vorigen Abschnitten ergibt, ist dieses auch ein schwieriges Unterfangen. Einige Teilaspekte wurden in anderen Systemen aber schon durch Rückkopplungsarchitekturen erfaßt. Eine Generiere-und-Teste-Architektur mit diesem Ziel wurde von Seligmann und Feiner für die automatische Generierung von Illustrationen für die Wartung und Reparatur von technischen Geräten vorgestellt (COMET/IBIS: [83], [84], [279], [280]).

COMET/IBIS

Als Erweiterung der Vorläuferarbeit zum APEX-System war es bei COMET/IBIS das Ziel, auch den Grund für die Darstellung eines Objektes explizit zu repräsentieren. Hierzu werden wiederum kommunikative Ziele verwendet. Kommunikative Ziele, die COMET/IBIS behandeln kann, sind: Anzeigen der Position eines physikalischen Objekts in einem Kontext, Anzeigen der relativen Position von zwei oder mehr Objekten mit Hilfe eines Kontextes, Eigenschaften von physikalischen Objekten sowie deren Zustandsänderungen. Kommunikative Ziele sind hier also objektzentriert und nicht Präsentator-zentriert formalisiert. Wie APEX, behandelt auch COMET/IBIS physikalische Objekte (Radioempfänger), es werden also keine abstrakten Visualisierungen aus gegebenen Daten und deren Charakteristiken sowie Informationsverarbeitungszielen des Rezipienten erzeugt (wie von BOZ oder SAGE). In IBIS wird jedes Objekt a priori aus einer bestimmten Sicht repräsentiert, es werden also durch IBIS selbst keine charakteristischen Sichten berechnet oder ausgewählt.

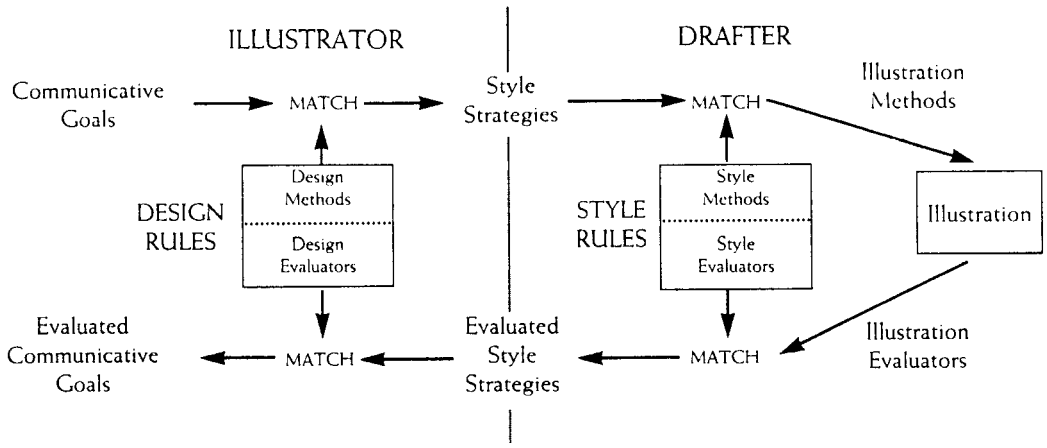


Abbildung 46. Funktionales Modell der Architektur des COMET/IBIS-Systems (aus [280], S. 127).

Die Präsentationsplanung ist als ein zweistufiger Prozeß modelliert, der hier in Abbildung 46 grob skizziert ist. In einem ersten Schritt werden kommunikative Ziele mit Hilfe von „Designmethoden“ auf „Stilstrategien“ abgebildet. Stilstrategien enthalten Beschreibungen für den Inhalt der Darstellungen und geben Hinweise (hervorheben, sichtbar, erkennbar) für die anschließende Wahl von visuellen Effekten (aufhellen, dämpfen). Wenn ein einzelner Effekt nicht ausreichend ist, werden mehrere Effekte kombiniert. Landmarken werden ähnlich wie in APEX behandelt (siehe oben).

Ob ein geplanter Effekt in der Graphik tatsächlich erreicht wurde, wird durch „Evaluierungsmethoden“ unter Zuhilfenahme von Schwellwerten überprüft (hierzu werden Standard-Techniken der Computergraphik eingesetzt). Die Werte von Schwellwerten hängen dabei von einem Wichtigkeitsattribut ab, das mit den Designregeln vergeben wird (`highest`, `high`, `medium`, `medium-low`, `low`). Falls die Bewertung eines speziellen visuellen Effektes unter den entsprechenden Schwellwert sinkt, werden andere Regeln verwendet.

Falls keine Regeln anwendbar sind, um bestimmte kommunikative Ziele zu realisieren, versucht IBIS, eine Darstellung in mehrere Teile zu zerlegen. Die Menge der Ziele wird in Partitionen zerlegt und der oben beschriebenen Prozeß startet von neuem, obwohl nach den Autoren von IBIS, die in der Praxis vorkommenden Hierarchien nie sehr tief oder breit sind.

In der mir zur Verfügung stehenden Literatur beschreiben die Autoren nicht, wie die Teilvisualisierungen zusammengesetzt werden (Plazierung und Größe von Darstellungen). Auch die Auswirkungen einer Zusammensetzung auf die Teildarstellungen ist nicht klar. IBIS modelliert nicht – oder nicht explizit – die Bedeutung der Teile für die Gesamtdarstellung.

WIP

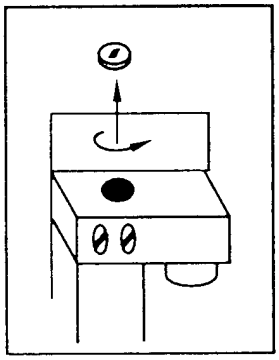
Die rhetorischen Beziehungen zwischen Teilen der Darstellung sind im System WIP [66]Ausgangspunkt sowohl für die Inhaltsbestimmung als auch für die Gestaltung der Präsentation ([330], [261], [3], [331]).

Das IMMPS WIP wurde für die automatische Erstellung von Bedienungsanleitungen für technische Geräte (Espressomaschinen, Rasenmäher, Faxgeräte) konzipiert. Mit WIP wurde eine Theorie zur agentenorientierten Präsentationsgenerierung entwickelt, mit der Graphik und Text kohärent miteinander verknüpft werden, um vorgegebene Präsentationsziele zu erfüllen. Die generelle Architektur kann ebenfalls als Instanz des Rahmenwerks aus Abbildung 43 gedeutet werden. Neben verschiedenen Wissensquellen besteht die Eingabe für WIP aus einer Menge von kommunikativen Zielen. Angeregt durch die Arbeiten zum Textverstehen und zur Generierung natürlicher Sprache von Mann et al. zur „Rhetorical Structure Theory“ (siehe [192], [193]) und die Operationalisierung dieser Theorie durch Hovy [132] und Moore [217], modellieren die in WIP verwendeten Planoperatoren rhetorische Beziehungen, die auch in handgefertigten (multimodalen) Illustrationen auftreten (vgl. auch die oben beschriebene Arbeiten von Maybury). Die kommunikativen Akte in der Dekomposition eines Operators werden in zwei Mengen eingeteilt (siehe Abbildung 47): Hauptakte (main acts) und Ergänzungsakte (subsidiary acts).

66. Ich verwendet hier zur Vereinfachung der Darstellung den Namen WIP auch als Stellvertreter für Nachfolgeprojekte (z.B. PPP).

```
[S1]  Name:
          Request-Enable-Motivate
      Header:
          (Request P A ?action T)[6]
      Effect:
          (BMB P A (Goal P (Done A ?action)))
      Applicability Conditions:
          (And   (Goal P (Done A ?action)) (Bel P (Complex-Operating-Action ?action))
                 (Bel P (Agent ?agent ?action)) (Bel P (Object ?object ?action)))
      Main Acts:
          (S-Request P A (?action-spec (Agent ?agent-spec) (Object ?object-spec)))
          (Activate P A (Action ?action) ?action-spec T)
          (Activate P A (Agent ?agent) ?agent-spec T)
          (Activate P A (Object ?object) ?object-spec T)
      Subsidiary Acts:
          (Motivate P A ?action ?mode-1)
          (Enable P A ?action ?mode-2)
```

Abbildung 47. WIP-Planoperator zur Modellierung einer rhetorischen Relation „Request-Enable-Motivate“ (P = presenter, A = actor).[67] (aus [3], Seite 8)



Remove the cover to fill the watercontainer

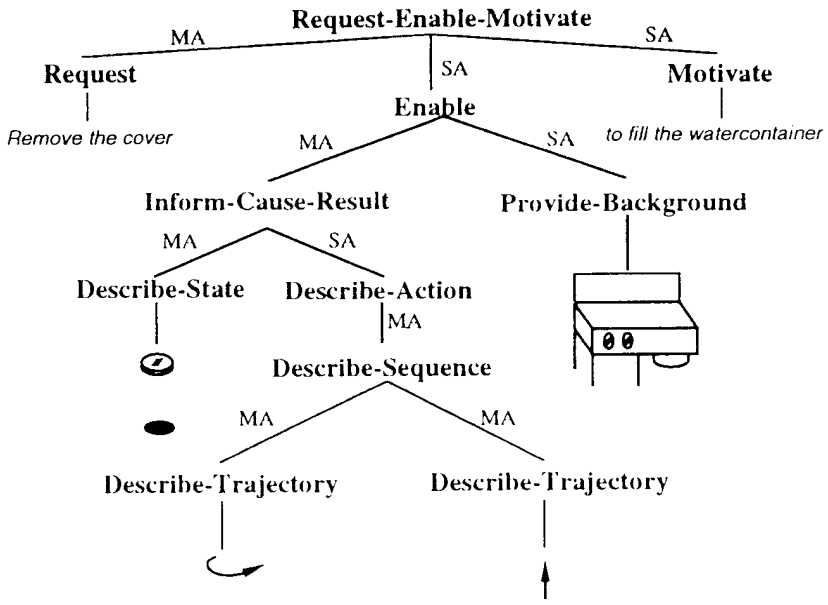


Abbildung 48. Expandierte Diskursstruktur for den Planoperator „Request-Enable-Motivate“ (MA = main act, SA = subsidiary act, aus [3], Seite 3).

67. (Bel P) bezieht sich auf die Glaubens-Partition des Präsentierers, (BMB P A ...) „adressiert“ eine Partition, die von der Benutzerpartition und der Präsentiererpartition geteilt wird. Die Notation ist angelehnt an Hovy [132], der sich wiederum auf frühe Arbeiten von Cohen and Levesque [54] bezieht (siehe aber Hovys Kommentare in [132] S. 87).

Planoperatoren werden in ähnlicher Weise wie bei Mayburys Ansatz expandiert. Ein Beispiel für eine Dialogstruktur, die durch den Planoperator „Request-Enable-Motivate“ aus Abbildung 47 aufgespannt wird, zeigt Abbildung 48.[68]

Rhetorische Relationen nach der RST-Theorie definieren einen Rahmen für die Formulierung von Planoperatoren, durch den die Bedeutung der Operatordekomposition auf höherer Ebene kenntlich gemacht werden kann. Planoperatoren können bzgl. der Klasse der Kohärenzkriterien, die sie unterstützen, kategorisiert werden (informationelle, interpersonelle und präsentation-bezogene Kohärenzkriterien). Die kommunikative Rolle der Einzelkonstituenten in Bezug auf den Gesamtdiskurs wird über die Beeinflussung des Benutzermodells explizit modelliert. Eine Unterscheidung zwischen Hauptakt (in den Originalarbeiten von Mann et al. [192] auch als Nucleus bezeichnet) unter Ergänzungsakt (Satellit) kennzeichnet die jeweiligen Konstituenten einer Dekomposition. Die Charakterisierungen werden dazu verwendet, Einschränkungen der Auswahlprozesse für Medien und Darstellungstechniken zu bestimmen.

In Abhängigkeit der Charakteristiken der zu kommunizierenden Informationen (die mit den Präsentationszielen gegeben sind), wählt WIP geeignete Medien und Modalitäten. Hierzu werden durch WIP Informationstypen charakterisiert: konkrete Informationen (mit visuellen Eigenschaften wie Form, Farbe, Textur), räumliche Information, zeitliche Information, kovariante Informationen, Quantifizierung, Negierung. In der Präsentationswissenbasis sind für diese Informationsarten geeignete Planoperatoren deklariert, die rhetorische Relationen realisieren, die für die Kommunikation dieser Informationsarten üblicherweise verwendet werden und den Erwartungen des Rezipienten entsprechen. Nach André und Rist ([3] p. 4ff.) werden folgende rhetorische Relationen betrachtet, die jeweils durch Bild oder Text ausdrückbar sind: Erzeugung von Aufmerksamkeit, Vergleichen, Herausarbeiten, Befähigen, Erläutern, Benennen, Motivieren, Beweisen, Vermittlung von Hintergrundinformationen, Zusammenfassen. Die hier widergegebenen natürlichsprachlichen Beschreibungen vermitteln nur einen Eindruck der Bedeutung der Relationen. Die Definition und Anwendung der rhetorischen Relationen erfolgt durch Planoperatoren.

Planoperatoren und die Dialogstruktur sind für die Verknüpfung der verschiedenen Wissensquellen zuständig. Anwendbarkeitsbedingungen für Planoperatoren sind in WIP nur Erfragefunktionen für die Datenbasis des Benutzermodell. Durch Anwendbarkeitsbedingungen können Schlußfolgerungsprozesse des Hintergrundsystems (siehe Abbildung 42) ausgelöst werden. Hierzu gehören: Subsumtionsinferenzen für Domänenpläne und -handlungen, Simulationen der Ausführung von Domänenplänen und temporale Projektion der Effekte (für eine detailliertere Darstellung siehe [331], S. 400f.).

Die Blätter der Diskurshierarchie werden auch Oberflächenakte (surface acts) genannt. Oberflächenakte definieren Einschränkungen für die Präsentation der assoziierten Informationseinheiten. Für jeden Informationstyp wird eine Menge von (Meta-)Regeln zur Festlegung von Präferenzen für Präsentationsarten und Modalitäten definiert (z.B. wird für räumliche Konzepte zuerst versucht, mit graphischen Techniken zu kommunizieren). Durch die explizite Priorisierung mit Regeln wird vermieden, daß z.B. die textuelle Reihenfolge der Definition der Planoperatoren in einer Datei ein Rolle spielt.

68. In der WIP-Terminologie wird die Diskursstruktur auch Dokumentplan genannt.

Auf der Präsentationsebene wird der Effekt eines Oberflächenaktes durch geometrische Datenstrukturen und Berechnungsalgorithmen evaluiert. Für Planoperatoren dieser Art, sieht WIP geometrische Berechnungsprozesse vor [261] (siehe auch die Ausführungen zu IBIS). Weiterhin wird ein inkrementeller Generierer für natürlichsprachliche Texte verwendet (z.B. auch zur Beschriftung von Graphiken).

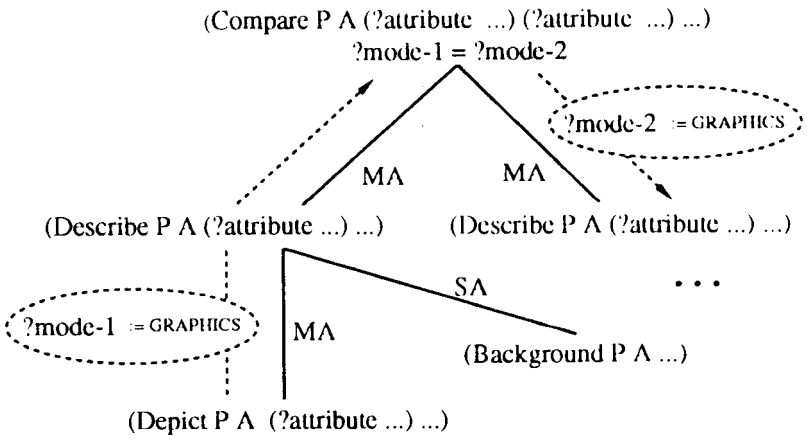


Abbildung 49. Auswahl von Modalitäten und Propagierung von Einschränkungen in der Dialogstruktur. (aus [3], Seite 13)

Der Präsentationsstrukturierungsprozess versucht, alle Einschränkungen zu erfüllen, die durch Oberflächenakte gegeben sind. Um z.B. Entscheidungen zur Wahl einer speziellen Darstellungsmodalität möglichst lange hinauszuzögern, verwaltet WIP auch hierfür zunächst einmal Einschränkungen (siehe Abbildung 49, in der ein `compare`-Operator die Gleichheit der Präsentationsmodalitäten von Unterakten fordert). Die Inferenzmaschine muß Rücknahmemechanismen verwalten, wenn der Planungsprozeß in eine Sackgasse läuft, d.h. keine Planoperatoren mehr anwendbar sind, obwohl noch nicht alle Blätter der Diskursstruktur durch Oberflächenakte ausgedrückt sind.

Auch eine vollständig expandierte Diskursstruktur kann aus Präsentationssicht noch „nachbearbeitet“ bzw. restrukturiert werden, um z.B. eine bessere Darstellung zu erreichen. Um z.B. eine übersichtliche Darstellung zu erzielen, können Teile der Diskursstruktur dupliziert werden (structure adding). Aus Gründen der Darstellung kann also Redundanz nicht immer vermieden werden.

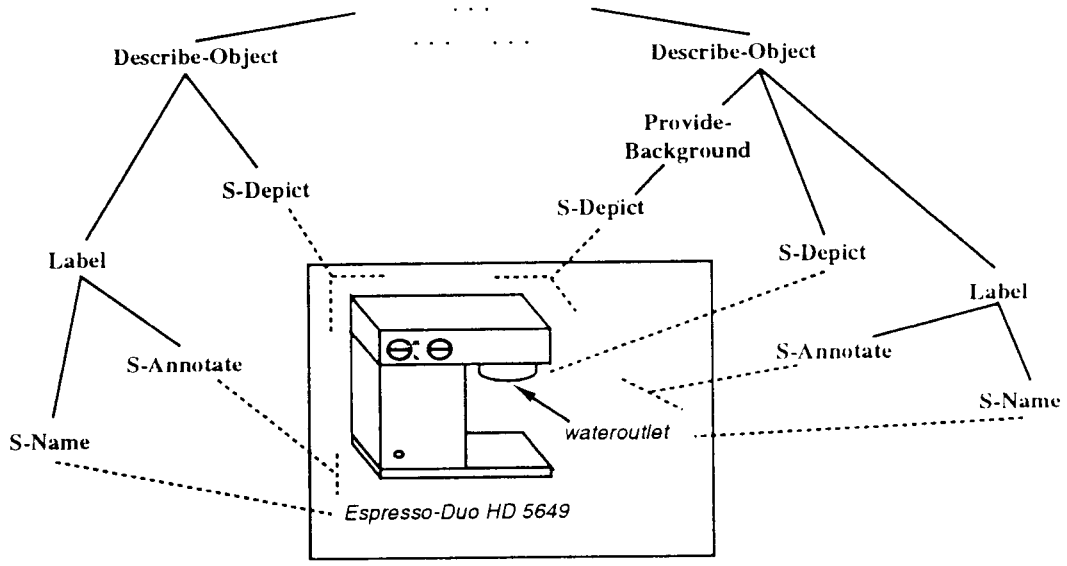


Abbildung 50. Ausnutzung von Präsentationsteilen für verschiedene kommunikative Ziele.

Nachdem Oberflächenakte auf der Präsentationsebene realisiert sind (nachdem also geometrische Datenstrukturen erzeugt worden sind), kann deutlich werden, daß bestimmte Domänenobjekte mehrfach in der Dialogstruktur referenziert werden. Abbildung 50 zeigt ein Beispiel, in dem das gleiche Präsentationsobjekt in mehreren Kontexten verwendet wird. In diesem Fall wird die Diskursstruktur restrukturiert und die Darstellungen werden verschmolzen (structure sharing).

Für weitere Details zum WIP-System möchte ich auf die Originalliteratur verweisen (z.B. auf die Dissertation von André [6]). Zur Repräsentation von Präsentationswissen und zur Entwicklung einer Antizipations-Rückkopplungsarchitektur auf der Basis von geometrischen Berechnungen siehe auch die Arbeiten von Rist [261].

Durch WIP (und auch durch Maybury) wurde gezeigt, daß die aus der Sprachgenerierung inspirierten Ansätze zur Modellierung von rhetorischen Relationen mit der Operationalisierung durch Planoperatoren auch eine Basis für die Generierung von multimodalen Präsentationen bilden. Auf der Architekturebene wird in WIP in ähnlicher Weise wie im COMET/IBIS-System ein Rückkopplungsmechanismus für Evaluierungen der konkreten Präsentation verwendet. Wir haben gesehen, daß von der Präsentationsebene (bzw. der Geometrieebene) Rückkopplungen auf die Dialogplanung möglich sind. In der konzeptuellen IMMPS-Architektur von Roth und Hefley in Abbildung 43 wird dieses durch gestrichelte Pfeile angedeutet. Die Behandlung von Rückkopplungsinformationen erfolgt in den verschiedenen Ansätzen (bedingt auch durch die unterschiedlichen Zielsetzungen) jeweils durch unterschiedliche Techniken. Bevor die Bedeutung der IMMPS-Systeme für die Konzeption von HAMVIS erläutert wird, gehe ich im nächsten Abschnitt zur Abrundung der IMMPS-Diskussion noch kurz auf die Kopplung von Dialog- und Präsentationsebene ein.

Verwaltung von Abhängigkeiten zwischen Diskurs- und Präsentationsebene: Modellierungs- und Kontrollaspekte

In den vorigen Diskussionen über den Konstruktionsraum für Präsentationen haben wir gesehen, daß die Diskursplanung (strategische Komponente) und die modalitätenbezogene Präsentationsplanung (taktische Komponente) in IMMPS wechselseitig voneinander abhängig sein können. Die Dialogstruktur ist die Hauptplanungsstruktur und wird während der Planungsphase expandiert, mit zusätzlichen Einträgen erweitert und ggf. partiell restrukturiert. Die Kommunikationsplanung kann durch die Verwendung von Schemata oder Planoperatoren als Suchprozeß interpretiert werden. Der Konstruktionsraum wird dabei allerdings auch durch Einschränkungen auf der modalitätennahen Ebene (z.B. durch geometrische Beziehungen bei graphischen Darstellungen) beeinflußt. Die Definition des Konstruktionsraumes und auch die Kontrolle beim Durchsuchen dieses Raumes sind noch umstritten und bilden ein offenes Forschungsfeld. COMET/IBIS sieht eine sequentielle Kontrollstruktur vor, die genauer betrachtet eine Tiefensuche realisiert. WIP hingegen verwendet verschiedene Prozesse, die jeweils für die Diskursplanung und für die Auswahl von Modalitäten auf der Realisierungsebene zuständig sind. Prozesse kommunizieren über die Dialogstruktur und durch die Bereitstellung von „Aufträgen“ (tasks) für jeweils nachgeschaltete Prozesse (inkrementelle Kaskaden-Architektur). Ein Prozeß braucht nicht auf die Bearbeitung eines Auftrags zu warten, sondern kann mit der Planung fortfahren, sofern die eigenen zu bearbeitenden Aufträge nicht von den Aufträgen abhängen, die der Prozeß an andere Prozesse delegiert hat.

Neben der Organisation des Kontrollflusses und der Behandlung von Abhängigkeiten ist die Kaskadenarchitektur auch durch Berücksichtigung von Ressourcenbeschränkungen motiviert. Wenn schon Teile der Präsentationsebene festgelegt sind, bevor die vollständige Diskursstruktur erstellt ist, so können Ausgaben früher erfolgen. In einige Fällen ist es notwendig, sehr schnell eine „Antwort“ zu produzieren, auch wenn das Ergebnis bei längerer Planung verbessert werden könnte.

Die Kaskaden-Architektur von WIP ist unter anderem auch durch kognitionswissenschaftliche Forschungsarbeiten motiviert, jedoch – wie andere Kaskaden-Architekturen auch – recht kompliziert. Zu beachten ist m.E., daß Computer andere Ressourcenbeschränkungen haben als Menschen. Daher sind kognitionswissenschaftliche Architekturen (siehe z.B. den Ansatz von Levelt [173] für eine Architektur zur Sprachproduktion), die als Modelle für menschliches Kommunikationsverhalten dienen, nicht unbedingt auch aus Informatiksicht zur Lösung eines Anwendungsproblems optimal. Leider ist das „Problem“ bei einem IMMPS schlecht gestellt, so daß Problemstellung und Lösung nicht klar getrennt werden können. Siehe auch den in Kapitel 2 diskutierten Begriff des „bösartigen Problems“ von Holz (in Anlehnung an Rittel, siehe [130]). Kognitionspsychologische Erkenntnisse können daher wesentliche Ideen für die Gestaltung von automatischen Präsentationsgenerierungssystemen liefern. Es ist allerdings ebenfalls sinnvoll, nach einer einfacheren Architektur zu suchen, die z.B. Zwischenstrukturen verwendet, die sich mit kognitionswissenschaftlichen Theorien nur sehr schwer in Einklang bringen lassen. In einem solchen „ingenieursmäßigem“ Ansatz schlagen Arens et al. vor, die Kommunikationsplanung auf der Diskursebene und die medien-spezifische Präsentationsplanung als eine Pipeline zu organisieren, um zu erreichen, daß die intentionale Struktur und die präsentationsorientierte, oberflächennahe Struktur strikt voneinander getrennt werden können. Die Präsentationsplanung wird als Prozeß verstanden, der die Diskursstruktur in eine Präsentationsstruktur umwandelt [12]. Sie weisen darauf hin, daß sie keine Notwendigkeit für eine Rückkopplung von der präsentationsorientierten Ebene auf die Diskursebene sehen. Ihr Ansatz geht davon aus, daß die Diskursstruktur

die „minimale Informationsmenge“ enthält, die nötig ist, um aus intentionaler und rhetorischer Sicht die Kommunikation zu gestalten. Während der medienspezifischen Präsentationsplanung können durchaus noch weitere Informationen hinzugefügt werden. Arens et al. sind der Meinung, daß für eine adäquate Präsentationsgenerierung die Diskursstruktur vollständig vorliegen muß. Sie formulieren die medienorientierte Phase der Präsentationsgenerierung als Bottom-up-Prozeß. Charakteristiken der zu übermittelnden Information sind einer der wesentlichen Faktoren für die Gestaltung der Darstellung, intentionale und rhetorische Informationen aus der Diskursstruktur werden schrittweise von unten nach oben bei der Darstellung berücksichtigt. Bislang wurde die vorgeschlagene Vorgehensweise an einem kleinen Prototypsystem genauer untersucht.

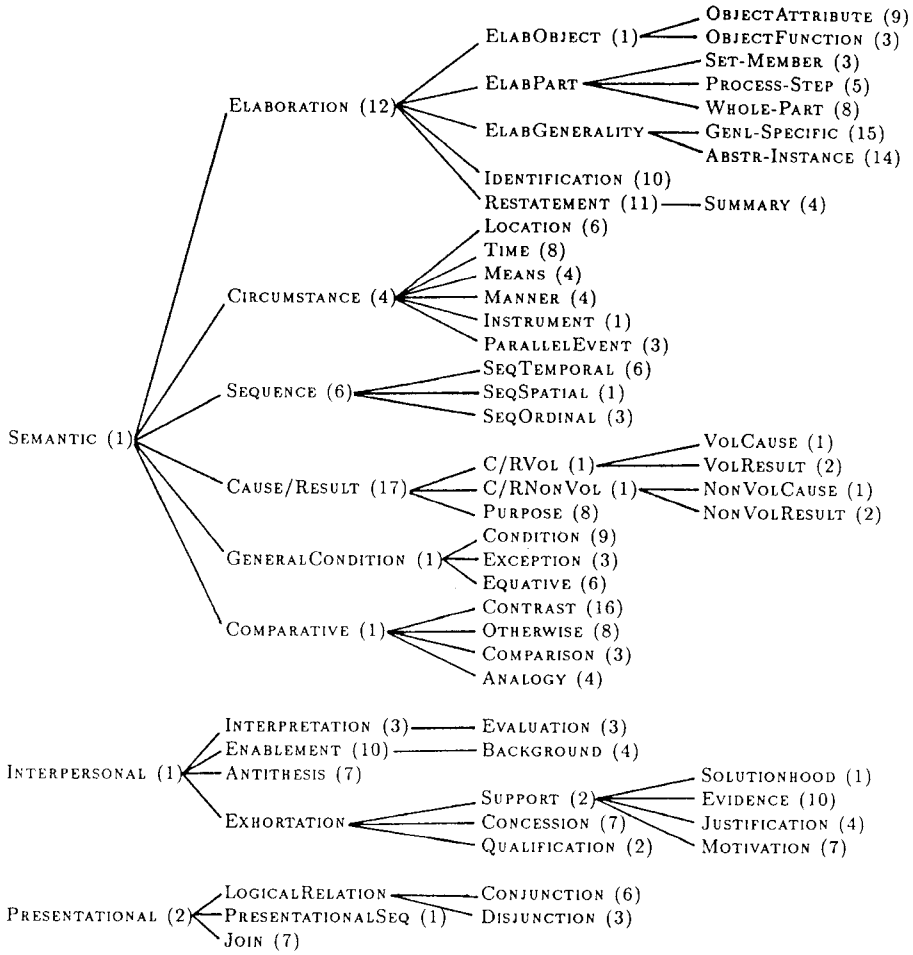


Abbildung 51. Eine Taxonomie von Diskurssegmentrelationen (aus [134], S. 377). Die Zahlen in Klammern charakterisieren die Anzahl der Veröffentlichungen, in denen die jeweiligen Relationen als wichtig hervorgehoben wurden.

Bei dieser Vorgehensweise spielen die Konzepte (bzw. Begriffe) zur Beschreibung der Diskursstruktur eine besondere Rolle. Hovy hat z.B. für die Charakterisierung von Diskurssegmentbeziehungen drei generelle Typen herausgestellt: informationelle (semantische), interpersonelle und präsentationsbezogene Relationen. Abbildung 51 zeigt eine Taxonomie von spezielleren Segmentrelationen für jeden Grundtyp. Semantische Relationen kennzeichnen, wie Domäneninformationen in einem spezifischen Kommunikationskontext zueinander in Beziehung gesetzt sind. Interpersonelle Relationen schränken

die Wahl der Präsentationstechniken durch Angaben über den jeweils beeinflußten mentalen Zustand des Rezipienten ein. Eine andere Gruppe von auf der Diskursebene relevanten Relationen (presentational relations) ist schon an Oberflächengesichtspunkten ausgerichtet. Durch diese Art von Relationen wird z.B. die Darstellungsreihenfolge schon auf der Diskursebene determiniert.

Für eine natürlichsprachliche Realisierung bieten sich zur Verdeutlichung der besonderen Bedeutung der Reihenfolge entsprechende Schlüsselworte an („als erstes“, „zuerst“, usw., siehe [134] S. 376ff. für detailliertere Beispiele). Die konkrete Realisierungsform an der Oberfläche hängt dann vom jeweils verwendeten speziellen Unterkonzept der Relation ab.

Wenn aus intentionalen oder rhetorischen Gründen in der Diskursstruktur schon eine starke Darstellungseinschränkung vorgesehen werden soll, so wird ein sehr spezielles Konzept für eine Diskurssegmentrelation verwendet. Weniger spezielle Konzepte lassen mehr Spielraum für die Darstellung.

Die Menge der Beschreibungskonzepte für Diskursstrukturrelationen hält Hovy für offen: Es gibt also nicht *das* Basisvokabular. Wenn deutlich wird, daß in einer speziellen Kommunikationssituation eine ganz besondere Darstellungsform benötigt wird oder üblich ist, wird die Taxonomie der Diskursstrukturrelationen erweitert. Die Präsentationsgenerierungskomponente wird dahingehend erweitert, daß für das neu hinzugefügte, speziellere Konzept die vorgesehene Darstellungsform gewählt bzw. auf weniger Alternativen eingeschränkt wird.

Die Betrachtung der IMMPS-Architekturen hat gezeigt, welche Beschreibungsformen für die Repräsentationsstrukturen auf der Diskurs- bzw. Dialogdesignebene bekannt sind. Im folgenden Abschnitt wird die Anwendung dieser Techniken im HAMVIS-Kontext besprochen.

Vergleich der Anforderungen von automatischen IMMPS und HAMVIS

Wie wir in der Einleitung gesehen haben, ist es für ein System wie HAMVIS wichtig, zwischen Entwicklungszeit- und Laufzeitaktivitäten zu unterscheiden. Zur Entwicklungszeit liegen (zum großen Teil) nur konzeptuelle Informationen vor. Für das Laufzeitsystem müssen ggf. entsprechende Fallunterscheidungen vorgesehen werden. Antizipationsrückkopplungsarchitekturen sind also für HAMVIS nur sehr bedingt interessant, da konkrete geometrische Objekte zur Entwicklungszeit nicht notwendigerweise bereitstehen. Zur Entwicklungszeit muß aber die Dialogstruktur schon aufgebaut werden, da viele Belange der Darstellung gleich beim Erzeugen des Darstellungsfensters für die Gesamtaktivität bekannt sein müssen (z.B. Anzahl und Typ der benötigten Teilfenster, Layout der Teilfenster). Die Dialogplanung kann aber auch mit konzeptuellen Informationen in ähnlicher Weise gestaltet werden, wie bei IMMPS-Systemen. Darstellungsteile haben eine bestimmte Aufgabe und beziehen sich aufeinander in spezieller Art und Weise. Diese wiederum hat Auswirkungen auf die Darstellung usw.

Für HAMVIS ist es notwendig, verschiedene Varianten der Dialogstruktur zu erstellen. Diese können dann von den an der Systementwicklung beteiligten Personengruppen diskutiert und bewertet werden. Ein inkrementeller Generierungsansatz, der beim System WIP realisiert wurde, würde im Prinzip jeweils schon bestimmte Teilaspekte der Dialogstruktur in Präsentationsstrukturen umwandeln, wäre also auch hier interessant. Zu beachten ist aber, daß die Gefahr groß ist, daß Abhängigkeiten zwischen den an der Generierungskaskade beteiligten Prozessen von den Designern nicht mehr nachvollzogen und verstanden werden könnten. Um die Interaktion der Designer mit dem HAMVIS-System zu vereinfachen, habe ich mich daher für einen Pipelining-Ansatz entschieden, der an das von Arens, Hovy und anderen vorgeschlagene sequentielle Schema angelehnt ist. Die Konzepte zur Beschreibung von

Diskursstrukturen von IMMPS werden auch in HAMVIS zur Beschreibung der Dialogstruktur einer Anwendung verwendet. Die Dialogstruktur wird schrittweise aufgebaut. Die Aufgabe von HAMVIS ist die Verwaltung von Varianten. HAMVIS sucht automatisch nach Möglichkeiten, die Diskursstruktur zu vereinfachen (z.B. durch Mehrfachverwendung von Referenzsystemen, siehe die Diskussion in der Einleitung und vgl. die Arbeiten zu WIP bzgl. des „Structure Sharing“). Wenn HAMVIS eine solche Variante bestimmen kann, wird sie dem Designer als mögliche Variante durch HAMVIS präsentiert (HAMVIS stellt zum Design der Dialogstruktur eine interaktive Oberfläche bereit). Der Designer kann dann entscheiden, ob er die Variante für geeignet hält, weiterentwickelt zu werden, oder ob er sie verwirft. Damit diese Entscheidungen durchführbar sind, muß das Dialogmodell so einfach wie möglich gehalten werden. Nach meiner Einschätzung sind Kaskadenarchitekturen für diesen Einsatzzweck zu komplex.

Um Diskursstrukturierung und Präsentationsgenerierung zu entkoppeln, erscheint es sinnvoll, zunächst beim Aufbau der Diskursstruktur eventuell die Darstellung von Informationen einzuplanen, die dann aus Präsentationsgründen nachher fortgelassen werden. In der Diskursstruktur ist die kommunikative Funktion einer Einheit vermerkt (z.B. als Ergänzungsakt, subsidiary act). Zu beachten ist auch, daß in einer interaktiven Anwendung bestimmte Informationen erst auf Anfrage präsentiert werden können. Teile können also aus Gründen der Übersicht zunächst zurückgehalten werden (siehe entsprechende Interaktionstechniken zur „Auffaltung“). Die Interaktion des Benutzer mit den generierten Graphiken wurde in jüngster Zeit auch von IMMPS als relevant betrachtet.

2.4.4 Unterstützung von Interaktionstechniken in Präsentationen

Jüngere Entwicklungen im IMMPS-Bereich stellen Erweiterungen der konzeptionellen Architektur aus Abbildung 43 bereit, um Benutzeraktionen auf graphischen Objekte, die in Präsentationen gezeigt werden, zu unterstützen. André und Rist sprechen von passiver und aktiver Betrachtung [3] und beschreiben, wie die WIP-Architektur erweitert werden kann, um graphische Objekte explorierbar zu machen. Diese Funktionalität wird über das Konzept der benutzerinitiierten Präsentationsziele in die WIP-Generierungsarchitektur integriert. Bei der dynamischen Vorgehensweise von WIP wird kein statisches Layout erzeugt, sondern die Präsentation soll den Informationsbedürfnissen angepaßt werden, während die Interaktion fortschreitet. Das System IBIS wurde in ähnlicher Weise um Möglichkeiten zur Navigation, Bewegung von Objekten usw. erweitert [280]. Auch hier werden Benutzeraktionen interpretiert als Hinzufügung neuer Präsentationsziele bzw. neuer Einschränkungen, die durch nachfolgende, von IBIS dynamisch zu generierende Präsentationen, erfüllt werden sollen.

Durch Benutzeraktionen können demnach in dieser Konzeption von WIP und IBIS die Auswahl und die Darstellungsformen von präsentierten Informationen beeinflußt werden. Allerdings werden keine Aktionen auf Domänenebene wie z.B. die Lokalisierung einer Küche unterstützt. Die von HAMVIS betrachteten Benutzeraktionen bedürfen einer direkten Rückkopplung bei der Darstellung. Für die Verschiebung eines Objektes müssen z.B. die möglichen Freiheitsgrade der Position bekannt sein, damit entsprechende Rückkopplungsmechanismen bereitgestellt werden können. Während der Interaktion (z.B. während einer Ziehen-und-Fallenlassen-Geste) ist es kaum sinnvoll, eine komplexe Planungsmaschinerie mit Präsentationszielen zu verwenden, auch nicht bei einer inkrementellen Generierungsarchitektur.

2.4.5 Diskussion

In diesem Abschnitt haben wir die agentenorientierte Sicht auf Sprache (bzw. Kommunikationsmodalitäten) unter dem Thema „Intelligente Multimedia-Präsentationssysteme“ diskutiert. Es wurde deutlich, daß Ausgaben gewissen Konventionen genügen müssen und bestimmte Kohärenzkriterien erfüllen sollen, um den Erwartungen des Rezipienten gerecht zu werden. Anstelle einer reinen „Kodierung“ von Informationen wie z.B. noch bei APT tritt in IMMP-Systemen ein Schlußfolgerungsprozeß über die Konstituenten der Kommunikation und deren Wirkung auf den mentalen Zustand des Rezipienten. Der Kern eines IMMPS ist die Diskursstruktur zur Modellierung der kommunikativen Funktion von Konstituenten (intentionale und attentionale Aspekte) und deren rhetorische Relationen. Diskurssegmentrelationen bilden das Gerüst für die Definition von Schemata und Planoperatoren. Die Diskussion der verschiedenen Ansätze zur Konzeption eines IMMPS hat gezeigt, daß jedoch über die genaue Ausprägung der IMMPS-Architektur noch kontroverse Meinungen bestehen. Je nach Zielsetzung des Systems wurden unterschiedliche Strategien vorgeschlagen zur Berücksichtigung von Abhängigkeiten zwischen der strategischen Komponente, die für den Aufbau der Dialogstruktur sorgt, und der taktischen Komponente, die für die konkrete Präsentation zuständig ist.

Ähnlich wie schon in den Präsentationsgenerierungssystemen aus dem Abschnitt „Automatisches Design von graphischen Präsentationen“ (BOZ, SAGE) dienen die Präsentationen, die durch IMMPS generiert werden, einem kommunikativen Ziel (definiert als logische Operatoren oder Informationsverarbeitungsziele). In IMMPS hingegen sind kommunikative Ziele aus der Perspektive eines Präsentationsagenten repräsentiert, der Schlußfolgerungen über die Glaubens- und Wissenszustände des Rezipienten durchführt. Die Hauptidee von IMMPS ist nicht nur, den Benutzer in die Lage zu versetzen, bestimmte Aktionen durchzuführen, sondern auch seinen „mentalen Zustand“ entsprechend zu beeinflussen. Kommunikation wird damit als Handlung aufgefaßt. Das Wissen, welche Effekte bestimmte Sprechakte (oder besser Kommunikationsakte) auf das Rezipientenmodell haben kann, wird implizit durch Schemata und in expliziter Form durch Planoperatoren erfaßt.

Zur Repräsentation von Kommunikationswissen mit Schemata oder Planoperatoren wurden verschiedene Sprachen entworfen. Operatoren werden sowohl für (domänenspezifische) Kommunikationsakte höherer Ebene als auch für generische, oberflächennahe Akte verwendet. Durch Oberflächenakte wird die Verbindung zu den Kommunikationsmedien und -modalitäten hergestellt. Auffällig ist aber, daß in der Definition von Planoperatoren immer wieder Einschränkungen für Objekte auftreten wie „sichtbar“, „identifizierbar“, „erkennbar“ usw. Da dieses generelle Anforderungen sind, scheint es mir günstiger, einen generelleren Mechanismus vorzusehen, durch den diese Bedingungen realisiert werden, anstatt bei jeder Planoperatoranwendung wieder über komplexe Rückkopplungsprozesse darauf achten zu müssen, daß vorherige Präsentationsziele immer noch erreicht werden. Es wäre vorteilhaft, wenn Domänenmodelle so formuliert werden könnten, daß z.B. „Überlappungsfreiheit“ schon auf der Konzeptebene zugesichert werden kann, und nicht nach jeder Planoperatoranwendung wieder durch komplexe geometrische Berechnungsprozesse überprüft werden muß.

Wir haben in diesem Kapitel gesehen, daß bei Verwendung von direktmanipulativen Interaktionsformen die Sichtweise der Mensch-Computer-Interaktion als Dialog unter Verwendung der Kommunikationsmetapher umstritten ist. Bei einem (objektorientierten) Zeichenprogramm zum Beispiel ist der Charakter der direkten Manipulation so ausgeprägt, daß eine Deutung der Interaktion als Kommunikation (mit entsprechenden Kommunikationszielen usw.) eher fern liegt. Nicht alle Anwendungen mit direkter Manipulation lassen sich jedoch sinnvoll als „Werkbank“ deuten, die der Benutzer selbst

organisiert. Gerade wenn dem Benutzer Objekte für bestimmte Aktionen präsentiert werden, erfolgt die methodische Auswahl der Objekte und auch die Wahl der Darstellungsformen durchaus nach Kriterien, die sich intern aus Sicht einer zielgerichteten Kommunikation gut beschreiben lassen, auch wenn diese Deutung extern vielleicht für den Endbenutzer irrelevant oder nebensächlich ist. Wichtig in diesem Zusammenhang ist die Organisation des Kompositionsraumes für Visualisierungen. Die kommunikative Funktion eines einzelnen „Bausteins“ sowie die Beziehungen zwischen den Einzelelementen müssen explizit gemacht werden, so daß Ableitungen und Schlußfolgerungsprozesse definiert werden können. Hierdurch wird der Kompositionsraum strukturiert und die Wahl der Darstellungsattribute eingeschränkt.

Zum Entwurf und zur Strukturierung von Visualisierungen und ihrer Einbettung in eine Oberfläche werden zur Entwicklungszeit ähnliche Konzepte benötigt, wie sie IMMP-Systeme zum Aufbau der Diskursstruktur bereitstellen. Die zu kommunizierenden Informationen sind durch eine anwendungsspezifische Aktionenmodellierung gegeben, müssen aber noch strukturiert und zueinander in Beziehung gesetzt werden. Hierzu lassen sich die Diskursstrukturierungskonzepte einsetzen, die in IMMP-Systemen entwickelt wurden. Sie bilden das Gerüst für die Arbeit des Designers. IMMP-Systeme zeigen, daß mit Konzepten wie Diskursstruktur, Diskurssegmentrelationen, eine automatische Generierungsarchitektur für Präsentationen realisiert werden kann, mit der die für die Präsentationsgenerierung notwendigen Wissensquellen formalisiert und verknüpft werden können. Es wurde durch IMMPS-Forschungsarbeiten gezeigt, daß die Grundannahme einer kommunikationsorientierten Betrachtungsweise des Konstruktionsraumes sinnvoll ist, auch wenn der Benutzer nicht einen Dialog im klassischen Sinne führt (z.B. über natürliche Sprache). Eine der Ideen von HAMVIS ist, diese Techniken auch für ein Unterstützungssystem für die Entwurfsarbeit in einem Szenario einzusetzen, in dem zu präsentierende Objekte nicht unbedingt als Instanzen bekannt sind. Die Grundstruktur der Oberfläche soll aber schon zur Entwicklungszeit der Anwendung erfolgen, d.h. zu einem Zeitpunkt, in dem meist nur konzeptionelle Informationen bekannt sind.

Durch die Arbeiten von Suthers ([301], [302]) im Bereich der Generierung von Erklärungen wurde eine neue Sicht auf Kommunikationsaktivitäten entwickelt. Suthers beschreibt das Kommunikationsverhalten in Erklärungssituationen als „interaktive Modellkonstruktion“. Er geht davon aus, daß von den Dialogpartnern während der Kommunikation Repräsentationen für Modelle (von Domänenobjekten und deren Relationen) aufgebaut werden. Anstatt also den mentalen Zustand als eine Menge von Fakten zu betrachten, werden hier die Informationen strukturiert. Das Ziel einer Erklärung ist, ein Modell aufzubauen bzw. die Differenz zu einem Modell zu minimieren. Diese Überlegung läßt sich auch auf die Generierung und Komposition von Visualisierungen übertragen. Einer Visualisierung liegt ein ganz bestimmtes Modell der Domänenobjekte zugrunde. Die Aufgabe des Systementwicklungsteams besteht darin, mit den durch HAMVIS bereitgestellten Basiskonstrukten, solche Modelle zu erstellen. Im Gegensatz zu IMMPS-Ansätzen kann im HAMVIS-Kontext während des Entwurfs deutlich werden, daß die bereitgestellten Modelle für Domänenwissen noch aktualisiert bzw. angepaßt werden müssen. Anstatt also von der Möglichkeit auszugehen, Domänenwissen sei adäquat präsentiert und müssen „nur“ noch kommuniziert werden, geht das HAMVIS-Szenario (vielleicht etwas pessimistischer bezüglich der Möglichkeiten der KI-Technologie) davon aus, daß aufgrund von Anforderungen der Kommunikation das Domänenwissen entsprechend erfaßt und – nach der Idee von Suthers – in Modellen strukturiert werden muß.

Die Flexibilität, die ein IMMPS bereitstellt, wird für interaktive Oberflächen nicht in jedem Fall benötigt. Die für Anwendungen wie z.B. XKL zu entwerfenden Oberflächen und Visualisierungen sind für ganz bestimmte Aufgaben vorgesehen (im Gegensatz zu Oberflächen zur Informationsrecherche). Die zu unterstützenden Benutzeraktionen sind durch die Aktionenzerlegung a priori bekannt. Daher ist es sinnvoll, die in der Oberfläche zu präsentierenden Visualisierungen in einer vorangehenden Planungsphase genau auf die Erfordernisse der Anwendung abzustimmen. Hierdurch kann auch der Berechnungsaufwand zur Benutzungszeit der Oberfläche minimal gehalten werden. Eine A-priori-Bewertung einer Oberfläche wird dadurch ebenfalls ermöglicht. So scheint es z.B. für Leitstände in Kraftwerken oder Kontrolleinrichtungen für technische Prozesse günstig zu sein, bei Störfällen die Informationen möglichst gut auf den jeweiligen Systemzustand abzustimmen (Redundanzvermeidung) und weiterhin möglichst schnell Informationen zu präsentieren (Reaktivität). Hierfür wäre eventuell eine inkrementelle Generierungsarchitektur, wie WIP sie bietet, einzusetzen. Andererseits ist die situative Informationsdarstellung vielleicht für den Operateur ungewohnt, und in einer Gefahrensituation können Darstellungen von Teilinformationen zu vorschnellen Entscheidungen führen. Für sicherheitsrelevante Anwendungen scheinen mir IMMPS-Ansätze nicht unproblematisch. Eine Überprüfungsmöglichkeit der Gesamtapplikation für einen menschlichen Systemdesigner ist hier notwendig. In diesem Kontext ist die von HAMVIS vorgenommene Trennung von Entwicklungszeit und Laufzeit sehr vorteilhaft.

Wichtig ist auch, daß die notwendigen Einschränkungen an die Graphik (auch) durch die zu unterstützenden Aktionen motiviert sind. Falls ein graphisches Objekt (durch einen Planoperator) in die Präsentation integriert wird, so wird durch aktuelle IMMPS-Ansätze angenommen, daß der Benutzer dieses Objekt „kennt“, d.h. daß er es gesehen und „aufgenommen“ hat und dieses graphische Objekt mit dem korrespondierenden Anwendungsobjekt in Beziehung gesetzt und die relevanten Merkmale zugeordnet hat. Mag dieses bei linearem Text noch eine brauchbare Annahme sein, so ist bei Graphiken eher zweifelhaft, ob das einmalige Darstellen eines Objektes ausreicht. Im Gegenteil, nach der Ausführung einer Handlung können die nicht von der Handlung betroffenen Objekte nicht einfach weggelassen werden, weil im Rezipientenmodell vermerkt ist, daß der Rezipient sie „kennt“. In interaktiven Graphiken gelten also andere Gesetze. Hier scheint mir bei IMMPS-Ansätzen noch Forschungsbedarf vorzuliegen. Das Rezipientenmodell wird auf der Dialogstrukturebene verwaltet, was ein Betrachter aufnimmt (perzipiert) hängt aber offensichtlich von der gewählten Darstellungsform ab. Das Feld der intelligenten Multimedia-Präsentationssysteme ist ein interessantes, aber noch offenes Forschungsfeld. Durch diesen Abschnitt wurde aufgezeigt, welche Aspekte für HAMVIS relevant sind.

3
Entwurf von Visualisierungen mit HAMVIS

Eingebettet in die Gesamtentwicklung einer Anwendung gliedert sich die Erstellung von Visualisierungen für eine interaktive Oberfläche in mehrere Phasen, die gegebenenfalls mehrfach durchlaufen werden (siehe Abbildung 4). Dieses Kapitel beschreibt die Konzeption der von HAMVIS bereitgestellten Dienste und zeigt auf, welche Modellierungswerkzeuge für die einzelnen Teilphasen für den HAMVIS-Benutzer bereitgestellt werden. Im Vordergrund dieser Arbeit stehen die Modelle für Aktionen und Dialogstrukturen und das Zusammenspiel der vordefinierten Wissensbasen mit speziellen, für die jeweilige Anwendung aufzustellenden Modellen. Anhand der prototypischen Implementierung wird diskutiert, wie die Arbeit des Oberflächenentwicklers mit HAMVIS aussieht und wie die in HAMVIS integrierten interaktiven Oberflächen den Entwickler in den jeweiligen Entwicklungsphasen unterstützen. Die vorgestellten interaktiven Oberflächenkomponenten von HAMVIS sind als Vorschläge zu verstehen. Im vorigen Kapitel wurde allerdings schon herausgestellt, daß automatische Ableitungen über diesen Modellen so gestaltet und verwaltet werden müssen, daß sie dem HAMVIS-Benutzer kommuniziert werden können und er mit den abgeleiteten Repräsentationen agieren kann. Die „Kommunizierbarkeit“ von Informationen innerhalb einer interaktiven Oberfläche prägt also nicht nur die Gestaltung der mit HAMVIS entworfenen Anwendungen, sondern auch die Konzeption und Gestaltung von HAMVIS selbst.

Ein Überblick über das Gesamtsystem und über die Generierung von Visualisierungen mit HAMVIS wurde schon in der Einleitung gegeben. Abbildung 52 faßt noch einmal die wesentlichen Punkte zusammen. In der linken Säule stehen die von HAMVIS bereitgestellten Teilmodelle (Grundmodell, Aktionenmodell, Dialogstrukturmodell, Markierungsmodell, UIMS-Erweiterungen). In der Mitte sind die Aktionen des Oberflächenentwicklers zusammen mit den in den jeweiligen Phasen (unter Bezugnahme auf die HAMVIS-Modelle) definierten anwendungsspezifischen Teilmodelle aufgezeigt. Rechts davon sind in einer weiteren Säule (wiederum grau hinterlegt) die in den Teilschritten jeweils benutzten Dienste von HAMVIS skizziert. Konsistenzprüfungen für Modelle und die Ableitung von impliziten Informationen wurden direkt unter Rückgriff auf die verwendete Beschreibungslogik realisiert. Die anderen Dienste stellen eine Erweiterung der beschreibungslogischen Basisinferenzen dar. Das UIMS-Fundament ist in hellerem grau im unteren Teil der Abbildung eingezeichnet. Es besteht aus anwendungsübergreifenden Erweiterungen und einem anwendungsspezifischen Teil, der aus den Modellen, die der Oberflächenentwickler definiert hat, automatisch generiert wird.

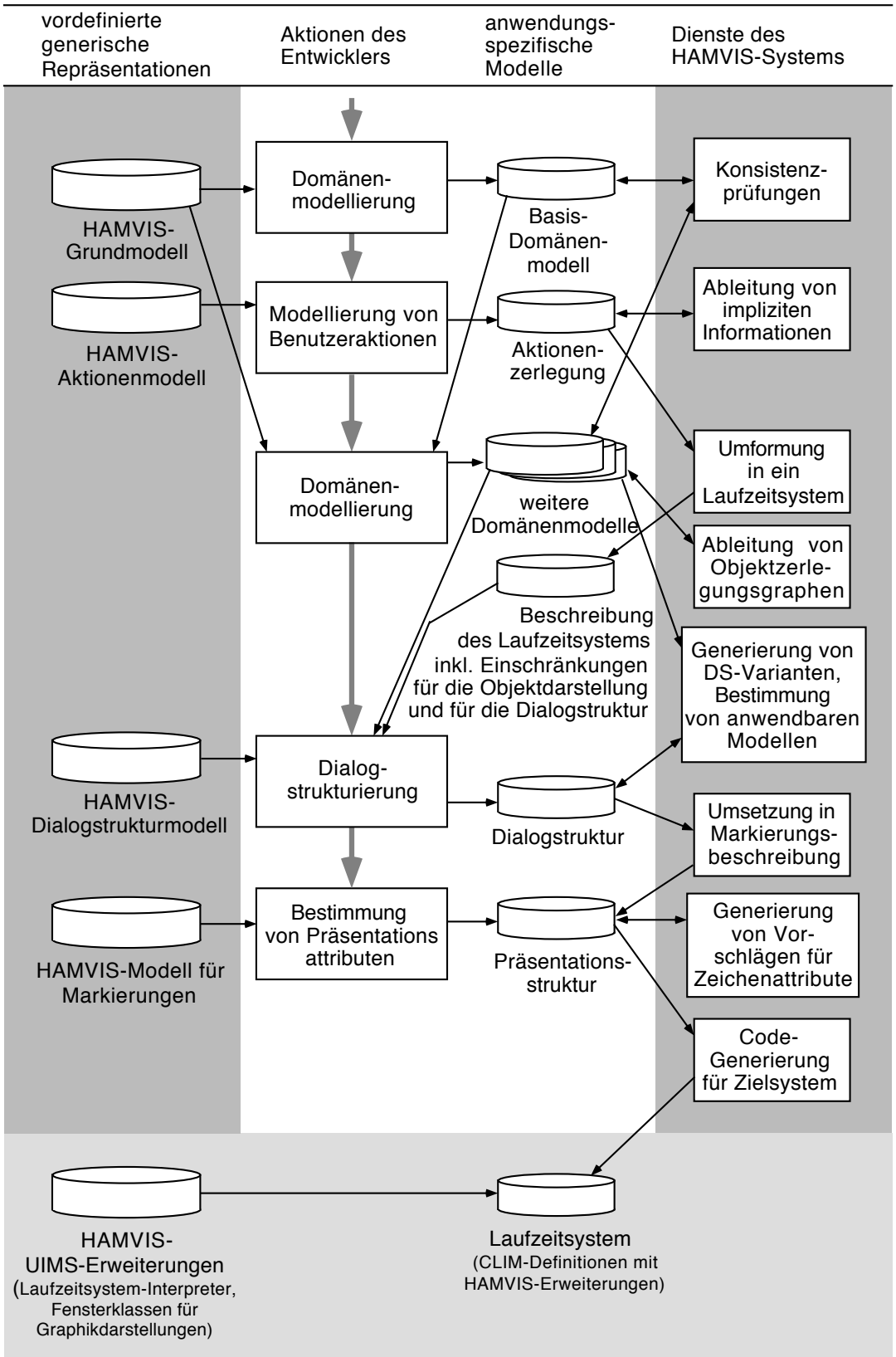


Abbildung 52. Überblick über das Gesamtsystem.

Ich werde in den nachfolgenden Unterkapiteln die jeweiligen Modellierungs- und Repräsentationsprobleme in den Teilphasen aus Abbildung 52 zunächst einmal auf der Wissensebene grob skizzieren und natürlichsprachlich umreißen. Im Anschluß daran werden dann in den jeweiligen Unterabschnitten die gewählten technischen Realisierungen beschrieben. Ich versuche dabei, z.B. mit Hilfe der Beschreibungslogik, Ableitungen formal zu beschreiben. Unumgängliche Einbettungen von objektorientierten Datenstrukturen und Algorithmen z.B. für die Verwaltung von geometrischen Daten lassen sich wiederum auf der Logikebene deuten.

Als Zusammenfassung der Teilkapitel werden die Möglichkeiten der technischen Realisierung wieder auf der Wissensebene gedeutet. Mit den bei der Vorstellung der technischen Lösung eingeführten Begrifflichkeiten kann das jeweilige Teilproblem auf der Wissensebene präzisiert werden, bzw. es kann abgeschätzt werden, was überhaupt repräsentierbar ist.

3.1 Vordefinierte und anwendungsspezifische Modelle

Wie schon in der Einleitung erläutert, steht die Modellierung des für den Anwendungskontext relevanten Domänenwissens am Anfang der Entwicklung einer Anwendung. Dieses gilt offensichtlich auch für Anwendungen mit visuellen Benutzungsoberflächen. In dem Entwicklungsszenario aus Abbildung 5 wird diese Aufgabe durch das „Systementwicklungsteam“ übernommen. Ich gehe dabei von der Prämisse aus, daß die Oberfläche nicht als „Anhängsel“ im Nachhinein auf die automatischen Programmteile der Anwendung aufgesetzt werden soll, sondern gleichberechtigt zusammen mit der Anwendung entworfen werden muß (siehe die Diskussion in Kapitel 2.2). Modelle für Anwendungsobjekte müssen also sowohl für die Implementation von Anwendungsfunktionen als auch für Visualisierungsbelange geeignet sein.

3.1.1 Modelle auf der Wissensebene und ihre Rolle bei der Visualisierungsgenerierung

Um eine Wiederverwendbarkeit von vordefinierten generischen Aktionen- und Dialogstrukturmodellen sicherzustellen, sieht die Konzeption von HAMVIS ein Grundmodell vor, mit dem es möglich ist, Modelle für Anwendungs-, Präsentations- und Interaktionswissen nicht nur für eine spezielle Anwendung, sondern für eine ganze *Klasse von Anwendungen* bereitzustellen (*HAMVIS-Grundmodell*). Dieses Grundmodell muß dann durch spezielle Domänenmodelle ergänzt werden, die durch das Systementwicklungsteam für die jeweilige Anwendung aufzustellen sind. HAMVIS greift hier den Ansatz von Bateman et al. auf, die mit ihrem „Upper Model“ einen ähnlichen Ansatz im Bereich der Sprachgenerierung verfolgen ([18], siehe auch die Arbeiten zum System Penman [131]). Das HAMVIS-Grundmodell ist jedoch kleiner und auf die Erfordernisse der Visualisierungsgenerierung ausgerichtet.

Die Kernidee des HAMVIS-Grundmodells besteht darin, vordefinierte Konzepte für die Verwendung in Domänenmodellen bereitzustellen, die sich jeweils durch eine spezielle Visualisierungsform für die damit modellierten Objekte auswirken. Ich möchte diese Vorgehensweise anhand von einigen Beispielen verdeutlichen.

Im Grundmodell ist z.B. ein Konzept „implizit definierter räumlicher Bereich“ enthalten. Dieses Konzept sieht eine Relation „definiert durch“ zu einem „physikalischen Objekt“ vor, das die Ausdehnung des räumlichen Bereichs bestimmt. Der Bereich ist also in gewisser Weise nur vorhanden, wenn auch das definierende Objekt vorhanden ist. Wenn ein solcher Bereich zu präsentieren ist, so sollte das defi-

nierende Objekt ebenfalls dargestellt werden. Ein solcher Bereich könnte in einem Domänenmodell z.B. der Schwenkbereich eines Krans sein. Wenn ein Kranschwenkbereich in einer Visualisierung benötigt wird, so sollte auch der diesen Bereich definierende Kran mit eingezeichnet werden. Für die Krandarstellung selbst ergeben sich Einschränkungen aus der rhetorischen Stellung des Krans als „Rechtfertigung“ für den Schwenkbereich. Der Kran könnte durch blasse Farben abgeschwächt dargestellt werden, um die rhetorische Funktion graphisch umzusetzen.

Weiterhin kann ein räumlicher Bereich scharf abgegrenzt sein oder als räumlicher Bereich mit fließendem Übergang in andere Bereiche modelliert werden. Im ersteren Fall würde bei der Darstellung eine scharfe Abgrenzung etwa durch ein schwarze Linie gewählt werden, während dieses im zweiten Fall nicht erfolgen sollte. Durch Hinzunahme des Konzeptes „scharf abgegrenzter Bereich“ wird ein Schwenkbereich eines Krans entsprechend visualisiert.

Ein Beispiel aus der XKL-Anwendung sind Gangbereiche innerhalb des Flugzeugrumpfes. Es bestünde die Möglichkeit, Freiräume wie z.B. Gänge indirekt durch die Objekte, die die Form definieren (Sitze, Küchen, Waschräume), darzustellen, sofern es die Diskursstellung erlaubt. Für diese Darstellungsform wird im Grundmodell ein Konzept „indirekt darstellbarer Bereich“ vorgesehen. Die indirekte Darstellung von Objekten ist insbesondere dann attraktiv, wenn die Objekte, die das indirekt dargestellte Objekt definieren, aus anderen Gründen sowieso präsentiert werden müssen. Gänge werden ohne die formbestimmenden Objekte allerdings eventuell nicht als Gänge erkannt und müssen ggf. durch Beschriftungen als solche ausgewiesen werden. Diese und andere Darstellungsanforderungen lassen sich durch Verwendung eines entsprechenden Oberkonzeptes aus dem Grundmodell erben (siehe die ähnlichen Arbeiten von Bateman für die Textgenerierung [18]).

Neben der taxonomischen Beschreibung von Objekten muß auch eine Beschreibung der Zerlegung von Objekten unterstützt werden (partonomische Beschreibung). Ein Flugzeug läßt sich z.B. für die XKL-Anwendung als Komposition aus einem Nasenteil, einer Kabine und einem Heckteil repräsentieren. Jedes dieser Teile besteht wieder aus weiteren Teilen usw. Bei Teil-von-Relationen gibt es jedoch verschiedene Ausprägungen, die sich durch entsprechende Darstellungsformen auszeichnen können. Nach dem vielzitierten Papier von Winston et al. [341] kann z.B. zwischen einer meronymischen Inklusion (Zerlegungsformen Component/Object, Member/Collection, Portion/Mass, Stuff/Object, Feature/Activity, Place/Area) und einer „räumlichen Inklusion“ unterschieden werden. Im Kontext von HAMVIS spielt zunächst die Component/Object-Beziehung eine Rolle. Mit dieser Beziehung kann z.B. ein Flugzeug in seine Bestandteile zerlegt werden (s.o.). Auch die Teil-von-Beziehung „räumliche Inklusion“ ist für HAMVIS relevant. Eine Kabine bzw. ein Kabinenrumpf enthält beispielsweise Kabinenobjekte wie z.B. Küchen, Waschräume, Sitze usw. Ein Nasenteil kann Elemente der elektronische Ausstattung, das Bugfahrwerk usw. räumlich enthalten.

Nehmen wir an, die für eine Anwendung aufgestellte Aktionenmodellierung enthalte eine Aktion „Verschiebe Objekt“, wobei bekannt ist, daß die manipulierten Objekte Elemente einer Menge M von Kabinenobjekten sind, die jeweils durch eine Berechnungsfunktion zur Laufzeit bestimmt werden. Die Objekte der Menge M müssen in einem Teilfenster der Anwendung dargestellt werden. Es reicht jedoch bei einer graphischen Ausgabe nicht, nur die reinen Kabinenobjekte zu präsentieren. Für die Interaktion wird mindestens noch ein Referenzsystem benötigt. Hierzu ist das Objekt geeignet, das die Kabinenobjekte räumlich enthält, also beispielsweise ein Objekt vom Konzept „Kabinenrumpf“, das wiederum „Teil“ der Kabine ist und deren Form bestimmt. Neben den verschiebbaren Kabinenobjekten sollte also noch der Rumpf in die Visualisierung aufgenommen werden. Es muß allerdings

geschlossen werden können, daß sich alle Elemente der Menge M im gleichen Flugzeug befinden, ein durchaus nicht trivialer Schluß. Weiterhin wird für eine Verschiebehandlung die Umgebung eines Objektes relevant sein. Es müssen also die in der Kabine enthaltenen Objekte, die nicht Elemente von M sind, ebenfalls präsentiert werden usw.

Die Komposition von Visualisierungen betrifft die Verschmelzung von Darstellungen in Teilfenstern. Nehmen wir also z.B. an, für eine andere Aktion, in der z.B. Bestandteile der elektronischen Ausrüstung festgelegt werden (siehe z.B. Abbildung 3) werde der Nasenteil des Flugzeugs als Referenzsystem benötigt. Zusammen mit dem obigen Kabinenrumpf ergeben sich zwei Möglichkeiten: zum einen können zwei Visualisierungen in verschiedenen Teilfenstern präsentiert werden, zum anderen könnte eine gemeinsame Visualisierung erzeugt werden, indem in der Zerlegungshierarchie der Knoten gesucht wird, der beide Referenzsysteme zu einem gemeinsamen Objekt verknüpft (in diesem Fall also vielleicht der gesamte Flugzeugrumpf). Voraussetzung für diese Art von Schlußfolgerung ist wiederum, daß gezeigt werden kann, daß der Nasenteil und der Kabinenrumpf Bestandteil desselben Flugzeugs sind. Die Form des Flugzeugs setzt sich auf dieser Ebene der Zerlegung allerdings auch aus dem Heckteil zusammen, so daß auch dieses mit dargestellt wird. Wenn in einer anderen Visualisierung wiederum das Heckteil als Referenzsystem benötigt wird, so kann auch hier eine Verschmelzung möglich sein, wenn bestimmte Bedingungen erfüllt werden können.

Diese Beispiele illustrieren, wie die Zerlegung von Objekten mit verschiedenen Teil-von-Relationen (Component/Object, räumliche Inklusion) zur Bestimmung des Inhaltes einer Darstellung verwendet wird und charakterisieren die notwendigen Inferenzschemata. Es wird deutlich, daß verschiedene Teil-von-Relationen mit jeweils spezieller Bedeutung für die Visualisierung im HAMVIS-Grundmodell bereitgestellt werden müssen. Der Grund für die Darstellung eines Objekts, d.h. die rhetorische Funktion eines Objekts, bestimmt die Wahl der zur Präsentation verwendeten Zeichenattribute. Damit dieses bei der Bestimmung von Standardwerten für Zeichenattribute berücksichtigt werden kann, müssen diese Informationen explizit in einem Dialogmodell vermerkt werden (näheres zu dieser Art der Visualisierungskomposition und zu den Eingriffsmöglichkeiten des Oberflächendesigners siehe Kapitel 3.3.).

Die Beschreibung von Objekten einer Anwendung mit ihrer Teil-von-Struktur und der entsprechenden dreidimensionalen geometrische Modellierung erfolgt auf der konzeptuellen Ebene durch das Systementwicklungsteam in einem sog. *Basismodell einer Anwendung*. In dem Basismodell für die XKL-Anwendung sind das Flugzeug und seine Bestandteile schon zur Entwicklungszeit als Instanzen bekannt. XKL wird für einen bestimmten Flugzeugtyp ausgelegt und dieser kann durch ein prototypisches Objekt beschrieben werden. Nicht zur Entwicklungszeit als Instanzen bekannt sind jedoch die in der Kabine enthaltenen Objekte wie z.B. Küchen, Waschräume oder Sitze.

In der Einleitung zu dieser Arbeit wurden verschiedene Beispiele besprochen, die darauf hinweisen, daß es vorteilhaft ist, bei der Modellierung verschiedene Sichten zu unterstützen (siehe insbesondere die Diskussion der Abbildungen 2 und 3). In der in Abbildung 53 gezeigten Visualisierung ist im Gegensatz zu der ähnlichen Darstellung in Abbildung 3 (siehe die Teilabbildung rechts oben) ein wesentlich detaillierteres Modell zugrundegelegt worden.

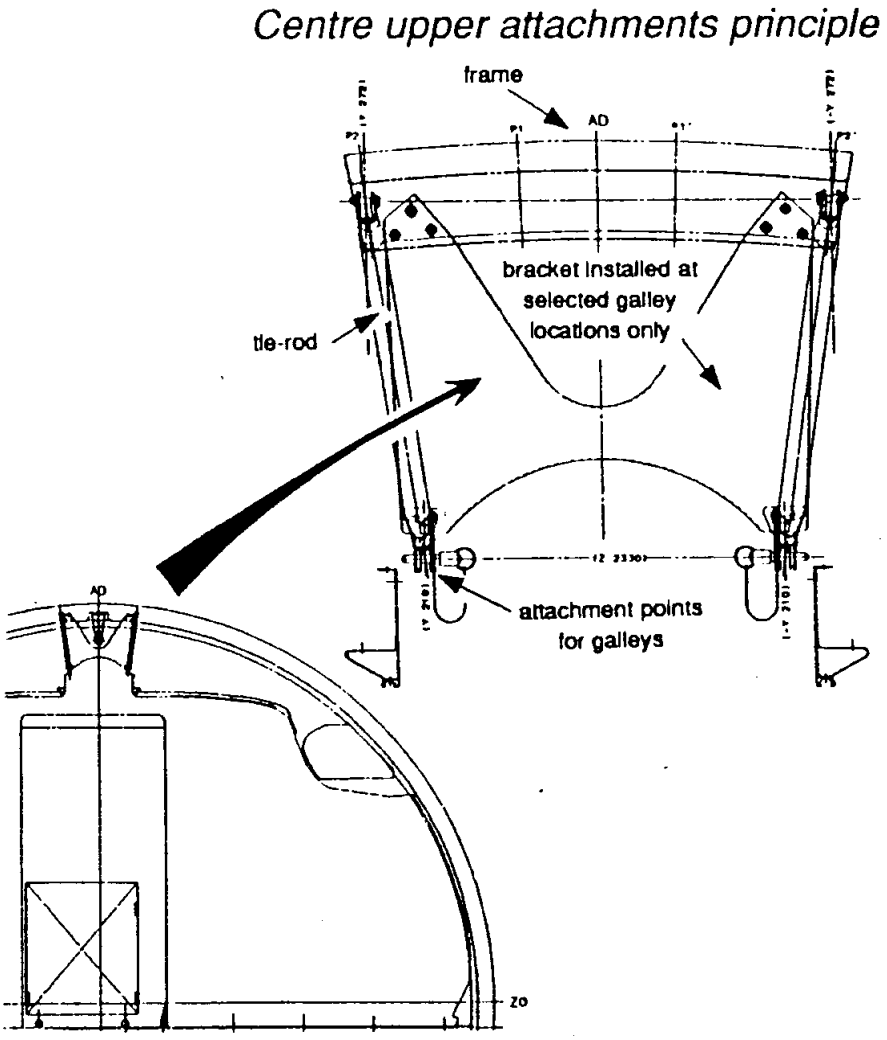


Abbildung 53. Visualisierung der Befestigungsmöglichkeiten für Küchen (entnommen aus einem Handbuch eines Flugzeugherstellers zur Konfiguration der Inneneinrichtung: „Cabin Configuration Guide“).

Der Begriff „detaillierter“ bezieht sich hier auf eine zusätzlich dargestellte Befestigungseinrichtung, die in der Kabine räumlich enthalten und als spezielles Konzept erfaßt ist. Die „Befestigungsklammern“ sind nur an speziellen Punkten installiert, so daß eine beliebige Verschiebung von Küchen nicht möglich ist.

Für eine Verschiebehandlung kann die Darstellung von Zusatzinformationen notwendig sein (z.B. die Befestigungspunkte aus Abbildung 53, siehe auch die Darstellung des Wasseranschlusses in Abbildung 2). Wenn nun für die Generierung einer detaillierten Visualisierung sehr spezielle Modellierungsformen benötigt werden und die hierzu notwendigen Deklarationen schon im Basismodell der Anwendung stehen, so müßte bei der Generierung von „gröberen“ Visualisierungen mit weniger Detailreichtum entschieden werden, was weggelassen wird. Anders ausgedrückt: Bei der Festlegung des Inhaltes von Visualisierungen würde festgelegt, welches Objekt präsentiert wird und welches nicht (komplexes Auswahlproblem bzgl. darzustellender Domänenobjekte). Bei der Diskussion von IMMP-Systemen in Abbildung 2.4 wurde dieses schon eingehend diskutiert. Es muß in einem Aus-

wahlprozeß entschieden werden, „was“ in einem bestimmten Kontext kommuniziert wird und – genauso wichtig – was nicht. Hierzu müssen kommunikative Ziele eines Präsentationsagenten und Wissens- und Glaubenzustände explizit modelliert werden und viele Detailentscheidungen formalisiert werden, wenn nicht mit Schemata gearbeitet werden kann.

In der betrachteten Anwendungsklasse der geometrischen Layoutsysteme wird durch die Betrachtung verschiedener handgezeichneter Visualisierungen deutlich (siehe die Abbildungen 2, 3 und 53), daß die wissensbasierte Modellierung der Welt eng an die geometrische Repräsentation der Objekte gekoppelt ist, d.h. die *logisch-begriffliche* Sicht ist eng an die *geometrische* Sicht gekoppelt (und umgekehrt). Eine solche logische-begriffliche Sicht, die an die geometrische, zweidimensionale Sicht gebunden ist, möchte ich als *Modell* bezeichnen.

Statt nun also von *einem* Weltmodell auszugehen und aus diesem (zur Laufzeit) die darzustellenden Objekte auszuwählen, sieht HAMVIS vor, zur Entwicklungszeit ein Modell zu bestimmen, mit dem zur Laufzeit dann die Darstellung von Objekten erfolgt, ohne daß aufwendige Auswahlentscheidungen zu treffen sind. Der Oberflächenentwickler definiert also verschiedene Modelle und HAMVIS schließt, welche Modelle in welchem Kontext eingesetzt werden können. Bevor jedoch die Schlußfolgerungen über Modelle genauer betrachtet werden, sollen noch einige Termini zu Charakterisierung von Konzepten eingeführt werden.

Charakterisierung von Konzepten

HAMVIS wurde für die Anwendungsklasse der interaktiven Layoutsysteme für geometrische Objekte entwickelt. Zur Beschreibung von Domänenwissen und zur Untersuchung der Integrationsmöglichkeiten von Grundmodell und Domänenmodellen erschien es mir daher sinnvoll, eine objektzentrierte Sicht auf die Repräsentation von Wissen mit Konzepten und Relationen bzw. Kasusrollen zu verwenden. Objektzentrierte begriffliche Strukturen für Domänenobjekte können allerdings auf vielfältige Weise gebildet werden. Ich möchte in diesem Kontext den Ansatz von Knospe aufgreifen. Knospe stellt eine Charakterisierung von Konzepten zur Repräsentation von begrifflichem Wissen auf der Meta-Ebene vor [151]. Seine Konzeptcharakterisierung orientiert sich an kognitionspsychologischen Befunden.

Knospe stellt heraus, daß die menschliche Begriffsbildung durch Perzeptionsprozesse bestimmt ist (sensorische Begriffsbildung). Er weist auf die Arbeiten von Rosch et al. hin, nach denen bestimmte „Begriffsebenen“ in der menschlichen Kognition eine herausragende Stellung einnehmen. Rosch bezeichnet diese Ebene als *Basisebene* der Begriffskonzepte. Die Konzepte dieser Ebene (*Basiskonzepte*) sind die Begriffe, denen jeweils eindeutig ein bestimmtes mentales Bild zugeordnet sind, das aufgrund von vorangegangenen Perzeptionsprozessen aufgebaut worden ist. Innerhalb einer Abstraktionsebene existieren Konzepte, die bezüglich der Charakterisierung von Oberbegriffen Prototypcharakter haben (vgl. Eiche und Lärche als Unterkonzepte zu Baum). Die abstraktesten, noch sensorischen Konzepte werden auch als *Primärbegriffe* bezeichnet.

Die Bildung von *kategorialen Begriffen* ist durch Ereignisse oder Aktionen beeinflußt. Primärbegriffe werden zu einem kategorialen Oberbegriff zusammengefaßt, wenn sie in einer bestimmten Relation zu einer Aktion stehen. Knospe führt das Beispiel des Konzepts „Werkzeug“ an, das zu einer Handlung „Arbeiten“ über die Kasusrolle „Instrument“ in Beziehung gesetzt wird. Primärkonzepte unterhalb von „Werkzeug“ sind „Hammer“, „Säge“, „Schraubendreher“ usw. Durch die Beziehung zu der

Aktion „Arbeiten“ entsteht ein neuer, abstrakter *kategorialer Oberbegriff*. Sensorische Begriffe (wie z.B. Gans, Ente, Schwan) unterhalb eines Primärbegriffs (wie z.B. Vogel) können zu einem *kategorialen Unterbegriff* zusammengefaßt werden, wenn dieser in einer besonderen Beziehung zu einem Ereignis oder zu einer Handlung steht.[69] „Gans“, „Ente“ und „Schwan“ können z.B. zum kategorialen Unterbegriff „Schwimmvogel“ zusammengefaßt werden. Ein „Schwimmvogel“ steht in der Beziehung „Akteur“ zu einer „Schwimmen“-Handlung. Kategoriale Begriffe dienen lediglich als *Platzhalter* und werden anforderungsabhängig durch sensorische Begriffe ersetzt.

Die Begriffsbildung innerhalb der Theorie der Kognitionspsychologie ist allerdings nicht einheitlich. Weiterhin gibt es aus Sicht der Kognitionspsychologie keine klaren Grenzen, d.h. die Merkmale der sensorischen Repräsentation nehmen in der Konzepthierarchie eher kontinuierlich von unten nach oben ab, so daß die Charakterisierung von Primärbegriffen auch umstritten ist. Trotzdem lassen sich die Erkenntnisse der Kognitionspsychologie als Anregung für die Charakterisierung von Konzepten bei der formalen Wissenrepräsentation verwenden.

Wie können diese Charakterisierungen nun auf die Leitanwendung von HAMVIS übertragen werden? Nehmen wir an, Küchen, Waschräume und Sitze werden zu dem Konzept „Kabinenobjekt“ zusammengefaßt. Kabinenobjekte können in einer Visualisierung interaktiv verschoben werden. Kabinenobjekt ist ein kategorialer Oberbegriff, der in Bezug auf eine Aktion „Verschieben“ über die Kasusrolle „manipuliertes Objekt“ in Beziehung steht. Das Konzept „Sitz“ kann als kategorialer Unterbegriff gedeutet werden, denn durch „Sitz“ werden die Primärkonzepte „Passagiersitz“ und „Flugbegleitersitz“ zusammengefaßt. Das Konzept „Sitz“ könnte beispielsweise für eine andere Aktion „Wählen eines Bezugstoffs“ relevant sein.

Das kategoriale Oberkonzept Kabinenobjekt unterteilt sich also in mehrere Primärkonzepte („Küche“, „Waschraum“„Passagiersitz“, „Flugbegleitersitz“). Zu beachten ist, daß die geometrische Form von Waschräumen und Küchen vielleicht übereinstimmt. Es ist daher sinnvoll, daß Primärkonzepte bezüglich ihrer Zeichenattribute unterschieden werden. Die Aufgabe von HAMVIS besteht darin, Zeichenattribute so zu bestimmen, daß Primärkonzepte in jedem Fall untereinander graphisch unterscheidbar sind (z.B. können Küchen blau, Waschräume rot und Sitze grau dargestellt werden). Es müssen also zur Laufzeit der Anwendung bei der Ausgabe entsprechende *Fallunterscheidungen* vorgenommen werden, die zur Entwicklungszeit der Anwendung vorzubereiten sind. Über die Konzeptcharakterisierung als Primärkonzepte wird die Unterscheidung von Objekttypen durch Zeichenattribute vorbereitet (siehe auch die Ausführungen zur Einbeziehung von anwendungsspezifischen Visuellen Notationen als spezielle, konzeptspezifische Zeichenattribute). Zur Entwicklungszeit der Anwendung wird eine entsprechende Fallunterscheidung für das Laufzeitsystem eingeplant. Mit kategorialen Unterkonzepten können weitere Gruppierungen von Objekten gebildet werden, die sich in einer graphischen Darstellung bezüglich ihrer Zeichenattribute unterscheiden sollen. Zu beachten ist noch, daß nicht jedes Konzept als kategorialer Oberbegriff oder auch Unterbegriff fungiert.[70]

---

69. Kategoriale Unterbegriffe stehen in der Vererbungshierarchie oberhalb von den zusammengefaßten Primärkonzepten, da es sich formal um eine ODER-Verknüpfung handelt. Kategoriale Unterkonzepte stehen aber unterhalb von kategorialen Oberkonzepten.

70. Für eine Zusammenfassung von Konzepten durch ein Oberkonzept können auch Implementationaspekte bei automatischen Berechnungsfunktionen sprechen (z.B. zur Berechnung von entsprechenden Methodenkombinationen für generische Funktionen).

Definition und Auswahl von Modellen

Objekte werden dargestellt, um dem Benutzer einer Anwendung interaktive Handlungen zu ermöglichen. Die Art und Weise der Darstellung von Objekten muß sich an gewissen Dialogstrukturen orientieren, um benutzerfreundliche Interaktionsformen zu gewährleisten. Beide Arten von Wissen, Wissen über Handlungen und Wissen über Dialogstrukturen müssen daher für die Unterstützung der Visualisierungsgenerierung formal modelliert werden.

Während das Basismodell die dreidimensionale Modellierung von Domänenobjekten beschreibt, wird in speziellen „Visualisierungsmodellen“ die Welt aus zweidimensionaler Sicht beschrieben. In Visualisierungsmodellen sind Objekte in der von Knospe geschilderten Weise als Basiskonzepte bzw. Primärkonzepte charakterisierbar. Sie gelten als Basiskonzepte, wenn die zweidimensionale Form in einem Modell eindeutig bestimmt werden kann. Ob dieses der Fall sein soll, muß im Modell explizit festgelegt werden. Objekte, deren zweidimensionale Form in einem Modell nicht näher beschrieben wird, werden auch nicht dargestellt. Obwohl es also schon im Basismodell z.B. Wasseranschlüsse als 3D-Objekte geben kann, werden sie nur in dem speziellen lateralen Schnittmodell, das der Detaildarstellung aus Abbildung 2 zugrundeliegt, als Primärkonzepte repräsentiert, treten also in anderen Visualisierungen nicht auf. Die Aufgabe eines Modells besteht also auch in der Informationsfilterung. Visualisierungen sollen so detailreich wie nötig sein, nicht alle Objekte sollen aber in allen Visualisierungen erscheinen, um sie nicht zu überladen. Für jedes Teilfenster einer Anwendung wird ein spezielles Modell festgelegt.

Für die Durchführung von Aktionen können bestimmte Bedingungen an die darzustellenden Objekte geknüpft sein. Für die Verschiebung von Kabinenobjekten ist es z.B. notwendig, daß sich die entsprechenden zweidimensionalen Darstellungen nicht überlappen. Es muß durch HAMVIS ein Modell ermittelt werden, in dem dieses auf konzeptueller Ebene zugesichert worden ist. Zur Entwicklungszeit der Anwendung sind die konkreten Kabinenobjekte noch nicht bekannt, trotzdem soll das Modell eines Darstellungsfensters schon festgelegt werden, damit die entsprechenden Interaktionsformen geplant werden können. Für ein Modell ist also Wissen dieser Art auf konzeptueller Ebene zu erfassen. Modelle, die die Bedingungen von Aktionen nicht erfüllen, scheiden als „Kandidaten“ aus.

Betrachten wir noch einmal die Visualisierungskomposition der Kabine und des Nasenteils zum Gesamtreferenzsystem Flugzeug. Dieses ist nur möglich, wenn für beide Teilvisualisierungen ein Modell bestimmbar ist, das die gleiche Abbildung der dreidimensionalen Daten auf zweidimensionale Bildrepräsentation vorsieht. Für die Verschiebung wird z.B. eine Aufsicht benötigt. Wenn für die Festlegung der Parameter der Elektronik im Nasenteil nur ein Schnitt von der Seite in Frage kommt (aus welchen Gründen auch immer), so ist die oben beschriebene Komposition der Referenzrahmen natürlich nicht möglich.

Die Beispiele machen die Rolle von Modellen bei der Visualisierungsgestaltung und -komposition deutlich.

- Modelle legen eine Sicht auf die repräsentierten Objekte der realen Welt fest,
- Modelle ermöglichen damit eine Strukturierung des repräsentierten Wissens,
- Modelle werden bei der Ausgabe von Informationen in einem (Teil-)fenster für den Zugriff auf Objektinformationen verwendet,
- Modelle legen den Detailreichtum von Visualisierungen fest.

Die hier auf der Wissensebene skizzierten Inferenzschritte über Modelle müssen formalisiert werden. Modelle müssen für die dreidimensionale und zweidimensionale Beschreibung von Domänenobjekten eine Art „Kontext“ bereitstellen, der eine „relevante“ Menge von Konzepten und Rollen repräsentiert. Wenn für die Darstellung von Objekten in einem Teilfenster der Oberfläche zur Entwicklungszeit der Anwendung ein Modell festgelegt wird, so werden zur Laufzeit nur die Instanzen dargestellt, die von im Modell „sichtbaren“ Primär- bzw. Basiskonzepten subsumiert werden. Allerdings erfolgt hier keine Auswahl mehr. Wenn also z.B. die Kabinenobjekte, die in einem Kabinenrumpf enthalten sind, präsentiert werden (etwa zur Darstellungen der Umgebung der verschiebbaren Objekte), so werden alle diejenigen Instanzen graphisch angezeigt, deren Konzepte Basiskonzepte sind. Falls der Oberflächendesigner entscheidet, daß es zu viele sind, so kann er ein neues Modell definieren, in dem z.B. Stauschränke, Trennwände, Hundeboxen usw. keine Basiskonzepte sind. Durch Wahl eines Modells ist der Detaillierungsgrad der Darstellung bestimmt.

Bei der Entwicklung legt also der Modellierer fest, welche Konzepte Basiskonzepte sind und wie Objekte in Teile zerlegt werden. Wir werden bei der Diskussion der formalen Beschreibung von Konzepten sehen, wie Basiskonzepte charakterisiert werden. Wenn sich Modelle nicht wesentlich voneinander unterscheiden, z.B. wenn in einem Modell nur einige Konzepte spezieller definiert werden sollen, was dazu führt, daß sie als Basiskonzepte charakterisiert werden, ist es sinnvoll, ein Modell als „abgeleitetes“ oder „feineres“ Modell zu definieren. Der Begriff „fein“ bezieht sich dabei auf die Menge der Angaben zur Beschreibung eines Konzeptes.

Die Aufgabe von HAMVIS besteht zunächst darin, die „gröbsten“ Modelle zu finden, die die Anforderungen der Aktionen erfüllen. Bei einer Visualisierungskomposition muß wiederum das feinste gemeinsame Modell verwendet werden bzw. durch eine Komposition wird die Menge der möglichen Modelle entsprechend eingeschränkt. Der Oberflächendesigner kann sich bei der detaillierten Dialogmodellierung auch für ein feineres Modell entscheiden als laut Aktionenanforderungen notwendig ist, wenn ihm die Darstellung besser gefällt. Auf die Verwendung von Modellen zur Visualisierungskomposition wird detailliert in Kapitel 3.3 eingegangen.

Mit dem hier eingeführten Modellbegriff kann die Visualisierungsgestaltung auf grober Ebene durch den Oberflächendesigner festgelegt werden. Das Wissen über die Gestaltung von Präsentationen wird nicht implizit aus der Sicht eines automatischen Präsentationsplanungsagenten vorgenommen, sondern explizit durch die sichtenorientierte Modellierung der Welt. Die konkrete Ausgestaltung von Darstellungen wird dadurch allerdings weit weniger auf den Benutzer und auf die spezielle Benutzungssituation abgestimmt sein. Da jedoch bei visuellen Präsentationen, die durch IMMP-Systeme adaptiv erzeugt werden, nicht ausgeschlossen werden kann, daß zuviel Adaptivität auch schädlich sein kann, nehme ich dieses in Kauf. Nach meinen Erfahrungen sind auch gute, manuell gestaltete visuelle Oberflächen zumindest für unerfahrene Benutzer doch erst nach einer Einarbeitungsphase *produktiv* verwendbar. Die Frage ist, was passiert, wenn sich durch eine adaptive Gestaltung von Ausgaben die Interaktionsformen ständig ändern?

In den nachfolgenden Abschnitten, wird geschildert, wie im HAMVIS-System die Beschreibung von Modellen mit Hilfe von Beschreibungslogiken auf der formalen Ebene unterstützt wird.

3.1.2 Beschreibungslogische Modellierung mit dem HAMVIS-Grundmodell

Da Fragen der Speicher- und Verarbeitungseffizienz in dieser Arbeit nicht im Vordergrund stehen, wird im Kontext von HAMVIS nicht ein speicherorientiertes Objektmodell verwendet (z.B. Objekte mit „Slots“ wie aus der objektorientierten Programmierung bekannt). Wichtiger ist die Möglichkeit, mit Hilfe von deklarierten Domänenmodellen Ableitungen für die Visualisierungsgestaltung durchzuführen und die Inferenzen (in möglichst vielen Aspekten) formal zu beschreiben. HAMVIS verwendet daher als grundlegende Modellierungsstruktur ein logisches bzw. relationales Objektmodell, das als Terminologische Logik bzw. als Beschreibungslogik bezeichnet wird ([183], [237], [345], [228]). Durch die logische Semantik von Beschreibungslogiken können Repräsentationsstrukturen und „Berechnungen“ in vielen Aspekten formal beschrieben werden. Im Gegensatz zur Prädikatenlogik lassen sich entscheidbare Teilsprachen definieren. Der Vorteil der Objektzentriertheit wird allerdings mit einer Einschränkung in der Ausdrucksmächtigkeit erkauft.

Für die Erstellung der Prototypimplementierung von HAMVIS wurde als konkrete Ausprägung einer Beschreibungslogik das System CLASSIC bzw. KRSS-CLASSIC in der Version 2.2 verwendet [258].[71] In den nachfolgenden Abschnitten wird die Anwendung von beschreibungslogischen Repräsentations- und Inferenzformen im Rahmen von HAMVIS aufgezeigt. Weiterhin wird geschildert, welche Erweiterungen notwendig waren und wie sich die Erweiterungen wiederum in den theoretischen Rahmen der Beschreibungslogiken einbetten lassen. Beispiele für Deklarationen von Konzepten und Rollen durch logische Axiome werden – soweit möglich – in der standardisierten KRSS-Lisp-Syntax angegeben (siehe [244] und [258]). Ich möchte hier darauf hinweisen, daß es sich bei den Beispielen zumeist um logische Formeln handelt. Vereinzelt werden prozedurale Kontrollkonstrukte in einigen Beispielen in der Sprache Common Lisp sowie Erweiterungen hiervon angegeben.[72]

Als erste Erweiterung von CLASSIC bzw. KRSS wurde in HAMVIS eine Namensraumverwaltung realisiert, die im folgenden kurz erläutert wird, da sie zum Verständnis der Beispiele notwendig ist.

Modelle als Namensräume

Namensräume sind aus Sicht der Theorie der Semantik einer Beschreibungslogik vielleicht ohne Bedeutung, aber unumgänglich für die praktische Verwaltung von Definitionen. Ein Namensraum für beschreibungslogische Deklarationen wird im HAMVIS-Kontext als *Modell* bezeichnet.[73] Modelle können Namen aus anderen Modellen importieren und selbst Namen exportieren. In Abbildung 54 ist die Deklaration des Basismodells für die XKL-Anwendung aufgeführt.

---

71. Das Upper Model von Bateman et al. ist mit LOOM (Version 2.1) realisiert. Eine erste Version von HAMVIS wurde ebenfalls mit LOOM implementiert. Es hat sich jedoch herausgestellt, daß für HAMVIS die von CLASSIC angebotenen Repräsentationsmechanismen noch ausreichend sind bzw. mit vertretbarem Aufwand entsprechend erweitert werden konnten. Die verwendete Implementation von CLASSIC 2.2 hat sich nicht nur in Bezug auf die Vollständigkeit der Algorithmen als wesentlich stabiler erwiesen als die von LOOM. Ein großer Vorteil von CLASSIC sind auch die Erklärungsmöglichkeiten für Subsumtionsbeziehungen bzw. – genauso wichtig – fehlende Subsumtionsbeziehungen.

72. Die äquivalente aber kürzere Notation mit logischen Formeln erfordert ein Textverarbeitungssystem, das für umfangreiche mathematische Formelnotationen geeignet ist (z.B. LaTeX). Das von mir verwendete FrameMaker-System (Version 3.0) ist hierfür wenig geeignet, so daß ich die ASCII-orientierte, Lisp-basierte Notation gewählt habe.

```
(define-model (basic-domain-model-class xkl-model)
  (:use hamvis-upper-model)
  (:export cabin-object ...))
```

Abbildung 54. Deklaration für das Basismodell der mit HAMVIS entworfenen XKL-Anwendung.

Das deklarierte Modell heißt `xkl-model`, importiert alle Namen aus dem HAMVIS-Grundmodell `hamvis-upper-model`, und exportiert Namen für Konzepte und Rollen, die im Namensraum `xkl-model` deklariert werden. Der Bezeichner `basic-domain-model-class` beschreibt die Klasse der Modells. Modellklassen sind relevant für die Visualisierung von Objekten eines Modells und werden etwas später besprochen. Die allgemeine Deklarationsform für ein Modell sieht wie folgt aus:[74]

```
(define-model (model-class model-name ...)
  [ (:nicknames ... ) ]
  [ (:use model-name ...) ]
  [ (:shadow name ...) ]
  [ (:import-from model-name name ... ) ]
  [ (:shadowing-import-from model-name name ... ) ]
  [ (:export name ...) ] )
```

In Namensräumen können verschiedene Definitionen von benannten Konzepten koexistieren. Ich spreche dann auch von einer *Konzeptdefinition in einem Modell* oder, etwas kürzer, von einem *Konzept in einem Modell*. Auch Rollen gleichen Namens können in Namensräumen verwaltet werden. So können z.B. verschiedene Unterrollen mit gleichem Namen aber unterschiedlichem Namensraum definiert werden. Auch hier spreche ich dann von einer *Rolle in einem Modell* oder, synonym, von einer *modellspezifischen Rolle* (Notation: `model-name:role-name` für exportierte Namen bzw. `model-name::role-name` für nicht-exportierte Rollennamen).

Betrachten wir hierfür wieder die Beispiele aus Abbildung 3. Die Flugzeugnase ist hier z.B. aus unterschiedlicher geometrischer Perspektive dargestellt, d.h. es müssen verschiedene zweidimensionale geometrische Formen verwaltet werden (sog. projektive Formen). Es bietet sich an, hierzu zwei verschiedene Unterrollen einer Rolle `has-projective-form` zur Beschreibung der verschiedenen zweidimensionalen Darstellungen zu verwenden. Obwohl irrelevant für die logische Semantik, ist es doch für den menschlichen Modellierer von Nachteil, wenn er hierfür stets neue Namen vergeben müßte (z.B. mit angehängter laufender Nummer). Mit dem Konzept des Namensraums kann dieses

---

73. Die Bezeichnung „Modell“ könnte eventuell mit dem Modellbegriff aus der prädikatenlogischen Semantik für Beschreibungslogiken in Konflikt stehen. Im sprachlichen Kontext dieser Arbeit halte ich aber eine Verwechslung für recht unwahrscheinlich. Falls doch eine Verwechslung nicht auszuschließen ist, wird explizit auf die intendierte Deutung hingewiesen. Statt „Modell“ hätte auch der Begriff „Wissensbasis“ verwendet werden können. Dieser Begriff scheint mir jedoch zu allgemein verwendet zu werden. HAMVIS verwaltet Modelle als Objekte erster Klasse und führt Berechnungen mit diesen Objekten durch.

74. Nonterminale sind kursiv gedruckt, die Angaben in eckigen Klammern sind optional.

Für jedes Modell wird zur Implementierung der Namensraumverwaltung ein Modul (Package) von Common Lisp verwendet. Entsprechend gelten die gleichen Skopusregeln wie bei dem Package-System von Common Lisp. Die optionalen Angaben entsprechen denen von `defpackage` (Common Lisp the Language 2nd Edition [294] S. 270ff.). Mit Hilfe der `:export`-Option werden die exportierten Namen deklariert. Mit der Option `:use` werden alle exportierten Namen der dahinter aufgeführten Modelle importiert. Durch `:shadow` können hierzu Ausnahmen deklariert werden. Die Option `:import-from` bzw. `:shadowing-import-from` gestattet einen selektiven Import bestimmter Namen aus den angegebenen Modellen.

vermieden werden. Die beiden Unterrollen werden mit gleichem Namen in unterschiedlichen Modellen definiert. Durch die Speicherung von geometrischen Informationen in modellspezifischen Unterrollen kann z.B. zur Laufzeit der Anwendung wieder auf einfache Weise ein bestimmtes geometrisches Objekt zugegriffen werden, das kennzeichnend für ein Modell ist. Dieses ist z.B. in einer Zeichenmethode der Bibliothek des UIMS für die generische Funktion `present` relevant (siehe Kapitel 2.1.4).

```
(define-presentation-method present (individual
                                     (presentation-type spatial-object)
                                     (model geometry-model-class)
                                     stream)
  (let ((2d-image (get-fillers individual 'has-projective-form model))
    (draw-image stream individual 2d-image)))
```

Die Funktion `get-fillers` ermittelt den Wert der Unterrolle `has-projective-form` in dem Modell, das für die Darstellung von Objekten auf dem Ausgabestrom `stream` verwendet wird. Es werden die *Rollenfüller einer Rolle in einem Modell* bestimmt. Die Klassen `spatial-object` und `geometry-model-class`, eine Unterklasse von `clim:view`, werden etwas später besprochen.

Weitere Beispiele für modellspezifische Rollen sind die Rollen zur Beschreibung der Zerlegung eines Objekts. Durch `define-model` wird automatisch für alle importierten Rollen eine entsprechende Unterrolle in dem definierten Modell erzeugt, so daß `get-fillers` und `set-fillers` immer wohldefiniert sind.

HAMVIS sieht auch eine Erweiterung der Erfragefunktion für die subsumierenden Konzepte eines Individuums vor. Es können von allen subsumierenden Konzepten diejenigen herausgefiltert werden, deren Namen in einem bestimmten Modell sichtbar sind. Ich spreche auch von *Subsumtionsbeziehungen in einem Modell*.

Das Beispielmodell `xkl-model` aus Abbildung 54 importiert alle exportierten Bezeichner des HAMVIS-Grundmodells (`hamvis-upper-model`).

Das HAMVIS-Grundmodell

Das HAMVIS-Grundmodell enthält in der momentanen Fassung Konzept- und Rollendeklarationen zur Beschreibung von Aspekten von räumlichen Objekten, die für die Visualisierung relevant sind. Neben der Geometrie wird auch die Komponentenstruktur von Objekten mit verschiedenen Konzepten und Rollen beschrieben. Abbildung 55 zeigt einen Ausschnitt der Konzepthierarchie des HAMVIS-Grundmodells. In der Abbildung 55 sind im Graphikfenster einige Konzepte unterhalb des Basiskonzeptes `um-object` skizziert (`um-object` steht für `Upper-Model-Object`). Das Textfenster (`Concept Info`) vermittelt einen Eindruck von der Definition des Konzeptes `physical-object` mit Oberkonzepten und Rollenfüllereinschränkungen. Mit diesem Konzept werden starre Körper modelliert. Mit den Konzepten des HAMVIS-Grundmodells ist es möglich, Wissen über geometrische Objekte dieser Art auf konzeptueller Ebene zu beschreiben. Das HAMVIS-Grundmodell sieht z.B. Konzeptbeschreibungen vor, mit denen ausgedrückt werden kann, daß die zur Laufzeit erzeugten Instanzen in jedem Fall geometrisch überlappungsfrei sind (`non-overlapped-spatial-object`). Weiterhin kann über Konzepte ausgedrückt werden, daß z.B. die dreidimensionale Form einer räumlichen Region (`spatial-region`) nicht intrinsisch festgelegt ist, sondern über die angrenzenden (physikalischen) Objekte der Umgebung definiert wird (Unterkonzept `implicitly-defined-spatial-region`, siehe die motivierenden Beispiele von oben).

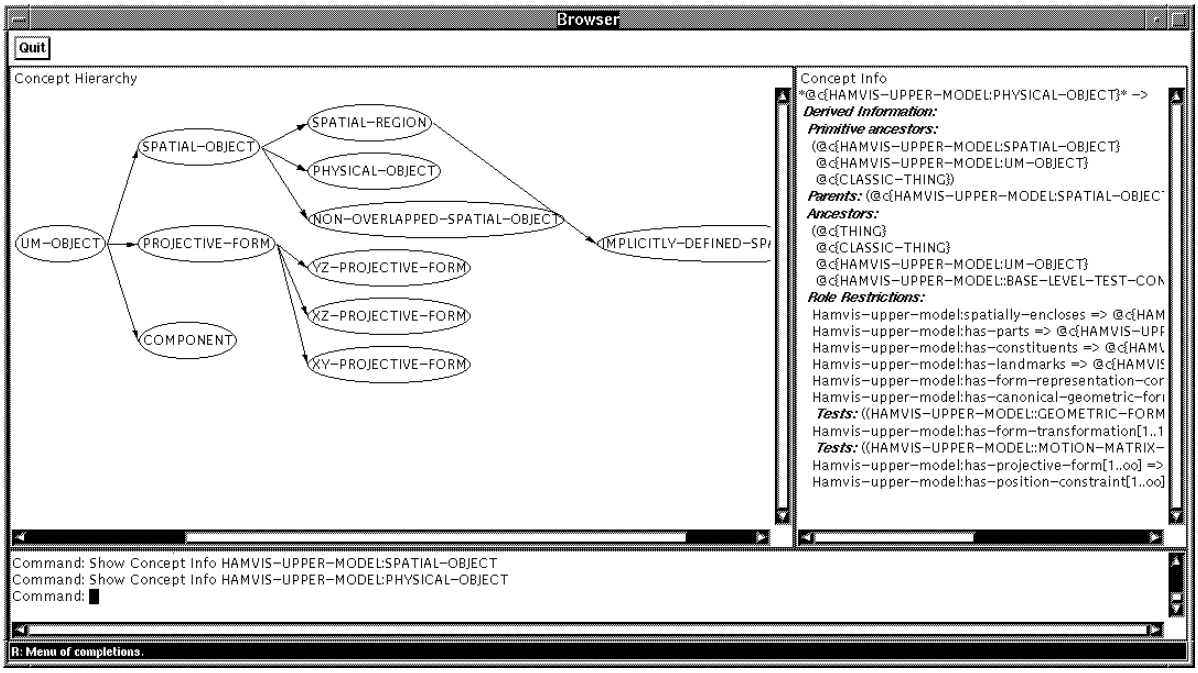


Abbildung 55. Konzeptinspektorfenster von HAMVIS. Im linken Teilfenster wird eine Graphdarstellung für Ober- oder Unterkonzepte gezeigt. Im rechten Teilfenster kann man sich durch Mausklick auf Graphknoten eine textuelle Darstellung der Konzeptbeschreibung anzeigen lassen. Namen für Konzepte in der Textdarstellung sind wiederum maussensitiv, d.h. durch Mausklick kann im Graphfenster die entsprechende Konzepthierarchie eingeblendet werden usw.

Die Deklarationen zur Repräsentation von geometrischen Daten im HAMVIS-Grundmodell werden in textueller Form in Abbildung 56 präsentiert. Aus der Konzeptdefinition von `spatial-object` geht hervor, daß räumliche Objekte zu einer kanonischen Formrepräsentation über die Rolle `has-canonical-form-representation` in Beziehung gesetzt werden.[75] Verschiedene Arten der Teil-Ganzes-Zerlegung werden im HAMVIS-Grundmodell in Form einer Rollenhierarchie modelliert. Die Basisrolle ist `has-decomposition`. Die Unterrolle `has-parts` bildet die Basis für die meronymische Zerlegung von Objekten nach dem klassischen Papier von Winston et al. ([341], Seite 429). Das HAMVIS-Grundmodell repräsentiert mit `has-constituents` als Unterrolle von `has-parts` die „Component/Object"-Relation. Die beiden Unterrelationen `has-form-representation-constituents` und `has-landmarks` erlauben eine weitere Ausdifferenzierung dieser Rollen. Ein Flugzeug kann z.B. allgemein aus Nase, Cockpitfenstern, Kabine, Kabinentüren und -fenstern sowie einem Heckteil bestehen (`has-constituents`). Cockpitfenster und Türen können Landmarken sein,

75. CLASSIC und KRSS-CLASSIC gestatten nur die Definition von primitiven Rollen. Im folgenden Text wird also nicht zwischen definierten und primitiven Rollen unterschieden. Wenn von Rollen gesprochen wird, so sind immer primitive Rollen gemeint. Rollenhierarchien über primitive Rollen sind jedoch möglich. Die zweite Angabe bei einer Rollendefinition durch `define-primitive-role` beschreibt die Oberrolle. Die Konstante `nil` an dieser Stelle kennzeichnet eine Basisrolle. Der Begriff „Relation" wird in diesem Zusammenhang häufig synonym für den Begriff „Rolle" verwendet.

Die Möglichkeit der Angabe von Prädikatsfunktionen als Konzeptterme mit den Ausdrucksformen von CLASSIC (`test-h` und `test-c`) stellt eine Erweiterung der KRSS-Spezifikation dar.

d.h. auch in der Unterrelation `has-landmarks` zu dem Flugzeug stehen. Die geometrische Form eines Flugzeugs setzt sich aus der Form der Nase, der Kabine und des Heckteils zusammen, d.h. diese Teilobjekte stehen in der Unterrelation `has-form-representation-constituents` zu dem Flugzeug. Es ist auch möglich, daß ein Objekt sowohl Landmarke ist als auch zur Formbestimmung beiträgt (z.B. der Nasenteil).

```
(in-model hamvis-upper-model)

(define-primitive-concept um-object classic-thing)

(define-primitive-concept component um-object)

(define-primitive-role has-canonical-geometric-form-representation nil)
(define-primitive-role has-form-transformation nil)
(define-primitive-role has-position-constraint nil)
(define-primitive-role has-projective-form nil)

(define-primitive-role has-decomposition nil)

(define-primitive-role has-parts has-decomposition)

(define-primitive-role has-constituents has-parts)

(define-primitive-role has-form-representation-constituents
  has-constituents)

(define-primitive-role has-landmarks has-constituents)

(define-primitive-role spatially-encloses has-decomposition)

;;; ---------------------------------------------------------------------

(define-primitive-concept spatial-object
    (and um-object
         (all has-canonical-geometric-form-representation
              (test-h geometric-form-representation-p))
         (at-least 1 has-canonical-geometric-form-representation)
         (at-most 1 has-canonical-geometric-form-representation)
         (all has-form-transformation (test-h motion-matrix-p))
         (at-least 1 has-form-transformation)
         (at-most 1 has-form-transformation)
         (all has-projective-form projective-form)
         (at-least 1 has-projective-form)
         (all has-position-constraint position-constraint)
         (at-least 1 has-position-constraint)))

;;; ---------------------------------------------------------------------

(define-method generic-form-representation-p ((object t)) nil)
(define-method generic-form-representation-p ((object vantage:csgnode)) t)

(define-method motion-matrix-p ((object t)) nil)
(define-method motion-matrix-p ((object vantage:motion-matrix)) t)
```

Abbildung 56. Ausschnitt aus dem HAMVIS-Grundmodell mit den Basisrelationen für die Dekompositionsbeschreibung und Beschreibung von geometrischen Daten. Die Formrepräsentation erfolgt mit dem CSG-Modellierer VANTAGE (siehe Text).

Eine räumliche Inklusion wird durch die Rolle `spatially-encloses` beschrieben (vgl. die Klassifikation von Winston et al. [341], Seite 429). Eine Kabine kann z.B. Sitze, Küchen usw. enthalten. Auf die Zerlegung von Aktionen in Teilaktionen mit einer anderen Unterrolle zu `has-decomposition` gehe ich später ein.

Nicht alle deklarierten Konzepte und Rollen werden aus dem HAMVIS-Grundmodell exportiert (z.B. nicht `um-object` und andere interne Konzepte). Für Domänenmodelle stehen nur bestimmte Konzepte und Relationen zur Spezialisierung zur Verfügung. Einige Rollen werden für Visualisierungszwecke „intern“ verwaltet, d.h. im Bereich der Modellbildung wird ein Modularisierungseffekt erzielt. Modelle sind voneinander abgrenzbar. Die genaue Bedeutung der hier vorgestellten Relationen für die Visualisierungsgenerierung wird in nachfolgenden Abschnitten noch im Detail besprochen.

Das Konzept `spatial-object` bildet die Basis für speziellere Charakterisierungen von räumlichen Objekten. Abbildung 57 zeigt die Definition von `physical-object` und `spatial-region` für die spezielle Einschränkungen für die Komponentenstruktur vorgegeben sind.

```
(define-disjoint-primitive-concept physical-object
    (kinds-of-spatial-objects)
  (and spatial-object
       (all spatially-encloses spatial-object)
       (all has-constituents spatial-object)))

(define-disjoint-primitive-concept spatial-region
    (kinds-of-spatial-objects)
  (and spatial-object
       (all spatially-encloses spatial-object)
       (all has-constituents spatial-object)))
```

Abbildung 57. Physikalische Objekte und räumliche Regionen.

Kopplung von Geometriedaten mit beschreibungslogischen Modellen

Die Repräsentation von geometrischen Daten erfolgt in HAMVIS nicht mit Hilfe von beschreibungslogischen Konstrukten. Obwohl dieses theoretisch möglich wäre, werden für die Verarbeitung von geometrischen Informationen meist spezielle Datenstrukturen benötigt, die mit zweistelligen Relationen nur schwer nachzubilden sind. Da der Realisierungsaufwand für ein Geometriemodul darüberhinaus immens wäre, ist aus softwaretechnischer Sicht der Rückgriff auf bestehende Softwarebibliotheken unumgänglich. Für die Repräsentation von räumlichen Daten wird im HAMVIS-Grundmodell ein geometrisches Modelliersystem verwendet, das auf der Technik der „Constructive Solid Geometry“ basiert. Es wurde das an der Carnegie-Mellon Universität entwickelte CSG-System VANTAGE [166] in das beschreibungslogische Repräsentationssystem CLASSIC integriert. Für die Untersuchungen zur Visualisierungsgenerierung im Rahmen dieser Arbeit sind die von VANTAGE bereitgestellten Repräsentationsstrukturen und geometrischen Algorithmen ausreichend.[76]

```
(define-csgnode %cabin-body
    vantage:cylinder
  (list (/ +cabin-height+ 2)
        +cabin-length+
        +faces+))

(define-csgnode cabin-body
    vantage:move
  '%cabin-body
  :trans *xkl-coordinate-transformation*)

(define-csgnode aircraft
    vantage:union
  (nose-and-cabin tail))

(define-csgnode aircraft-body-floor-1
    vantage:diff
  (aircraft-body floor-section-subtraction-block-1))

(define-csgnode aircraft-body-floor
    vantage:diff
  (aircraft-body-floor-1 floor-section-subtraction-block-2))

(define-csgnode aircraft-body-floor-fuselage
    vantage:diff
  (aircraft-body-floor aircraft-body-interior))
```

Abbildung 58. Beispiel für die Definition von geometrischen Objekten mit VANTAGE.

Abbildung 58 vermittelt einen Eindruck für die Möglichkeiten zur Definition von primitiven und zusammengesetzten geometrischen Objekten mit Hilfe der CSG-Operationen von VANTAGE. Basisobjekte der CSG-Repräsentation sind Zylinder, Quader, Kegel, Kegelstümpfe usw. Sie können mit den bekannten CSG-Operationen verknüpft werden. Es ergibt sich eine hierarchische Struktur von geometrischen Objekten. In der Abbildung 59 werden die geometrischen Daten für das in dieser Arbeit als Beispiel verwendete Flugzeug visualisiert.

---

76. Geometrische Informationen werden im HAMVIS-Grundmodell als CLOS-Objekte repräsentiert. Das von VANTAGE verwendete Frame-System wurde mit einer „CLOS-Hülle“ versehen. Statt der Verwendung von VANTAGE wäre auch ein Anschluß an ein CAD-System oder ein Austausch von geometrischen Informationen über Standardschnittstellen möglich (z.B. STEP [19], [20], [125], [150]). Die Ankopplung von verschiedenen Geometrierepräsentationsformen wird im HAMVIS-Grundmodell über generische Funktionen ermöglicht (siehe [294], [146]). Falls ein anderes System als VANTAGE integriert werden soll, müssen bestimmte generische Funktionen mit entsprechenden Methoden versehen werden. Abbildung 56 zeigt im unteren Teil einige Methoden für die in den Konzeptdefinitionen verwendeten generischen Funktionen für Testprädikate, die für VANTAGE-Objekte den Wert t liefern (siehe den `test-h`-Term in der Konzeptdefinition von `spatial-object`). Konzeptprädikate, die durch Test-Funktionen beschrieben sind, gelten als primitiv, da sie im Sinne der Beschreibungslogik nicht bewiesen werden können.

Für HAMVIS habe ich eine Erweiterung der generischen Funktionen von CLOS vorgenommen, so daß neben einer Methodendiskriminierung über Standard-Lisp-Typen auch eine Diskriminierung über CLASSIC-Konzepte möglich ist (`define-generic-function` und `define-method`). Die Semantik der Diskriminierung über Konzepte (ausgedrückt durch den Algorithmus zur Berechnung der Klassenpräzedenzliste) ist an die Semantik von CLOS angelehnt. Näheres hierzu schildert [215].

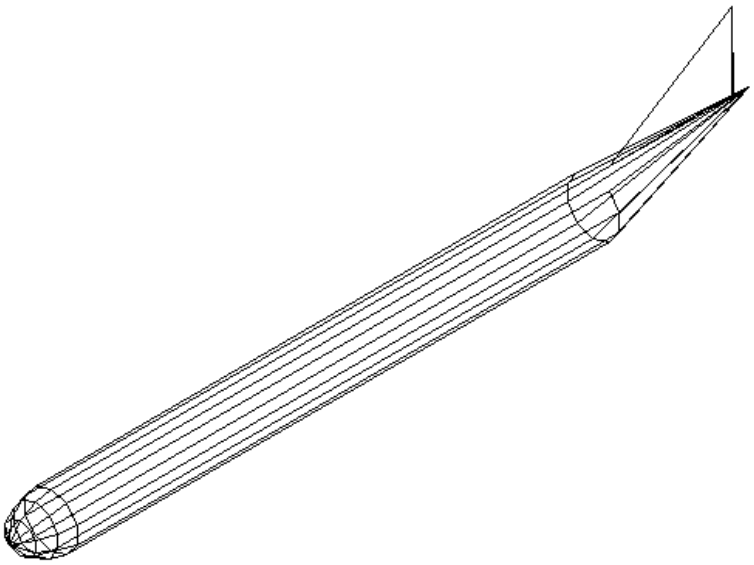


Abbildung 59. Visualisierung der dreidimensionalen Form des Flugzeugs aus der XKL-Anwendung.

Voraussetzung für die Eignung eines Geometrie-Repräsentationssystems für die Einbettung in HAMVIS ist die Möglichkeit zur *expliziten Repräsentation der Komponenten eines geometrischen Objekts*. CSG-Repräsentationen ermöglichen dieses. Sie sind durch eine baumförmige Zerlegung gekennzeichnet. In Abbildung 60 wird die CSG-Zerlegung des Flugzeugrumpfes aus Abbildung 59 innerhalb des HAMVIS-CSG-Inspektorfensters gezeigt. Die ellipsoiden Knoten des Baumes sind CSG-Knoten. Dazwischen sind die algebraischen CSG-Verknüpfungsoperationen dargestellt. Die Blätter symbolisieren die verwendeten Primitive.[77] In dem Beispiel setzt sich die Form eines Flugzeugs (`aircraft`) aus einem Nasenteil (`nose`), einer Kabine (`cabin`) und einem Heckteil (`tail`) zusammen, wobei jedoch noch einige „Hilfsknoten“ in der Zerlegungshierarchie auftreten können.

77. TRUN steht für Kegelstumpf (truncated cone), 2.5C steht für einen Kegelstumpf mit verschobener Spitze.

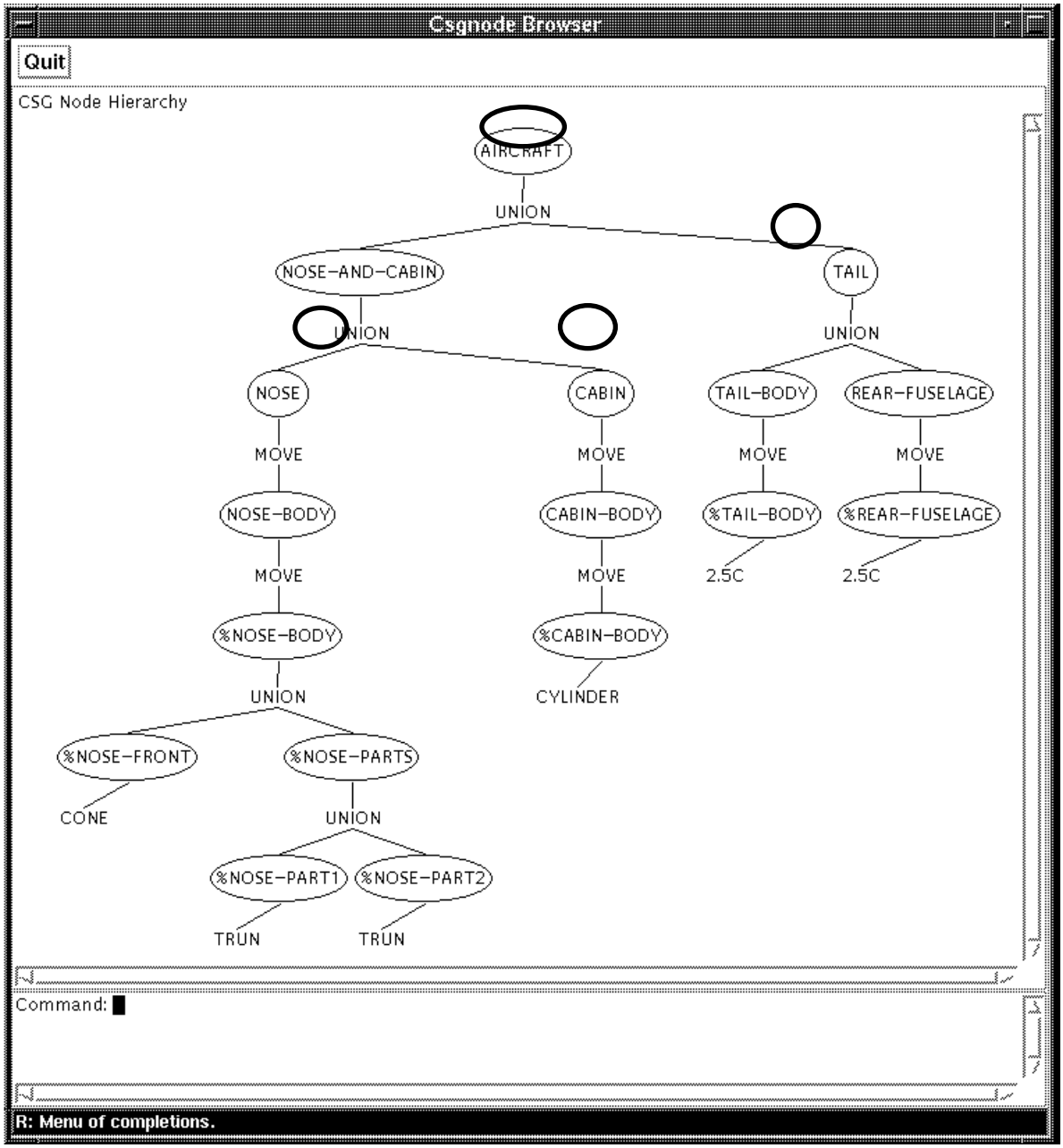


Abbildung 60. Zerlegung der geometrischen Form eines Flugzeugs.

In welcher Weise werden nun die geometrischen und die begrifflichen Informationen durch HAMVIS verknüpft? Ich möchte dieses an einem Beispiel erläutern. Nehmen wir an, die CSG-Knoten `aircraft`, `nose`, `cabin` und `tail` seien über die Relation `has-canonical-form-representation` mit entsprechenden Individuen `aircraft1`, `nose1`, `cabin1` und `tail1` auf der Seite der Beschreibungslogik verbunden. Weiterhin seien zur Deklaration der Objektzerlegung die Individuen `nose1`, `cabin1` und `tail1` über die Relation `has-constituents` mit `aircraft1` in Beziehung gesetzt, wobei über `has-constituents` noch andere Objekte als „Teile“ von `aircraft1` deklariert seien, z.B. `door1`, `door2`, `seat1` usw. HAMVIS schließt nun aus der geometrischen Beschreibung (CSG-Struktur) von `aircraft1`, daß `nose1`, `cabin1` und `tail1` nicht nur „einfache“ Konstituenten des

Flugzeugs sind, sondern daß die Form des Flugzeugs sich aus diesen Teilen zusammensetzt. Es werden für `aircraft1` automatisch entsprechende ABox-Assertionen hinzugefügt, so daß die Individuen `nose1`, `cabin1` und `tail1` nicht nur über die Zerlegungsrelation `has-constituents` mit `aircraft1` in Beziehung gesetzt werden, sondern sogar über die Unterrelation `has-form-representation-constituents` (siehe Abbildung 56). Über diese Relation wiederum führt HAMVIS Teil-Ganzes-Schlüsse bei der Visualisierungskomposition durch (siehe Abschnitt 3.1.8).

Die beschreibungslogische Konstituentenstruktur und die geometrische Zerlegung müssen aufeinander abgestimmt sein. Wenn ein Objekt O der Beschreibungslogik mit einer geometrischen Repräsentation G (ermittelt über die Relation `has-canonical-form-representation`) sich in Konstituenten O_i zerlegt, müssen die geometrischen Objekte G_i der Objekte O_i über eine Relation R von G aus durch Traversierung der geometrischen Struktur erreicht werden können. HAMVIS führt entsprechende Überprüfungen durch und expliziert die geometrische Information aus der CSG-Struktur durch entsprechende ABOX-Aussagen bezüglich der Relation `has-form-representation-constituents` und testet über beschreibungslogische Inferenzen, ob sich eventuell Widersprüche zur TBox ergeben usw.

HAMVIS ist zur Geometriemodellierung nicht auf die CSG-Repräsentation festgelegt. Bei der CSG-Repräsentation kann als Relation R die Nachfolgerrelation im CSG-Baum verwendet werden. Ein Geometrie-Repräsentationssystem kann in HAMVIS integriert werden, wenn die Geometrie-Repräsentation das nachfolgend definierte *schwache Kompositionalitätskriterium* erfüllt.[78]

Definition: Schwache Kompositionalität

> Eine geometrische Repräsentation heißt schwach kompositional genau dann, wenn für jedes geometrische Objekt die der Formrepräsentation zugrundeliegenden Konstituenten über eine transitive Relation bestimmbar sind.

Das HAMVIS-Grundmodell sieht zur Beschreibung von geometrischen Eigenschaften eines Objekts weiterhin eine Trennung zwischen Form und Position vor. Durch die CSG-Repräsentation wird die dreidimensionale Form eines Objektes in einem bestimmten Koordinatensystem festgelegt. Die Koordinaten der Form werden also noch durch eine zusätzliche affine Abbildung transformiert. Zur Anbindung der Abbildungsmatrix dient die Relation `has-form-transformation` (siehe Abbildung 56). Für räumliche Objekte sind die Füller auf den Typ `vantage:motion-matrix` eingeschränkt (siehe die Konzeptdefinition von `spatial-object` und die Methoden zu `motion-matrix-p` in Abbildung 56). Durch diese Abbildung wird die Position der Objekte in Bezug auf einen Koordinatenursprung O festgelegt (siehe Abbildung 61). Mit Hilfe der Relation `has-position-constraint` kann noch die Variationsmöglichkeit der Formtransformationsabbildung explizit angegeben werden. Abbildung 61 zeigt ein Beispiel für eine spezielle Positionseinschränkung, ein rechteckiger Bereich auf der xy-Ebene.

78. Zur Bestimmung der Nachfolger eines Knotens zur kompositionalen Geometrie-Repräsentation wird wiederum von HAMVIS eine generische Funktion verwendet. Für die Anbindung eines speziellen Repräsentationsformalismus muß eine entsprechende Methode definiert werden.

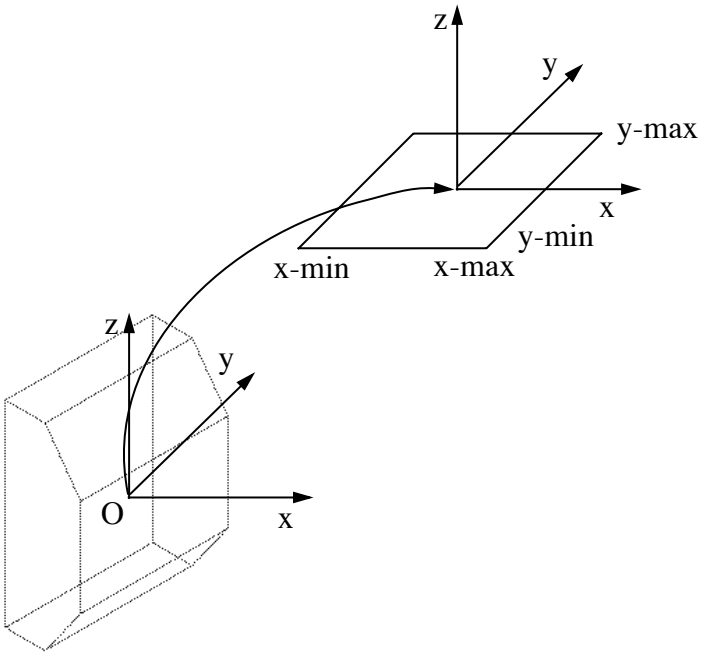


Abbildung 61. Festlegung der Position eines Objekts mit Freiheitsgeraden.

Abbildung 62 zeigt den entsprechenden Ausschnitt mit den formalen Definitionen aus dem HAMVIS-Grundmodell.

```
(define-primitive-concept position-constraint classic-thing)
(define-primitive-concept position-constraint-with-default
                          position-constraint)
(define-primitive-concept spatial-position-constraint position-constraint)

(define-primitive-role has-bounding-rectangle nil)

(define-concept bounding-rectangle-position-constraint
    (and spatial-position-constraint
         position-constraint-with-default
         (at-least 1 has-bounding-rectangle)
         (at-most 1 has-bounding-rectangle)
         (all has-bounding-rectangle (test-h bounding-rectangle-p))))

(define-method bounding-rectangle-p ((object t)) nil)
(define-method bounding-rectangle-p ((object clim:bounding-rectangle)) t)
```

Abbildung 62. Positionseinschränkungen zur Formulierung von Änderungsmöglichkeiten für die Formabbildung.

Für die Generierung von Visualisierungen müssen aus der dreidimensionalen Formrepräsentation (Rolle `has-canonical-geometric-representation`) projektive Abbildungen bestimmt werden. Die hierzu notwendigen Algorithmen werden durch VANTAGE bereitgestellt. Ein dreidimensionales Objekt wird zusammen mit einem Kameraobjekt zu einem Szenenobjekt zusammengefaßt

(siehe Abbildung 63). Eine projektive Abbildung (entweder perspektivische oder parallele Projektion) liefert ein 2D-Bildobjekt, das aus einzelnen Flächenobjekten mit Kantenobjekten und Eckpunkten besteht (für eine genaue Beschreibung siehe [166]).

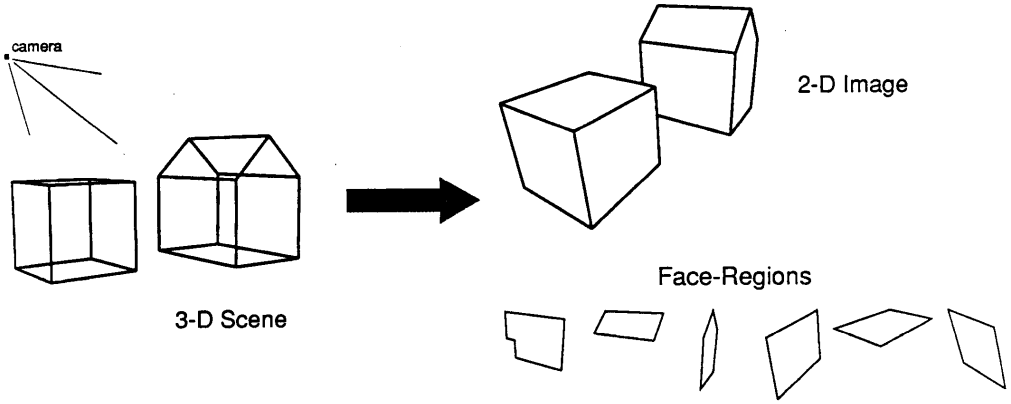


Abbildung 63. Verschiedene Objektklassen zur Verwaltung visueller Informationen: 3D-Szenen, 2D-Bilder und zweidimensionale Flächen (face regions) (aus [166], Seite 11).

```
(define-primitive-role has-2d-form-representation nil)

(define-primitive-concept projective-form
  (and um-object
       (all has-2d-form-representation
            (test-h projective-form-representation-p))))

(define-disjoint-primitive-concept xy-projective-form (form-partition)
  projective-form)

(define-disjoint-primitive-concept xz-projective-form (form-partition)
  projective-form)

(define-disjoint-primitive-concept yz-projective-form (form-partition)
  projective-form)

(define-method projective-form-representation-p ((object t)) nil)
(define-method projective-form-representation-p ((object vantage:2d-image)) t)
```

Abbildung 64. Modellierung von projektiven geometrischen Informationen.

Zweidimensionale geometrische Informationen werden im HAMVIS-Grundmodell als projektive Formen modelliert (Konzept `projective-form`, siehe Abbildung 64). Die CLOS-Objekte von VANTAGE werden über die Rolle `has-2d-form-representation` angebunden. Als besondere projektive Formen werden Abbildungen auf die `xy`-, `xz`- und `yz`-Ebene modelliert.

Die Konzepte und Rollen des HAMVIS-Grundmodells sind darüberhinaus noch auf die *formale Modellierung von Benutzerhandlungen* mit Domänenobjekten ausgerichtet. Der hierfür relevante Teil des HAMVIS-Grundmodells (Modellierung von generischen Aktionenkonzepten) wird in Kapitel 3.2 ausführlich erläutert. Ich halte das Grundmodell nicht für abgeschlossen. Wenn bestimmte Konzepte und Relationen für verschiedene Anwendungen einer Anwendungsklasse relevant sind, so können sie

auch ins Grundmodell aufgenommen werden. Das HAMVIS-Grundmodell ist offen für Erweiterungen wie z.B. andere Geometrie-Repräsentationssysteme oder die für eine Anbindung von CAD-Datenbanken.

Die Vorstellung der Konzepte des HAMVIS-Grundmodells und des verwendeten Geometrie-Repräsentationssystems auf der technischen Ebene ist recht komplex. Ich möchte daher zur Diskussion des Grundmodells noch einige allgemeine Aspekte der Wissensmodellierung auf der ontologischen Ebene diskutieren.

3.1.3 Ontologische Betrachtung des HAMVIS-Grundmodells

Im Bereich der Modellierung technischer Systeme geht der Trend von einer rein geometrischen CAD-Modellierung immer mehr in Richtung einer vollständigen Produktmodellierung, die in eine Gesamtunternehmensmodellierung und eine Modellierung des Produktionsprozesses eingebettet ist (vgl. [150]). Damit Modelle für verschiedene Anwendungsprobleme wiederverwendet werden können, wurden verschiedene Ansätze zur Wissensrepräsentation entwickelt, die Wissen formal repräsentieren, jedoch erst nach einem Transformationsschritt Deduktionsalgorithmen anbieten. Die Transformation geht allerdings einher mit einer Reduktion der Ausdrucksmächtigkeiten des Formalismus [105]. Uschold et al. entwickeln z.B. eine Ontologie, die als Basis für die Erstellung von Unternehmensmodellen dienen soll (Enterprise Ontology, [326] [327]) und die Begrifflichkeiten folgender Kategorien modelliert:

- Aktivitäten, Pläne, Fähigkeiten Ressourcen,
- Organisation
- Strategien (Zwecke, Entscheidungen, Annahmen, Einflußfaktoren)
- Marketing (Verkäufe, Markt usw.)
- Zeit

Zu beachten ist, daß der angesprochene Transformationsschritt nicht nur aus Sicht der Deduktionsalgorithmen notwendig ist, sondern auch für die Abstraktionsbildung notwendig ist. Ein Modell ist gerade auch durch das Weglassen von im jeweiligen Anwendungskontext nicht relevanten Informationen gekennzeichnet.

Mit der Erstellung einer Ontologie wird ein sehr viel höher gestecktes Ziel verfolgt als mit dem HAMVIS-Grundmodell. Das HAMVIS-Grundmodell hat nicht den Status einer allgemeinen Ontologie für räumliche Objekte und kann dieses beim heutigen Stand der Forschung auch nicht haben. Aber es wurde eine Basis geschaffen, so daß begriffliche Strukturen und geometrische Repräsentationsformalismen gekoppelt und zur Formalisierung von Schlußfolgerungsprozessen verwendet werden können. Dabei wird vom ontologischen Standpunkt her primär eine komponentenorientierte Modellierungsform für Domänenobjekte unterstützt (räumliche Objekte, Regionen usw. sowie deren Zerlegung).

Ähnlich wie beim „Upper-Model“ von Bateman für die verbale Kommunikation soll mit dem HAMVIS-Grundmodell untersucht werden, inwieweit es möglich ist, allgemeine Konzepte bereitzustellen, die eine Kommunizierbarkeit von Domänenobjekten über Mechanismen der Vererbung sicherstellen. Um „Visualisierungsdienste“ in Anspruch zu nehmen, müssen die Basiskonzepte und -rollen des HAMVIS-Grundmodells vom Systementwicklungsteam zur Deklaration von Domänenkonzepten

(und -rollen) verwendet werden. Es stellt sich die Frage, inwieweit das Grundmodell in gewisser Weise „vollständig“ ist. Der Begriff der Vollständigkeit ist in diesem Fall allerdings schlecht definierbar. Erst wenn das Grundmodell für verschiedene Anwendungen eingesetzt wird, kann sich m.E. ein *Konsens* für notwendige Basiskonzepte und -relationen herausbilden. Dieses ist noch nicht erfolgt. Zunächst einmal muß das Gesamtkonzept eines modellbasierten Unterstützungssystems zur Generierung von Visualisierungen für Benutzungsschnittstellen entworfen und begründet werden. Dieses war das Ziel der Entwicklung von HAMVIS.

Ich möchte im folgenden auf zwei Aspekte, die in diesem Zusammenhang für das HAMVIS-Grundmodell relevant sind, besonders eingehen: auf die Integration von begrifflichem und räumlichem Wissen sowie auf die Modellierung von Teil-Ganzes-Beziehungen.

Kombination von räumlichen und begrifflichen Repräsentationsstrukturen

Für die Generierung von Visualisierungen ist es notwendig, räumliches Wissen adäquat zu repräsentieren. Auf der einen Seite sollen durch beschreibungslogische Repräsentationsformen Inferenzen zur Visualisierungsgestaltung formal modelliert werden, auf der anderen Seite müssen existierende geometrische Repräsentationssysteme eingebunden werden.

In den vorigen Abschnitten wurde skizziert, welche Geometrie-Informationen im HAMVIS-Grundmodell durch beschreibungslogische Rollen explizit gemacht werden. Als Beispiel wurde die Verwaltung von Informationen zur Formrepräsentation geschildert. Ein Objekt ist aufgrund der geometrischen Struktur der kanonischen Formrepräsentation nicht nur eine einfache Konstituente (`has-constituent`), sondern in einigen Fällen sogar Füller der speziellen Unterrolle `has-form-representation-constituent`.

Räumliche Beziehungen zwischen Objekten werden durch die (qualitativen) Teil-von-Relationen des Grundmodells definiert, die oben diskutiert wurden. Aus Sicht der Beschreibungslogik handelt es sich um primitive Rollen, d.h. es finden auf dieser Ebene keine beschreibungslogischen Schlußfolgerungsprozesse über die Gültigkeit von Relationen zwischen Individuen statt. Zu den üblichen ABox-Ableitungsschemata werden also noch weitere, „räumliche Inferenzmechanismen“ hinzugefügt. Geometrische Beziehungen werden auf der Ebene der Beschreibungslogik als spezielle ABox-Aussagen explizit gemacht (zweistellige Prädikate).

Die oben im Kontext von HAMVIS betrachteten Aussagen über die Formrepräsentation auf der ABox-Ebene über `has-form-representation-constituents` sind hierfür nur ein Beispiel. Andere Beispiele können leicht aufgezählt werden. Ob ein Objekt sich z.B. „in“ einem anderen befindet, muß i.a. auf der ABox-Ebene explizit ausgesagt werden. Es wäre allerdings leicht möglich, bei der Erzeugung von geometrischen Objekten durch entsprechende Berechnungen auch notwendige Aussagen für das beschreibungslogische System automatisch zu generieren (z.B. Aussagen der Form `(related cabin-1 galley-42 spatially-encloses)`).[79] Die Frage ist nur, welche Beziehungen sind für beschreibungslogische Inferenzen relevant?

79. Zur Syntax von KRSS-Relationsassertionen siehe [244] Seite 4. `Cabin-1` wird mit `galley-42` über die Relation `spatially-encloses` in Beziehung gesetzt.

Von Haarslev, Möller und Schröder wurden in [112] geometrisch-terminologische Ableitungsformen mit verschiedenen Beispielen demonstriert. Es wird hier gezeigt, wie räumliche Schlüsse mit Hilfe von qualifizierten Anzahlrestriktionen (Hollunder und Baader [129]) auf der terminologischen Ebene durchgeführt werden können. Ein „Haus“ wird beispielsweise zu einer „Villa“ klassifiziert, wenn sich „in-der-Nähe-von“ dem Haus mindestens ein „Swimming-Pool“ befindet. Die In-der-Nähe-von-Relation wird durch geometrische Algorithmen bestimmt.[80] Im Gegensatz zu der hier diskutierten einfachen Form der Explizierung von räumlichen Informationen durch die Teil-Ganzes-Beziehung `has-form-representation-constituents` wird in [112] durch beschreibungslogische Schlüsse die geometrische Information auch in weiteren Ableitungen verwendet („Villa“ als Spezialisierung von „Haus“).

Ich möchte im nächsten Abschnitt auf Teil-Ganzes-Beziehungen genauer eingehen.

Modellierung von Teil-Ganzes-Beziehungen

Einige Autoren plädieren dafür, Teil-Ganzes-Beziehungen zur Abgrenzung von „normalen“ Relationen einen besonderen ontologischen Status zuzuweisen (z.B. Artale et al. [14]), damit sich Inferenzschemata, die mit Teil-Ganzes-Relationen verbunden sind, realisieren lassen. Beispiele für Standard-Inferenzschemata sind:

- Aggregierung bei gegebenen Teilen [238] [115],
- Vervollständigungsinferenzen [291]:
  - Inferenzen über den transitiven Abschluß von Relationen
  - Automatische Erzeugung von Teilen
  - Automatischen Propagierung von Einschränkungen für das Aggregat auf die Teile
- Klassifikations- und Subsumtionsinferenzen über die Teilestruktur [68].

Die Einführung von neuen Relationen*typen* (mit jeweils spezieller Semantik oder speziell hinzugefügten Axiomen) als Ergänzung zu den Standardrelationen aus der Beschreibungslogik (zweistellige Prädikate) birgt die Gefahr in sich, daß für jedes Repräsentationsproblem ein spezieller Relationentyp entworfen wird. Es kommt u.U. zu einer Inflation von unterschiedlichen Basiskonstrukten.

Verschiedene Arten von Teil-von-Relationen werden im HAMVIS-Grundmodell durch *Unterrollen* modelliert, so daß eine entsprechende Differenzierung erreicht werden kann. Für die Klassifikation von verschiedenen Teil-Ganzes-Beziehungen wurden in der Literatur verschiedene Ansätze vorgeschlagen (siehe z.B. die Aufsätze in [80] und auch den Ansatz von Sattler [272]). Das HAMVIS-Grundmodell orientiert sich grob an dem Modell von Winston, Chaffin und Herrmann ([341], siehe auch die Diskussion dieses Modells in [97]). Die Verwendung von primitiven Unterrollen hat pragmatische Gründe. Mit CLASSIC wurde eine, von der Ausdruckstärke gesehen, schwache Beschreibungslogik verwendet. Mit mächtigeren Konstrukten wie sie z.B. LOOM bereitstellt, lassen sich komplexere Bedingungen in Teil-von-Zerlegungen beschreiben. Dieses möchte ich kurz am Beispiel der Modellierungssprache BHIBS [60] darstellen. Abbildung 65 zeigt ein Beispiel für eine Zerlegung

80. Deklarative Repräsentationsformen für räumliche Konstellationen und Algorithmen für deren Erkennung von wurden z.B. von Schick, Kockskämper und Neumann entwickelt [273].

(Objektattribut `has-parts`) des Konzeptes `passenger-cabin`, das als Unterkonzept zu `construction-object` modelliert wird. Es können ein bis drei Teile des Konzeptes `class` auftreten, wobei jedoch höchstens eine Instanz von `first-class`, `business-class` oder `tourist-class` als Füller des Attributs `has-parts` zugelassen ist. Die drei letztgenannten Konzepte sind disjunkt und werden von `class` subsumiert. Es dürfen keine weiteren Konzepte als Zerlegung auftreten, die Zerlegung ist also vollständig beschrieben.

```
(isa! (a passenger-cabin)
      (a construction-object
           (has-parts
               #{> #[(a class) 1 3] := #[(a first-class) 0 1]
                                       #[(a business-class) 0 1]
                                       #[(a tourist-class) 0 1]})))

(defconcept passenger-cabin
  :constraints
  (:and construction-object
        (:all has-parts
              (:and class
                    (:or first-class business-class tourist-class)))
        (:at-least 1 (:and has-parts (:range class)))
        (:at-most 3 (:and has-parts (:range class)))
        (:at-most 1 (:and has-parts (:range first-class)))
        (:at-most 1 (:and has-parts (:range business-class)))
        (:at-most 1 (:and has-parts (:range tourist-class)))))
```

Abbildung 65. Beispiel für eine Teil-von-Zerlegung in BHIBS mit entsprechender Darstellung durch beschreibungslogische Konstrukte.

Obwohl für BHIBS keine mathematische Semantikdefinition angegeben wurde, scheint die Nachbildung der BHIBS-Konstrukte in der darunter aufgeführten Übersetzung der natürlichsprachlichen Beschreibung mit den Ausdrucksmitteln von LOOM sinnvoll. Das Beispiel belegt, daß eine beschreibungslogische „Rekonstruktion“ von Teil-Ganzes-Beziehungen keineswegs trivial ist. Es werden definierte Rollen und ein Oder-Konstrukt benötigt (beides wird von CLASSIC nicht angeboten). BHIBS stellt noch weitere Mechanismen zur Beschreibung einer Teil-von-Beziehung bereit. Zwischen den Teilen können Einschränkungen angegeben werden. Dieses ist z.B. bei geometrischen Beziehungen sinnvoll. Ich möchte aber dieses Thema hier nicht vertiefen (siehe [277]).

Auch Pribbenow betont die besondere Rolle von geometrischen Daten bei der Bestimmung von konkreten Teil-von-Strukturen [251]. Bei räumlichen Objekten werden beispielsweise Teile durch den Konstruktionsprozeß definiert (z.B. der Henkel einer Tasse). Allerdings können in verschiedenen Kontexten Teile auch aufgrund der Funktion des Gesamtobjekts selbst oder der Funktion des Teilobjekts im Gesamtobjekt definiert werden (z.B. der Boden einer Tasse). Zwischen den Teilen und dem Ganzen sowie auch zwischen den Teilen selbst bestehen komplexe geometrische und funktionale Beziehungen (siehe z.B. auch [75]).

Abbildung 66 zeigt zwei mögliche Zerlegungen für eine Tasse. In der oberen Zerlegung wird neben dem Henkel, der ein Teil auch bezüglich des Herstellungsprozesses ist, der Boden als besonderes Teil repräsentiert. In der unteren Teilabbildung ist hingegen der obere Rand (vielleicht ein Goldrand) als ein besonderes, als Individuum repräsentiertes Teil in der Zerlegung repräsentiert. Sowohl Boden als

auch Becherrand sind keine Teile, die sich aus der Konstruktion der Tasse ergeben, sondern jeweils eine bestimmte Funktion innerhalb eines bestimmten Kontextes haben.

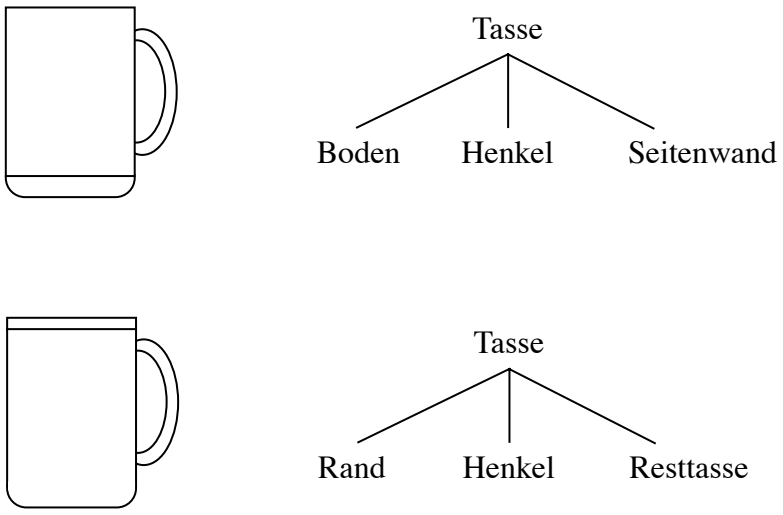


Abbildung 66. Zerlegung einer Tasse für verschiedene Anwendungen.

Beide Modelle können für verschiedene Zwecke auch innerhalb *einer* Anwendung benötigt werden. Dabei ist jedoch zu beachten, daß man trotz der unterschiedlichen Zerlegung aber jeweils nur ein einziges Objekt zur Repräsentation der speziellen Tasse verwalten möchte.

Ein Transformationsansatz im Sinne von Gruber (Ontolingua [105]), bei dem verschiedene Wissensbasen aus einer komplexen Ontologie generiert werden, ist in diesem Kontext nicht direkt anwendbar (siehe auch den Begriff „multiple engineering ontologies“ in [319]). Es entsteht die Notwendigkeit, verschiedene „Sichten“ auf *ein* Objekt zu erlauben, das als Individuum eindeutig identifizierbar ist. So kann es z.B. sinnvoll sein, in einer Sicht den Henkel der Tasse zu modellieren und in einer anderen Sicht gar nicht zu erfassen. In beiden Sichten werden eventuell vorhandenen Risse oder Sprünge ignoriert. In der zweiten Sicht ist die Präsentation des Henkels keine Frage der geometrischen Sichtbarkeit (Verdeckung durch die Tasse). Der Henkel ist in dieser Sicht als nicht vorhanden zu betrachten. Das Konzept „Henkel“ braucht nicht einmal definiert zu sein.

Die Fülle der verschiedenen Möglichkeiten macht deutlich, daß es nicht *ein* Modell für alle Anwendungsbereiche geben kann. Für spezielle Anwendungen werden neue Teil-von-Relationen bedeutsam sein. Weiterhin müssen in einer Anwendung mehrere Teil-von-Zerlegungen von Objekten verwaltet und für verschiedene Problemlösungsprozesse bereitgestellt werden. Ich möchte in diesem Zusammenhang von *multiplen Modellen* sprechen, da nicht nur die Sicht auf ein Objekt wichtig ist, sondern eine *Menge von Konzeptdefinitionen* betrachtet werden sollten, die für einen Aufgabenkontext eine Weltmodellierung und -strukturierung darstellen.

Für die Repräsentation der verschiedenen Zerlegungsformen werden jeweils unterschiedliche dreidimensionale geometrischen Repräsentationsformen benötigt (z.B. unterschiedliche CSG-Knoten von VANTAGE für die verschiedenen Tassenzerlegungsformen). Ich betrachte in dieser Arbeit aber zur Vereinfachung nur eine einzige „kanonische“ dreidimensionale geometrische Repräsentation. Die Verwaltung von sichten- bzw. modellspezifischen *zwei*dimensionalen geometrischen Repräsentations-

formen (Aufsichten, Seitenansichten usw.) in verschiedenen Modelle wird jedoch zur Visualisierungsgenerierung benötigt und durch HAMVIS unterstützt (siehe die Rolle `has-projective-form` aus dem Grundmodell und den Begriff der „modellspezifischen Rolle“).

Modelle sind also nicht nur Namensräume zur Strukturierung einer Wissensbasis (s.o.), sondern definieren auch einen Sichtenmechanismus für einzelne Objekte. HAMVIS rechnet aus der dreidimensionalen Formbeschreibung je nach Modell automatisch eine zweidimensionale geometrische Repräsentation aus (siehe Abschnitt 3.1.5). Wir werden in den folgenden Abschnitten sehen, daß Modelle selbst wieder Gegenstand von Schlußfolgerungsprozessen sein können. In ähnlicher Weise wie z.B. auch Struss die Bestimmung eines adäquaten Modells für ein Diagnoseproblem als Inferenzaufgabe beschreibt [296], möchte ich die Bestimmung eines Modells für eine Visualisierung als ein Inferenzproblem behandeln.[81] Die Basis der Modellauswahl soll jedoch bei HAMVIS auf der Basis von Schlußfolgerungen auf konzeptueller Ebene erfolgen, da konkrete Informationen für Domänenobjekte zur Entwicklungszeit einer Anwendung nicht notwendigerweise vorliegen. Die folgenden Abschnitte erläutern dieses anhand verschiedener Beispiele. Ich möchte zunächst einmal zur Fortsetzung der Illustration des XKL-Beispiels auf die Definition von anwendungsspezifischen Konzepten im Domänenmodell `xkl-model` genauer eingehen.

3.1.4 Basismodell für XKL

Wie schon erwähnt, importiert das `xkl-model` die aus dem HAMVIS-Grundmodell exportierten Namen. Ein Domänenmodell bildet einen eigenen Namensraum (siehe die Definition in Abbildung 54) und exportiert einige der definierten Konzepte und Rollen. Die Abbildungen 67, 68 und 69 illustrieren die Basisdefinitionen für die XKL-Anwendung, die vom Systementwicklungsteam aufgestellt wurden. Die Syntax der Beschreibungsformen entspricht dem KRSS-Lisp-Standard für dateibasierte beschreibungslogische Repräsentationsformen. Mit der Deklaration `in-model` (eine HAMVIS-Erweiterung) wird das Modell angegeben, in dem die Deklarationen gelten. Die aus dem HAMVIS-Grundmodell importierten Namen sind zur besseren Unterscheidung kursiv gesetzt. Im Basismodell der Anwendung XKL werden z.B. die Basisrelationen `has-constituents` und `spatially-encloses` verwendet. Speziellere Unterrelationen (siehe Abbildung 56) werden in anderen Modellen verwendet. Ich komme später darauf zurück. Ein Basismodell dient dazu, eine Aktionenzerlegung zu erstellen. Es müssen also Konzepte usw. deklariert werden, mit denen die Parameter und Werte von Berechnungs- und Speicherfunktionen sowie auch die Aktionen des Benutzers einer Anwendung beschrieben werden können (siehe Kapitel 3.2).

Im Basismodell für XKL werden Anwendungsobjekte wie z.B. Küchen (Konzept `galley`), Waschräume (Konzept `lavatory`) oder Sitze (Konzept `seat`) oder die verschiedenen Gangarten (Unterkonzepte von `aisle`) von dem Konzept `cabin-object` subsumiert. Kabinenobjekte wiederum können in einem Kabinenrumpf (`cabin-body`) räumlich enthalten sein. Plazierungsbereiche werden als räumliche Regionen (`spatial-region`, siehe Abbildung 57) modelliert. Aspekte der Komponentenstruktur spiegeln sich auch in der Konzeptbeschreibung der Objekte wider (siehe `aircraft-component` und `cabin-component` als Unterkonzepte von `component` aus dem HAMVIS-Grundmodell).

81. Die dynamische Auswahl von Modellen für Problemlösungsaufgaben in Anwendungen wird auch in anderen Kontexten untersucht (z.B. [271]).

Ein Flugzeug (`aircraft`) besteht aus maximal einer Nase, einer Kabine und einem Heckteil (siehe Abbildung 68).

```
(in-model xkl-model)

(define-primitive-concept cabin-object spatial-object)

(define-primitive-role has-trolleys nil)
(define-primitive-concept trolley spatial-object)

(define-primitive-concept galley
  (and cabin-object
       physical-object
       (all has-trolleys trolley)))

(define-primitive-concept lavatory
  (and cabin-object
       physical-object))

(define-primitive-concept seat
  (and cabin-object
       physical-object))

(define-primitive-concept placement-area spatial-region)

(define-primitive-concept technically-possible-placement-area
  placement-area)

(define-primitive-concept aisle (and cabin-object spatial-region))
(define-primitive-concept cross-aisle aisle)
(define-primitive-concept passageway aisle)

(define-primitive-concept aircraft-component component)

(define-primitive-concept cabin-component component)

(define-primitive-concept cabin-body
  (and physical-object
       cabin-component
       (all spatially-encloses cabin-object)))

(define-primitive-concept cabin-door
  (and physical-object cabin-component))

(define-primitive-role has-cabin-body has-constituents)
(define-primitive-role has-cabin-doors has-constituents)

(define-primitive-concept cabin
  (and physical-object
       aircraft-component
       (all has-constituents cabin-component)
       (all has-cabin-body cabin-body)
       (at-most 1 has-cabin-body)
       (all has-cabin-doors cabin-door)))
```

Abbildung 67. Definition des Basis-Domänenmodells `xkl-model` (Fortsetzung in Abbildung 68).

```
(define-primitive-concept tail-component component)

(define-primitive-concept tail-body
  (and physical-object tail-component))
(define-primitive-concept fuselage-body
  (and physical-object tail-component))

(define-primitive-role has-tail-body has-constituents)
(define-primitive-role has-fuselage-body has-constituents)

(define-primitive-concept tail
  (and physical-object
       aircraft-component
       (all has-constituents tail-component)
       (all has-tail-body tail-body)
       (at-most 1 has-tail-body)
       (all has-fuselage-body tail-body)
       (at-most 1 has-fuselage-body)))

(define-primitive-concept nose-component component)
(define-primitive-concept nose-body
  (and physical-object nose-component))
(define-primitive-concept cockpit-windows
  (and physical-object nose-component))

(define-primitive-role has-nose-body has-constituents)
(define-primitive-role has-cockpit-windows has-constituents)

(define-primitive-concept nose
  (and physical-object
       aircraft-component
       (all has-constituents nose-component)
       (all has-nose-body nose-body)
       (at-most 1 has-nose-body)
       (all has-cockpit-windows cockpit-windows)
       (at-most 1 has-cockpit-windows)))

(define-primitive-role has-nose has-constituents)
(define-primitive-role has-cabin has-constituents)
(define-primitive-role has-tail has-constituents)

(define-primitive-concept aircraft
  (and physical-object
       (all has-constituents aircraft-component)
       (all has-nose nose)
       (at-most 1 has-nose)
       (all has-cabin cabin)
       (at-most 1 has-cabin)
       (all has-tail tail)
       (at-most 1 has-tail)))
```

Abbildung 68. Fortsetzung der Definition des Domänenmodells `xkl-model`.

```
(in-model xkl-model)

(define-geometric-object (aircraft uac) aircraft)
(define-geometric-object (nose uac-nose) nose)
(define-geometric-object (cabin uac-cabin) cabin)
(define-geometric-object (tail uac-tail) tail)
(define-geometric-object (nose-body uac-nose-body) nose-body)
(define-geometric-object (cabin-body uac-cabin-body) cabin-body)
(define-geometric-object (tail-body uac-tail-body) tail-body)
...

(state (related uac-nose uac-nose-body has-constituents))
(state (related uac-nose uac-cockpit-windows has-constituents))

(state (related uac-cabin uac-cabin-body has-constituents))
(state (related uac-cabin uac-cabin-door-1 has-landmarks))
(state (related uac-cabin uac-cabin-door-2 has-landmarks))
(state (related uac-cabin uac-cabin-door-3 has-landmarks))
(state (related uac-cabin uac-cabin-door-4 has-constituents))
(state (related uac-tail uac-tail-body has-constituents))

(state (related uac uac-nose has-constituents))
(state (related uac uac-cabin has-constituents))
(state (related uac uac-tail has-constituents))
```

Abbildung 69. Fortsetzung des Domänenmodells `xkl-model`.

Im Modell `xkl-model` sind einige Domänenobjekte schon zur Entwicklungszeit bekannt (`uac` steht für „uncustomized aircraft“). Andere Objekte wie z.B. Kabinenobjekte (Instanzen von `galley`, `lavatory`, oder `seat`) werden jedoch erst durch Funktionen zur Laufzeit erzeugt. Die Deklarationsform

```
(define-geometric-object (concept-name individual-name)
  geometric-representation)
```

erzeugt ein Individuum *`individual-name`* vom Konzept *`concept-name`* und setzt dieses Individuum über die Relation `has-canonical-geometric-representation` zu dem geometrischen Objekt *`geometric-representation`* in Beziehung. Der untere Teil der Abbildung deklariert die Konstituentenstruktur der statisch bekannten Objekte.

Mit der Explizierung von geometrischen Beziehungen durch ABox-Aussagen haben wir schon die Verschmelzung von logisch-relationaler und geometrischer Sicht diskutiert. Die Verschmelzung von geometrischer und logischer Sicht wird in besonderem Maße durch die von HAMVIS eingeführten Modelle unterstützt. Die geometrische Sicht wird durch die Klasse eines Modells ausgedrückt. Die logische Sicht ergibt sich durch die Konzeptdefinitionen im Namensraum des Modells.

3.1.5 Modelle als Objekte erster Klasse: Modellklassen

Ein Modell kann auf der einen Seite als Namensraum für beschreibungslogische Deklarationen verwendet werden, zum anderen stellt ein Modell Dienste zur Berechnung von zweidimensionalen geometrischen Daten (projektiven Formen) aus der dreidimensionalen geometrischen Repräsentation von

Domänenobjekten bereit. Hierzu wird das Prinzip der Vererbung ausgenutzt. Abbildung 70 zeigt die Hierarchie der Modellklassen.

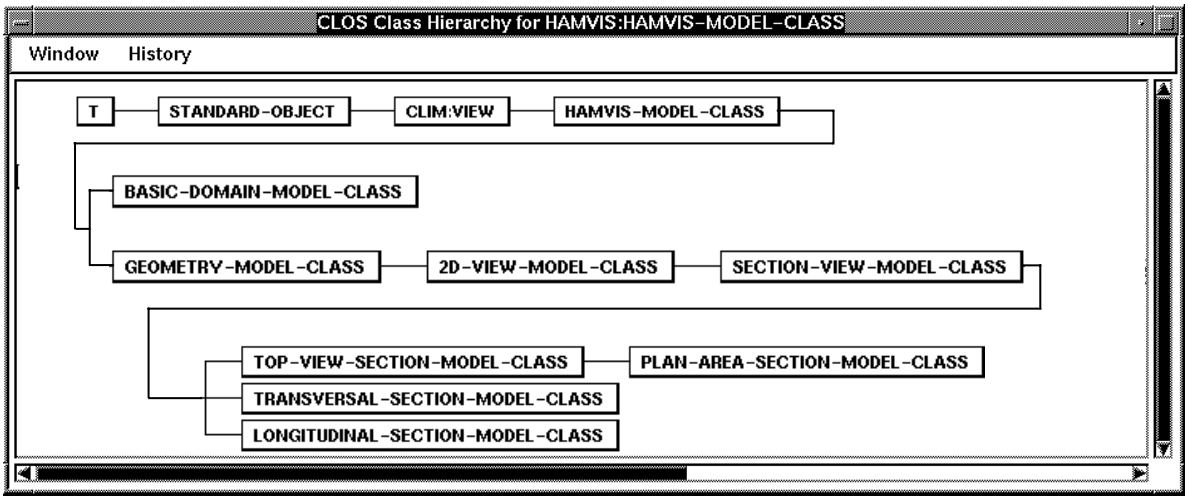


Abbildung 70. Modellklassenhierarchie.

Das Basismodell der mit HAMVIS modellierten Anwendung muß von der Klasse `basic-domain-model-class` sein. Genau diese Klasse wurde bei der Definition von `xkl-model` in Abbildung 54 angegeben. In diesem Modell werden nur dreidimensionale geometrische Daten betrachtet. Unterhalb der Klasse `geometry-model-class` werden projektive Abbildungen der dreidimensionalen Daten unterstützt. Jedes Modell enthält hierzu Informationen über die erforderliche Kameraposition (siehe die Definitionen in Abbildung 71).

```
(defclass 2d-view-model-class
          (geometry-model-class)
  ((camera :initarg :camera :accessor model-camera)))

(defgeneric model-projective-form-concept (2d-view-model-class))

(defclass section-view-model-class
          (2d-view-model-class)
  ())

(defclass longitudinal-section-model-class
          (section-view-model-class)
  ()
  (:default-initargs :camera (default-lateral-section-camera)))

(defclass top-view-section-model-class
          (section-view-model-class)
  ()
  (:default-initargs :camera (default-top-view-camera)))

(defclass plan-area-section-model-class
          (top-view-section-model-class)
  ())
```

Abbildung 71. Ausschnitt aus der Klassenbeschreibung für Modellklassen.

Mit Hilfe der Kameraspezifikation erzeugt HAMVIS unter Zuhilfenahme der Algorithmen von VANTAGE aus den 3D-Daten entsprechende 2D-Abbildungen zu erzeugen. Je nach Modellklasse erfolgt dieses über das Kameraobjekt aus der Sicht, die durch den Namen der Klasse angedeutet wird.

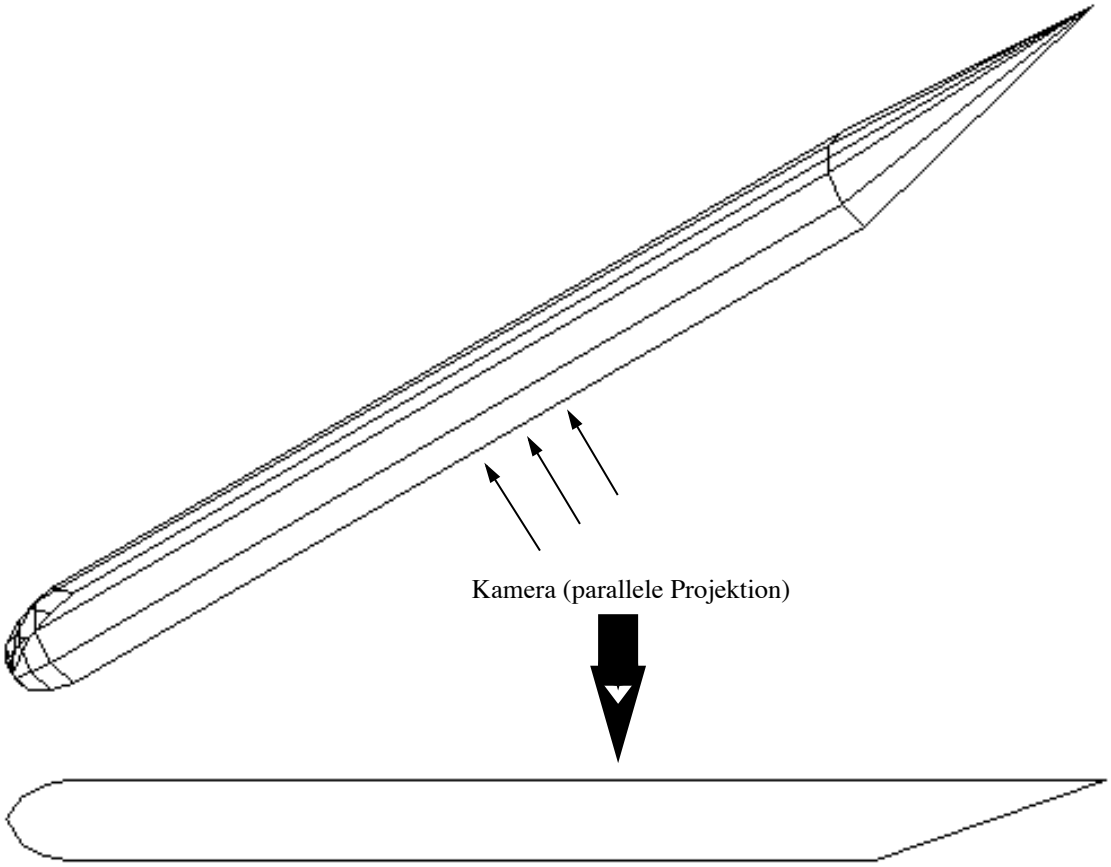


Abbildung 72. Beispiel für einen Longitudinalschnitt der Flugzeugs mit entsprechender 2D-Projektion.

Für alle Unterklassen von `section-model-class` wird eine parallele Projektion verwendet. Perspektivische Projektionen sind aus Sicht von VANTAGE ebenfalls möglich, werden aber in dieser Arbeit nicht näher untersucht. Die Abbildung 72 zeigt ein Beispiel für einen Longitudinalschnitt durch das Flugzeug aus Abbildung 59 mit entsprechender Projektion in einem Modell der Klasse `longitudinal-section-model-class`.

Modelle sind zentral für HAMVIS und werden als Objekte reifiziert. Sie haben zur Entwicklungszeit eine Bedeutung für die

- Verwaltung von Namensräumen, für die
- Verwaltung von sichtenspezifischen geometrischen Informationen und erlauben
- Schlußfolgerungen über den Inhalt und die Komposition von Visualisierungen.

Die Basisidee der Inhaltsfestlegung von HAMVIS besteht darin, jedem Ausgabefenster ein Modell zuzuordnen. Wenn zur Laufzeit z.B. die Konstituenten eines Objekts gezeigt werden, so ermittelt die von HAMVIS generierte Laufzeitkomponente alle Füller der modellspezifischen Rolle `has-constituents`. Zur Entwicklungszeit muß ein Modell bestimmt werden, das die notwendigen Visuali-

sierungsanforderungen erfüllt. Anforderungen sind z.B. gegeben durch eine notwendige Modellierungsgranularität, d.h. Objekte bestimmter Basiskonzepte müssen als Konstituenten auftreten, oder Subsumtionsbedingungen für die Konstituentenkonzepte (z.B. Überlappungsfreiheit) müssen gelten.

```
(define-model (plan-area-section-model-class
                floor-section-model)
    (:use hamvis-upper-model xkl-model)
  (:shadow aircraft-component
           cabin-object
           galley
           trolley
           lavatory
           seat
           ...)
  (:export aircraft-component
           cabin-object
           galley
           trolley
           lavatory
           seat
           ...))

(define-model (plan-area-section-model-class
                construction-floor-section-model
                floor-section-model)
  (:use hamvis-upper-model xkl-model floor-section-model)
  (:shadow aircraft-component
           cabin-object
           galley
           trolley
           lavatory
           seat
           placement-area
           ...)
  (:export aircraft-component
           cabin-object
           galley
           trolley
           lavatory
           seat
           placement-area
           ...))
```

Abbildung 73. Beispiele für die Deklaration zweier Modelle für die XKL-Anwendung.

Es wurde schon darauf hingewiesen, daß Modelle auch für die Ausgabe von graphischen Objekten zur Laufzeit der Anwendung eine entscheidende Rolle spielen. Hierzu erben Modellklassen von der Klasse `clim:view` (siehe Abbildung 70). Eine Sicht im Sinne von CLIM steuert die Ausgabe und bestimmt das Erscheinungsbild von graphischen Objekten auch bei der Eingabe von Werten (siehe die Ausführungen über die Konzepte von CLIM in Kapitel 2.1.4 und insbesondere den Abschnitt „Präsentation und Akzeptierung von Werten“). Bei der graphischen Ausgabe von Objekten zur Laufzeit der Anwendung wird auf die „richtigen“, im Modell relevanten geometrischen Daten zugegriffen. Wie schon erläutert, kann ein Objekt in verschiedenen Visualisierungen mit verschiedenen Sichten gezeigt

werden, da geometrische Daten für verschiedene Projektionen in modellspezifischen Unterrollen verwaltet werden. Für die Anwendung XKL werden durch den Oberflächendesigner z.B. die beiden Modelle aus Abbildung 73 deklariert.

Es werden zwei Modelle `floor-section-model` und `construction-floor-section-model` von der Klasse `plan-area-section-model-class` definiert. Sie importieren Konzept- und Rollennamen aus `hamvis-upper-model` und `xkl-model`.

Ich möchte in diesem Zusammenhang noch einmal auf die Übersicht in Abbildung 52 verweisen. Während das Basismodell für die Aktionenzerlegung benötigt wird und dementsprechend Konzept- und Rollendefinitionen enthält, die geeignet sind, sowohl Anwendungsfunktionen als auch Benutzeraktionen zu beschreiben, sind in den hier definierten Modellen Konzepte und Rollen für die Visualisierungsgenerierung definiert. Der Begriff „Modell für die Visualisierung“ darf nicht mißverstanden werden. Wenn für die Implementierung von Anwendungsfunktionen zweidimensionale geometrische Daten benötigt werden, können sie selbstverständlich auch den Visualisierungsmodellen entnommen werden. Visualisierungsmodelle können nach der Aktionenmodellierung definiert werden (vgl. Abbildung 52). Da in diesem Kapitel die Definition von Modellen besprochen wird, ziehe ich die Beispiele für die Definition von Visualisierungsmodellen vor (die Erläuterung der Aktionenzerlegung erfolgt in Kapitel 3.2).

Die Idee hinter der Modellierung aus Abbildung 73 besteht darin, das Modell `floor-section-model` für allgemeine „Überblicke“ und Referenzsysteme zu verwenden, während `construction-floor-section-model` für die Konstruktion der Inneneinrichtung gedacht ist. Für Überblicke sind Plazierungsbereiche irrelevant, sie sind in der für Konstruktion benötigten „Feinheit“ nur im Modell `construction-floor-section-model` deklariert (für eine formale Definition des Begriffs siehe unten). Für die Besprechung der konkreten Definitionen in den Modellen müssen noch einige Erweiterungen von KRSS erläutert werden, mit denen der Oberflächenentwickler festlegen kann, welche Konzepte Basiskonzepte sind und welche nicht.

3.1.6 Charakterisierung von Konzepten: Basiskonzepte und Primärkonzepte

In dem in dieser Arbeit betrachteten XKL-Beispiel wurden Konzepte verwendet wie z.B. `cabin-object` (siehe Abbildung 68). Ein Konzept wie `cabin-object` kann als kategorialer Oberbegriff aufgefaßt werden. Der Begriff wird gebildet, weil auf den darunterliegenden Primärkonzepten Handlungen ausgeführt werden (z.B. werden Kabinenobjekte in der Kabine lokalisiert, verschoben usw.). Erst für Spezialisierungen wie z.B. `galley` oder `lavatory` sind allerdings konkrete bzw. eindeutige räumliche Daten verfügbar. Man beachte hierzu die Konzeptdefinition von `cabin-object` aus Abbildung 67. Durch die Definition von `cabin-object` als Unterkonzept von `spatial-object` wird die Existenz genau einer geometrischen Repräsentation zwar zugesichert (siehe die Terme für die `at-least`- und `at-most`-Einschränkungen der Rolle `has-canonical-geometric-representation` in Abbildung 56), das heißt aber noch nicht, daß eine konkrete zweidimensionale Projektion tatsächlich vorhanden ist bzw. bestimmt werden kann. Auch eine Konzeptdefinition wie z.B.

```
(define-primitive-concept cabin-object
    (and spatial-object
         (all has-projective-form projective-form)
         (at-least 1 has-projective-form)))
```

ändert hieran im Prinzip nichts. Eine solche Definition definiert, daß es *konsistent* ist, Instanzen von `cabin-object` über die Rolle `has-projective-form` mit einer Instanz von `projective-form` in Beziehung zu setzen. Welcher *konkrete* Füller zu einer Instanz von `cabin-object` in Beziehung steht, bzw. ob dieser Füller überhaupt als Instanz bestimmt werden kann (oder muß), ist damit nicht ausgedrückt! Existenzaussagen im Sinne der Beschreibungslogik dienen zur Modellierung von unvollständigem Wissen, erzwingen also *nicht* die tatsächliche Berechnung von Rollenfüllern.[82] Um die Berechnung von Füllern zu veranlassen, wurde KRSS-CLASSIC um einen speziellen Konzeptbeschreibungsterm erweitert. Die Berechnung eines Füllers für eine Rolle `r` wird durch den Term `(at-least 1 (k r))` ausdrückbar (`k` steht für „known"). Wenn ein Individuum z.B. eine Instanz von `galley` ist, so soll nicht nur die Existenz eines Füllers der Rolle `has-projective-form` als Konsistenzkriterium etabliert werden, sondern der Füller soll auch tatsächlich berechnet werden. Konzeptterme der Art `(at-least` *`n-fillers`* `(k` *`role-name`*`))` werden nicht für die TBox-Klassifikation verwendet (hier genügt der Konzeptterm `(at-least` *`n-fillers role-name`*`)`). Terme der Art `(at-least` *`n-fillers`* `(k` *`role-name`*`))` können im Konsequenzteil einer Regel stehen. Die Deklaration heißt abgekürzt „K-Kardinalitätsrestriktion" (oder kürzer K-Deklaration).

Zur vereinfachten Definition von Regeln wurde KRSS-CLASSIC wiederum erweitert. Die Definition von KRSS-Regeln der Form `(define-rule` *`rule-name concept-name`* `(and` *`concept-expression`* `...))` kann direkt bei der Konzeptdefinition erfolgen. Die Syntax der erweiterten Konzeptdefinition sieht wie folgt aus:

```
(define-[primitive-]concept concept-name
   concept-expression
   [ (:asserted-concepts concept-expression ...) ]
   [ (:categorial-superconcept-p boolean) ] )
```

Durch die Option `:asserted-concepts` wird eine Implikationsregel für die ABox erzeugt und intern verwaltet. Eine Definition der Semantik von Regeln dieser Art wurde von Donini et al. in [76] über epistemische Operatoren angegeben. In dieser Arbeit wird auch der als Erweiterung von KRSS-CLASSIC realisierte K-Operator als epistemischer Operator formal definiert. Es ist allerdings zu beachten, daß in der von mir realisierten Erweiterung von KRSS-CLASSIC der K-Operator nur in den Konsequenzteilen von Regeln verwendet werden darf und daher nicht für die Konzeptklassifikation verwendet wird.

Die Option `:categorial-superconcept-p` markiert ein Konzept als kategoriales Oberkonzept. Für alle kategorialen Oberkonzepte gilt: Die Unterkonzepte schließen sich wechselseitig aus und Individuen werden garantiert von einem der Unterkonzepte subsumiert. HAMVIS sieht spezielle Darstellungsattribute zur Unterscheidung der Unterkonzepte vor. Ich komme später im Zusammenhang mit der Zerlegung von Objekten in verschiedenen Modellen noch einmal auf die Bedeutung dieser Option zu sprechen (siehe Abschnitt 3.1.8).

---

82. In einigen (tableaubasierten) Beweisern erfolgt die Füllerberechnung allerdings automatisch, jedoch ist dieses nicht zwingend notwendig, wie z.B. das System CLASSIC zeigt. Außerdem ist die automatische Generierung von Individuen aus praktischer Sicht nicht unkritisch. Ein Beispiel: `(define-concept large-stadium (and stadium (at-least 50000 seats)))`.

Nach der Betrachtung dieser theoretischen Aspekte möchte ich das XKL-Beispiel wieder aufgreifen. Eine Konzeptdefinition von `galley` könnte also wie folgt aussehen:

```
(define-concept galley
    spatial-object
  (:asserted-concepts (at-least 1 (k has-projective-form))))
```

Wenn ein Objekt von `galley` subsumiert wird, so gilt `(at-least 1 (k has-projective-form))`. Es wird also eine K-Kardinalitätsrestriktion verwendet, durch die die Füllerberechnung erzwungen werden kann.

Zu beachten ist noch, daß die projektive Form erst dann bestimmt werden kann, wenn ein Füller der Rolle `has-canonical-geometric-representation` bekannt ist. Auch dieses kann deklariert werden. Die volle Syntax der Termbeschreibung lautet:

`(at-least` *`n-fillers`* `(k` *`r`* `/ (at-least` *`n1`* `(k` *`r1`*`)) (at-least` *`n1`* `(k` *`r2`*`)) ...))`

Der Schrägstrich steht für „unter der Bedingung, daß“. Die Berechnung der Füller für die Rolle `r` wird also erst dann erfolgen, wenn *`n1`* Füller für *`r1`* usw. bekannt sind. Für `(at-least 1 (k` *`r`*`))` kann als Abkürzung `(k` *`r`*`)` geschrieben werden.

In unserem konkreten Beispiel ergibt sich für `galley`:

```
(define-primitive-concept galley
    cabin-object
  (:asserted-concepts (at-least 1 (k has-projective-form
                                     / (k has-canonical-geometric-representation)))))
```

Falls nicht genügend Füller berechnet werden (können), so heißt ein Objekt *unvollständig*. Hierzu einige Definitionen:

Definition: Unvollständigkeit eines Individuums bezüglich einer Rolle

> Ein Individuum, für das `n` Rollenfüller einer Rolle `r` mittels der K-Kardinalitätsrestriktion garantiert worden sind, wobei jedoch weniger Füller berechnet werden können, heißt *unvollständig* bezüglich der Rolle `r`.

Definition: Basiskonzept

> Im Kontext von HAMVIS wird ein Konzept als *Basiskonzept in einem Modell* bezeichnet, wenn für die Rolle `has-projective-form` in dem Modell die Berechnung *genau* eines Füllers garantiert wird. Wenn die Berechnung von `has-projective-form` von weiteren Berechnungen abhängt, müssen auch für die Rollen Berechnungen für die Füller garantiert werden. Es ergibt sich ein rekursives Testschema.

Im HAMVIS-Grundmodell wird die Bestimmbarkeit einer kanonischen geometrischen Repräsentation für `spatial-object` zugesichert. Im Gegensatz zu der in Abbildung 56 vereinfacht dargestellten Form wird folgende erweiterte Definition verwendet:

```
(define-primitive-concept spatial-object
    (and um-object
         ...)
  (:asserted-concepts (at-least 1 (k has-canonical-geometric-representation))))
```

Durch die Deklaration von räumlichen Objekten mit der Form `define-geometric-object` (s.o.) wird einer entsprechender Füller angegeben.

Basiskonzepte sind die Voraussetzung für die Visualisierung eines Objekts in einem geometrischen Modell der Klasse `geometry-model-class`. Einige Basiskonzepte werden noch in besonderer Weise charakterisiert:

Definition: Primärkonzept

> Als *Primärkonzepte* werden diejenigen Basiskonzepte bezeichnet, die nicht von einem anderen Basiskonzept subsumiert werden.

Primärkonzepte werden in einer von HAMVIS generierten Visualisierung durch Zeichenattribute graphisch unterschieden. Die Konzeptcharakterisierungen „Primärkonzept“ bzw. „Basiskonzept“ haben weiterhin eine Bedeutung für die Bestimmung der Zerlegung von Objekten in einem Modell aufgrund von konzeptuellen Informationen (zur Entwicklungszeit!). Dieses wird durch Betrachtung der Konzeptdefinitionen in den oben deklarierten XKL-Modelle `floor-section-model` und `construction-floor-section-model` deutlich. Bevor jetzt die Definition von Visualisierungsmodellen dieser Art näher beschrieben wird, möchte ich noch auf die Beziehungen zwischen Modellen eingehen.

3.1.7 Beziehungen zwischen Modellen

Für einige Namen soll im Modell `construction-floor-section-model` eine „verfeinerte“ Definition gegenüber `xkl-model` gegeben werden, d.h. es soll in `construction-floor-section-model` ein Konzept definiert werden, das den gleichen Namen wie in `xkl-model` trägt. Beziehungen zwischen Modellen wie z.B. „feiner“, sind für die Visualisierungsgestaltung relevant. HAMVIS versucht, die Anforderungen an eine Visualisierung im „gröbsten“ möglichen Modell zu erfüllen.[83]

Die Feiner-Beziehung zwischen Konzepten ist wie folgt definiert:

Definition: Feiner-Relation zwischen Konzepten

> Ein primitives Konzept `c` aus einem Modell `m2` heißt *feiner* als ein Konzept gleichen Namens aus einem Modell `m1` genau dann, wenn gilt `m1:c` subsumiert `m2:c` und es gibt eine Regel mit `m1:c` im Antezedenzteil und `m2:c` im Konsequenzteil.

Ein `m2:c` ist also aufgrund der Subsumtionsbeziehung in jedem Fall ein `m1:c`, ist aber durch zusätzliche Konzeptterme weiter eingeschränkt. Wenn ein `m1:c` als Individuum erzeugt wird, so wird durch die Regel auch `m2:c` etabliert. Mit anderen Worten: Es ist ausreichend, Objekte im Basisdomänenmodell zu erzeugen. Durch ABox-Schlußfolgerungen erfolgt die weitere Instantiierung in den verschiedenen Modellen. Man beachte, daß die in der Definition angesprochene Regel nur für die ABox-Subsumtion relevant ist, d.h. `m1:c` und `m2:c` sind nicht zyklisch definiert (und auch nicht identisch).

---

83. Die Namen für eine feinere Konzeptdefinition in einem Modell werden in der Modelldeklaration unter der Option `:shadow` in der Modelldeklaration aufgeführt (vgl. die Modelldefinition in Abbildung 73).

Die Feiner-Relation zwischen Konzepten kann auch auf Modelle ausgedehnt werden.

Definition: Feiner-Relation zwischen Modellen

> Gegeben seien zwei Modelle `m1` und `m2` der gleichen Modellklasse. Wenn für *alle* in `m1` sichtbaren benannten Konzepte `c` gilt: `c` ist auch in `m2` sichtbar, und `m2:c` ist feiner als `m1:c`, genau dann heißt `m1` *feiner* als `m2`.

Die Feiner-Relation zwischen Modellen kann automatisch hergeleitet werden. Bei `define-model` kann allerdings auch angegeben werden, ob eine Feiner-Relation zwischen zwei Modellen erwartet wird. Hierzu werden die weniger feinen Modelle innerhalb der Deklaration mit `define-model` nach dem Modellnamen angegeben (siehe die Definition von `construction-floor-section-model` aus Abbildung 73 für ein Beispiel).

HAMVIS bietet folgendes Repräsentationskonstrukt als Erweiterung von KRSS-CLASSIC an, mit dem die Feiner-Beziehung zwischen Konzepten syntaktisch übersichtlich definiert werden kann:

```
(define-concept-refinement (concept-name model-name)
   concept-expression
   [ (:asserted-concepts concept-expression) ]
   [ (:categorial-superconcept-p boolean) ] )
```

Durch diese Deklaration wird ein Konzept *concept-name* deklariert, das vom gleichnamigen Konzept aus *model-name* subsumiert wird. Das „neue“ Konzept *concept-name* wird als primitiv deklariert. Zu dem Konzept *concept-name* aus dem Modell *model-name* wird durch diese Deklaration automatisch eine Regel hinzugefügt, die die Implikation *model-name*:*concept-name* => *concept-name* repräsentiert, d.h. in diesem Modell ist *concept-name* feiner modelliert. Die Bedingungen in *concept-expression* gelten zusätzlich für alle Instanzen von *concept-name*.

Zur einfachen Definition von modellspezifischen Unterrollen dient das Konstrukt:
`(define-role-refinement (role-name model-of-super-role)).`

Diese Deklaration steht als Abkürzung für
`(define-primitive-role role-name :parent model-of-super-role:role-name).`

Die hier beschriebenen Konstrukte werden in der Definition des Modells `floor-section-model` verwendet. Die Abbildungen 74 und 75 zeigen die entsprechenden Deklarationen.

```
(in-model floor-section-model)

(define-role-refinement (has-projective-form hamvis-upper-model))
(define-role-refinement (has-form-representation-constituents
                         hamvis-upper-model))

(define-concept-refinement (aircraft-component xkl-model)
    (and (all has-projective-form xy-projective-form)
         (at-least 1 has-projective-form)
         (at-most 1 has-projective-form)))

(define-concept-refinement (nose-body xkl-model)
    (and (all has-projective-form xy-projective-form)
         (at-least 1 has-projective-form)
         (at-most 1 has-projective-form))
  (:asserted-concepts
    (at-least 1 (k has-projective-form
                   / (k has-canonical-geometric-form-representation)))))

(define-concept-refinement (nose xkl-model)
    (and aircraft-component
         (all has-form-representation-constituents nose-body)
         (at-least 1 has-form-representation-constituents)
         (at-most 1 has-form-representation-constituents))
  (:asserted-concepts
    (at-least 1 (k has-projective-form
                   / (k has-canonical-geometric-form-representation)))
    (at-least 1 (k has-form-representation-constituents
                   / (k has-canonical-geometric-form-representation)))))

(define-concept-refinement (cabin-body xkl-model)
    (and (all has-projective-form xy-projective-form)
         (at-least 1 has-projective-form)
         (at-most 1 has-projective-form))
  (:asserted-concepts
    (at-least 1 (k has-projective-form
                   / (k has-canonical-geometric-form-representation)))))

(define-concept-refinement (cabin xkl-model)
    (and aircraft-component
         (all has-form-representation-constituents cabin-body)
         (at-least 1 has-form-representation-constituents)
         (at-most 1 has-form-representation-constituents))
  (:asserted-concepts
   (at-least 1 (k has-projective-form
                  / (k has-canonical-geometric-form-representation)))
   (at-least 1 (k has-form-representation-constituents
                  / (k has-canonical-geometric-form-representation)))))
```

Abbildung 74. Erster Teil der Definition des Modells `floor-section-model`.

```
(in-model floor-section-model)

(define-concept-refinement (tail-body xkl-model)
    (and (all has-projective-form xy-projective-form)
         (at-least 1 has-projective-form)
         (at-most 1 has-projective-form))
  (:asserted-concepts
   (at-least 1 (k has-projective-form
                  / (k has-canonical-geometric-form-representation)))))

(define-concept-refinement (tail xkl-model)
    (and aircraft-component
         (all has-form-representation-constituents tail-body)
         (at-least 1 has-form-representation-constituents)
         (at-most 1 has-form-representation-constituents))
  (:asserted-concepts
   (at-least 1 (k has-projective-form
                  / (k has-canonical-geometric-form-representation)))
   (at-least 1 (k has-form-representation-constituents
                  / (k has-canonical-geometric-form-representation)))))

(define-concept-refinement (aircraft xkl-model)
    (and (all has-projective-form xy-projective-form)
         (at-least 1 has-projective-form)
         (at-most 1 has-projective-form)

         (all has-nose nose)
         (at-least 1 has-nose)
         (at-most 1 has-nose)

         (all has-cabin cabin)
         (at-least 1 has-cabin)
         (at-most 1 has-cabin)

         (all has-tail tail)
         (at-least 1 has-tail)
         (at-most 1 has-tail)

         (all has-form-representation-constituents aircraft-component)
         (at-least 3 has-form-representation-constituents)
         (at-most 3 has-form-representation-constituents))
  (:asserted-concepts
   (at-least 1 (k has-projective-form
                  / (k has-canonical-geometric-form-representation)))
   (at-least 3 (k has-form-representation-constituents
                  / (k has-canonical-geometric-form-representation)))))
```

Abbildung 75. Zweiter Teil der Definition des Modells `floor-section-model`.

Im Gegensatz zu dem Modell `xkl-model` wird in `floor-section-model` nicht die Relation `has-constituents` verwendet (vgl. die Konzeptdefinition von `aircraft` in Abbildung 68), sondern die Unterrelation `has-form-representation-constituents`. Die in den Abbildungen 74 und 75 verwendeten Deklarationsformen zeigen auch, wie die K-Deklarationen zur Festlegung der Konstituentenstruktur verwendet werden. Der nächste Abschnitt geht auf die Objektzerlegung genauer ein.

3.1.8 Objektzerlegungsgraphen: Modellspezifische partonomische Konzepthierarchien

Für die Bestimmung des Inhalts von Visualisierungen sind Zerlegungsbeziehungen zwischen Primärkonzepten relevant (siehe Abschnitt 3.1.1). Zur Entwicklungszeit kann die Bestimmung der Zerlegung von Objekten in Komponenten i.a. nur auf der Basis von konzeptuellen Informationen erfolgen (siehe hierzu die Definitionen in Abschnitt 3.1.6). Die aufgrund der konzeptuellen Information bekannte Zerlegung von Objekten wird für jedes Modell in einem sog. Objektzerlegungsgraphen explizit repräsentiert. Jedes Primärkonzept bildet einen Knoten im Objektzerlegungsgraphen. Für jeden Knoten, d.h. für jedes Primärkonzept, wird nun überprüft, ob aufgrund der Konzeptbeschreibung die Füller der Teil-von-Relationen `has-form-representation`, `spatially-encloses`, `has-constituents`, `has-landmarks` aus dem HAMVIS-Grundmodell Basiskonzepte sind. Falls ja, werden die den Basiskonzepten zugeordneten Primärkonzepte bestimmt und die entsprechenden Knoten des Objektzerlegungsgraphen werden als Söhne des Ausgangsknotens eingetragen. Die Kanten tragen je nach Relation entsprechende Beschriftungen. In Abbildung 76 wird der hierzu ein Beispiel präsentiert. Zur Darstellung von Objektzerlegungsgraphen wird der für HAMVIS entwickelte modellspezifische Konzeptinspektor verwendet. In Abbildung 76 wird zur Illustration der Objektzerlegungsgraph des Modells `floor-section-model` gezeigt.

In der Tabelle, die in der linken unteren Ecke plaziert ist, stehen die deklarierten Modelle zur Auswahl bereit. Als aktuelles Modell ist `floor-section-model` markiert. Das linke Graphikfenster zeigt die Zerlegungshierarchie der im aktuellen Modell sichtbaren Primärkonzepte bzgl. der Rollen unterhalb von `has-constituent` (`FRC` steht für `has-form-representation-constituents`, `I` stünde für `spatially-encloses`, `C` für `has-constituents`, `L` für `has-landmarks`). In dem Teilfenster für die taxonomische Hierarchie sind die „sichtbaren“ Unterkonzepte von `spatial-object` dargestellt. `Spatial-object` wurde aus dem HAMVIS-Grundmodell importiert.

Zu beachten ist, daß für die Bestimmung des Objektzerlegungsgraphen nicht die textuelle Beschreibung eines Konzeptes betrachtet wird. Zur Berechnung der Objektzerlegungsgraphen müssen durch HAMVIS die (modellspezifischen) Konzepthierarchien nach der TBox-Klassifikation noch traversiert werden. Bei Betrachtung der Definition von `aircraft` in Abbildung 75 fällt z.B. auf, daß als Beschreibung der Einschränkungen für Füller der Rolle `has-form-representation-constituents` das Konzept `aircraft-component` angegeben ist. Dieses wiederum ist in Abbildung 74 deklariert. Es handelt sich nicht um ein Basiskonzept (s.o.). Erst unterhalb von `aircraft-component` finden sich Basiskonzepte (bzw. Primärkonzepte, siehe die Definitionen von `nose`, `cabin` und `tail` in Abbildung 74 und Abbildung 75). Zu beachten ist auch, daß im Modell `floor-section-model` Konzepte wie `galley`, `lavatory` oder `seat` nicht als Basiskonzepte ausgewiesen sind, sie treten daher nicht als Zerlegung von `cabin-body` im Objektzerlegungsgraphen auf, werden also in keinem Fall in einer Visualisierung eingezeichnet, die auf diesem Modell basiert.

Im Modell `construction-floor-section-model` werden die angesprochenen Konzepte entsprechend feiner modelliert. In diesem Modell wird für die Konzepte die Verfügbarkeit von geometrischen Informationen zugesichert. Dieses Modell wird in den Abbildungen 77, 78 und 79 dargestellt.

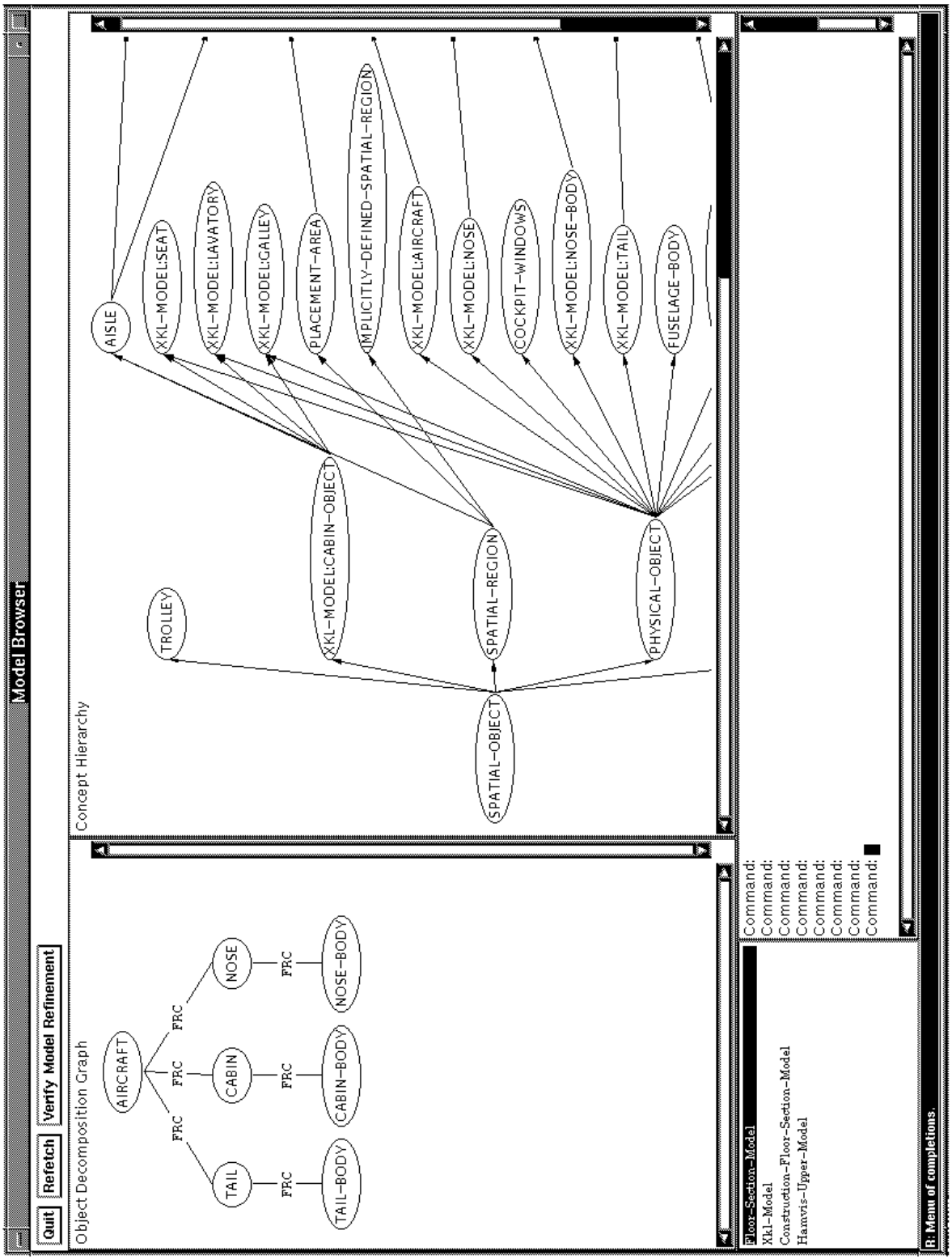


Abbildung 76. Modellspezifische Darstellung der partonomischen und taxonomischen Struktur von Konzepten im Modell `floor-section-model`.

```
(in-model construction-floor-section-model)

(define-role-refinement (has-projective-form hamvis-upper-model))
(define-role-refinement (has-form-representation-constituents
                         hamvis-upper-model))

(define-concept-refinement (cabin-object xkl-model)
    (and (all has-projective-form xy-projective-form)
         (at-most 1 has-projective-form)
         (at-least 1 has-projective-form)
         (at-most 1 spatially-enclosed-by)
         (fillers spatially-enclosed-by uac-cabin-body))
  (:categorial-superconcept-p t)
  (:asserted-concepts non-overlapped-spatial-object))

(define-concept-refinement (galley xkl-model)
    (and cabin-object
         (at-least 1 has-trolleys)
         (at-least 1 has-projective-form))
  (:asserted-concepts
   (at-least 1 (k has-projective-form
                  / (k has-canonical-geometric-form-representation)))))

(define-concept-refinement (lavatory xkl-model)
    (and cabin-object
         (at-least 1 has-projective-form))
  (:asserted-concepts
   (at-least 1 (k has-projective-form
                  / (k has-canonical-geometric-form-representation)))))


(define-concept-refinement (seat xkl-model)
    (and cabin-object
         (at-least 1 has-projective-form))
  (:asserted-concepts
   (at-least 1 (k has-projective-form
                  / (k has-canonical-geometric-form-representation)))))

(define-concept-refinement (placement-area xkl-model)
    (and cabin-object
         (at-least 1 has-projective-form))
  (:asserted-concepts
   (at-least 1 (k has-projective-form
                  / (k has-canonical-geometric-form-representation)))))

(define-concept-refinement (aircraft-component floor-section-model)
    (and (all has-projective-form xy-projective-form)
         (at-least 1 has-projective-form)
         (at-most 1 has-projective-form)))
```

Abbildung 77. Modell `construction-floor-section-model` (Teil 1).

```
(in-model construction-floor-section-model)

(define-concept-refinement (nose-body floor-section-model)
    (and (all has-projective-form xy-projective-form)
         (at-least 1 has-projective-form)
         (at-most 1 has-projective-form))
  (:asserted-concepts
   (at-least 1 (k has-projective-form
                  / (k has-canonical-geometric-form-representation)))))

(define-concept-refinement (nose floor-section-model)
    (and aircraft-component
         (at-least 1 has-projective-form)
         (all has-form-representation-constituents nose-body)
         (at-least 1 has-form-representation-constituents)
         (at-most 1 has-form-representation-constituents))
  (:asserted-concepts
   (at-least 1 (k has-projective-form
                  / (k has-canonical-geometric-form-representation)))
   (at-least 1 (k has-form-representation-constituents
                  / (k has-canonical-geometric-form-representation)))))

(define-concept-refinement (cabin-body floor-section-model)
    (and (all has-projective-form xy-projective-form)
         (at-least 1 has-projective-form)
         (at-most 1 has-projective-form)
         (all spatially-encloses cabin-object))
  (:asserted-concepts
   (at-least 1 (k has-projective-form
                  / (k has-canonical-geometric-form-representation)))))

(define-concept-refinement (cabin floor-section-model)
    (and aircraft-component
         (all has-form-representation-constituents cabin-body)
         (at-least 1 has-form-representation-constituents)
         (at-most 1 has-form-representation-constituents))
  (:asserted-concepts
   (at-least 1 (k has-projective-form
                  / (k has-canonical-geometric-form-representation)))
   (at-least 1 (k has-form-representation-constituents
                  / (k has-canonical-geometric-form-representation)))))

(define-concept-refinement (tail-body floor-section-model)
    (and (all has-projective-form xy-projective-form)
         (at-least 1 has-projective-form)
         (at-most 1 has-projective-form)
         (at-least 1 has-canonical-geometric-form-representation))
  (:asserted-concepts
   (at-least 1 (k has-projective-form
                  / (k has-canonical-geometric-form-representation)))))
```

Abbildung 78. Modell `construction-floor-section-model` (Teil 2).

```
(in-model construction-floor-section-model)

(define-concept-refinement (tail floor-section-model)
    (and aircraft-component
         (all has-form-representation-constituents tail-body)
         (at-least 1 has-form-representation-constituents)
         (at-most 1 has-form-representation-constituents))
 (:asserted-concepts
   (at-least 1 (k has-projective-form
          / (k has-canonical-geometric-form-representation)))
   (at-least 1 (k has-form-representation-constituents
          / (k has-canonical-geometric-form-representation)))))

(define-concept-refinement (aircraft floor-section-model)
    (and (all has-projective-form xy-projective-form)
         (at-least 1 has-projective-form)
         (at-most 1 has-projective-form)

         (all has-nose nose)
         (at-least 1 has-nose)
         (at-most 1 has-nose)

         (all has-cabin cabin)
         (at-least 1 has-cabin)
         (at-most 1 has-cabin)

         (all has-tail tail)
         (at-least 1 has-tail)
         (at-most 1 has-tail)

         (all has-form-representation-constituents aircraft-component)
         (at-least 3 has-form-representation-constituents)
         (at-most 3 has-form-representation-constituents))
  (:asserted-concepts
   (at-least 1 (k has-projective-form
          / (k has-canonical-geometric-form-representation)))
   (at-least 1 (k has-form-representation-constituents
          / (k has-canonical-geometric-form-representation)))))
```

Abbildung 79. Modell `construction-floor-section-model` (Teil 3).

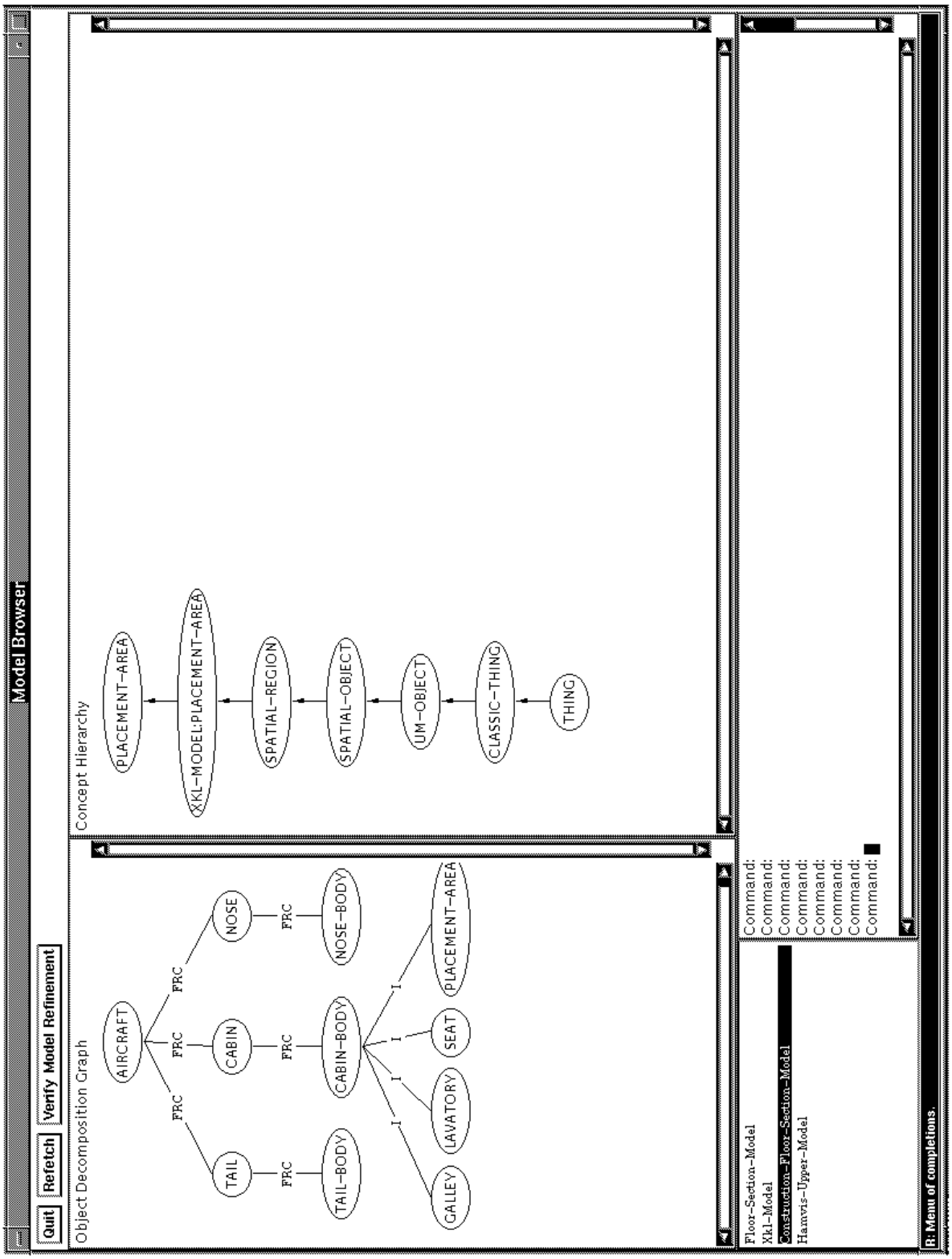


Abbildung 80. HAMVIS-Modellinspektor mit dem Objektzerlegungsgraphen für das Modell `construction-floor-section-model`.

Abbildung 80 zeigt für das Modell `construction-floor-section-model` den berechneten Objektzerlegungsgraphen. Zu beachten ist die Zerlegung von Objekten, die von `cabin-body` subsumiert werden. In der Konzeptedefinition von `cabin-body` in Abbildung 78 ist als Zerlegung über die Relation `spatially-encloses` das Konzept `cabin-object` angegeben. Gemäß der Konzeptdefinition von `cabin-object` (siehe Abbildung 77) handelt es sich um ein kategoriales Oberkonzept, das zur einem Unterkonzept spezialisiert werden muß (`galley`, `lavatory`, `seat`, `placement-area`). Die Unterkonzepte sind in diesem Fall auch Primärkonzepte, sie werden in der Darstellung des Objektzerlegungsgraphen in Abbildung 80 angezeigt. In diesem Fall wird durch die Präsentationsstrukturierungskomponente von HAMVIS zur Entwicklungszeit dafür gesorgt, daß bei der Ausgabe (zur Laufzeit) vier verschiedene Fälle unterschieden werden. Für jedes kategoriale Unterkonzept werden unterschiedliche Zeichenattribute verwendet, die zur Entwicklungszeit festgelegt werden (siehe Kapitel 3.4).

Mit dem HAMVIS-Modellinspektor kann auch eine erwartete Feiner-Relation zwischen Modellen verifiziert werden. Falls eine erwartete Feiner-Beziehung nicht gefunden werden kann, wird mit Hilfe der Erklärungsfähigkeiten von CLASSIC versucht, die Ursache darzustellen.

3.1.9 Motivation der Repräsentationskonstrukte: Charakterisierung der Schlußfolgerungen auf der Wissensebene

Für die Visualisierung von Anwendungsobjekten müssen geometrische Daten vorliegen bzw. bestimmt werden. In dem hier betrachteten Beispiel werden sie aus einer dreidimensionalen Formbeschreibung abgeleitet. Wir haben gesehen, wie die Daten mit Hilfe von Modellen verwaltet werden. Weiterhin wurde deutlich, daß es notwendig ist, *zur Entwicklungszeit* sicherzustellen, daß zur Laufzeit die erforderlichen Geometrie-Repräsentationen auch tatsächlich vorliegen. Dieses wird auch für noch nicht bekannte Instanzen auf der Konzeptebene durch das Konzeptschema `(at-least 1 (k has-projective-form / (k` *`role-name`*`) ...))` zugesichert. Konzepte, die dieses Konzept „implizieren“ (entweder im logischen Sinne oder durch Regeln), heißen in einem Modell *`model-name`* Basiskonzepte genau dann, wenn in dem Modell genau ein Füller für *`model-name`*`:has-projective-form` existieren darf.

Mit dem Objektzerlegungsgraphen wird die Zerlegung von Objekten beschrieben, die durch Primärkonzepte subsumiert werden. Je nach Modell werden unterschiedliche Objektzerlegungsgraphen bestimmt. Wir haben in Abbildung 80 ein Beispiel betrachtet, in dem eine Kabine bzw. ein Kabinenrumpf im Modell `construction-floor-section-model` Küchen, Waschräume, Sitze und Plazierungsbereiche enthält (Relation `spatially-encloses`). Im Modell `floor-section-model` aus Abbildung 76 ist dieses scheinbar nicht der Fall.

Ich möchte an dieser Stelle noch einmal diskutieren, welchen Vorteil in diesem Kontext die beschreibungslogische Repräsentation bietet. Nehmen wir einmal an, zur Wissensrepräsentation wäre ein einfaches Framesystem mit Objekten und Slots verwendet worden. Ob z.B. bestimmte Objekte in einem anderen Objekt enthalten sind, könnte dadurch gekennzeichnet werden, daß bei der entsprechenden Frame-Deklaration ein Slot „spatially-contains“ vorgesehen wird oder nicht. Das Problem ist nur: Eine „Verhinderung“ von Assoziationen durch fehlende Slots, wie z.B. in Frame-basierten Systemen üblich, ist auf der Wissensebene ohne jede Semantik!

Man beachte, daß beschreibungslogische Repräsentationssysteme nicht eine speicherorientierte Objektbeschreibung mit „Slots“ unterstützen, sondern eine logisch-relationale, d.h. solange ein Relationentupel im Sinne der Logik konsistent ist, kann ein solches Tupel in die Relation „eingetragen“ werden. Die Menge der Slots eines Objektes als die Menge der verfügbaren Informationen auf der Wissensebene zu betrachten, ist nicht unproblematisch, da auf der Programmierebene Assoziationen zwischen Objekten nicht nur über Slots, sondern auch über Hashtabellen, Felder usw. aufgebaut werden können. Eine fehlende bzw. nicht vorgesehene Repräsentation einer Assoziation mit einem Slot heißt nicht, daß auf der Wissensebene bestimmte Assoziationen nicht gebildet werden können. Eine Einschränkung der Form `(and (at-least 1 r) (all r C))`, die z.B. in einer Konzeptbeschreibung auftritt, als Slot `r` mit einem entsprechenden Füller zu interpretieren, ist ebenfalls durch nichts gerechtfertigt. Nichtsdestotrotz erfolgt dieses in vielen Systemen unmotiviert und ohne Analyse auf der Wissensebene! Die Menge der Slots im Sinne der Frame-basierten Wissenspräsentation ist nicht direkt durch Rollenrestriktionen der Form `(and (at-least 1 r) (all r C))` der Beschreibungslogik definierbar.

Der Vorteil der Beschreibungslogik liegt darin, daß man in diesem System nicht der Versuchung erliegen kann, fehlende Slots zur Speicherung dahingehend zu interpretieren, daß bestimmte Assoziationen nicht gebildet werden können. Wenn in einer Beschreibungslogik eine Assoziation von Objekten bestimmter Konzepte über eine Relation verhindert werden soll, so muß dieses explizit als Inkonsistenz deklariert werden, beispielsweise durch den Konzeptterm: `(at-most 0 spatially-encloses)`. Ohne eine solche Einschränkung kann nicht verhindert werden, daß zur Laufzeit der Anwendung ein Objekt, das vom Konzept `floor-section-model:cabin-body` subsumiert wird, zu einem Objekt vom Konzept `floor-section-model:galley` über die Relation `spatially-encloses` in Beziehung gesetzt werden kann. Sagen wir, das Objekt heiße `galley-17`, und es seien auch entsprechende zweidimensionale geometrische Form verfügbar. Dabei ist es auch noch möglich, daß `galley-17` zu genau einem geometrischen Objekt in der Relation `has-projective-form` in Beziehung steht. Aus Sicht der Beschreibungslogik ist eine solche Assertion ohne die obige Einschränkung konsistent. Es nicht einmal ausgeschlossen, daß auch ein Objekt vom Konzept `aircraft` zu `galley-17` über die Relation `spatially-encloses` in Beziehung gesetzt wird. Aus Sicht der Logik ist auch dieses konsistent und wird nicht „verhindert“. Auf der Wissensebene können Objekte über alle deklarierten Rollen in Beziehung gesetzt werden, solange dieses mit Hilfe der ABox der Beschreibungslogik als konsistent bewiesen wird bzw. beweisbar ist.

Um die Assoziation von einer `aircraft`-Instanz über die Rolle `spatially-encloses` mit `galley-17` auszuschließen, müßten aus Sicht der Logik zu Konzepten wie etwa `floor-section-model:aircraft` weitere Terme wie z.B. `(at-most 0 floor-section-model:has-projective-form)` hinzugefügt werden, so daß eine Inkonsistenz abgeleitet werden könnte, wenn ein entsprechendes Tupel in die Relation eingetragen werden sollte. Dieses ist nicht nur unintuitiv, sondern auch vom Standpunkt der Modellerstellung problematisch, da bei Einführung einer neuen Rolle, ggf. viele Konzeptdefinitionen editiert werden müßten.

Im Kontext von HAMVIS wird nicht gefordert, daß Einschränkungen der Form `(at-most 0 r)` hinzugefügt werden. HAMVIS geht davon aus, daß *im schlimmsten Fall* gar keine Füller für `r` bekannt sind, wenn nicht die Füllerberechnung mittels eine K-Deklaration erzwungen wird.

Die hier beschriebene Bestimmung eines Objektzerlegungsgraphen für ein Modell kann also auf der Wissensebene als *bewußtes Ignorieren* gedeutet werden. In dem einem Modell werden bestimmte

Assertionen betrachtet, in einem anderen Modell werden sie bewußt ignoriert, d.h. auch wenn zur Laufzeit entsprechende Füller auftreten, werden sie bei der Darstellung nicht betrachtet. Ein Modell kann also in einem gewissen Sinne auch als *Informationsfilter* interpretiert werden. Was heißt das genau?

Das Modell `xkl-model` importiert z.B. die Rolle `hamvis-upper-model:has-projective-form`, zu der in `floor-section-model` und `construction-floor-section-model` jeweils automatisch modellspezifische Unterrollen definiert werden.[84]

Ein Zugriff auf die Füller der Rolle `hamvis-upper-model:has-projective-form` zur Laufzeit der Anwendung liefert in jedem Fall die Vereinigung der Füller der Unterrollen, also sogar mehr Informationen. Doch was soll man damit anfangen? Welche der möglichen geometrischen Repräsentationen bilden die Grundlage für die Visualisierungsausgabe? Die Rolle `xkl-model::has-projective-form` ist bezüglich der Maximalkardinalität der Rollenfüllermenge nicht abgeschlossen. Mit der Definition von `construction-floor-section-model:galley` findet für die Unterrolle `construction-floor-section-model:has-projective-form` ein solcher Abschluß statt (Abbildung 77). Dieses gilt nicht im Modell `floor-section-model`. In diesem Modell ist das Konzept `galley` *kein* Basiskonzept, tritt also nicht im Objektzerlegungsgraphen auf. Im Modell `xkl-model` gibt es sogar kein einziges Basiskonzept. Der Objektzerlegungsgraph ist leer.

Instanzen, die in einem Modell nur von Konzepten subsumiert werden, die nicht Basiskonzepte sind, werden bei der Visualisierungsgenerierung bewußt ignoriert. Wenn also in einer Visualisierung Küchen oder Plazierungsbereiche etc. auftreten sollen, so muß das Modell `construction-floor-section-model` verwendet werden, da die Konzepte im Modell `floor-section-model` keine Basiskonzepte sind. Das Modell `xkl-model` wird in keinem Fall für die Visualisierungsgenerierung verwendet, die geometrischen Informationen sind ja auch mehrdeutig. Genauer ausgedrückt: Es kann zur Entwicklungszeit (auf der Ebene der Konzepte) nicht ausgeschlossen werden, daß die geometrischen Daten zur Laufzeit mehrdeutig sind.

Nehmen wir an, zur Entwicklungszeit wird das Modell `construction-floor-section-model` für die Darstellung von Objekten in einem Fenster festgelegt. Was ist die Konsequenz hieraus? Wenn z.B. die Bestandteile der Kabine dargestellt werden sollen, so werden die Füller der Rolle `spatially-encloses` bestimmt. Es werden aber nur diejenigen Objekte für die Darstellung verwendet, die von `galley`, `lavatory`, `placement-area` oder `seat` aus dem Modell `construction-floor-section-model` subsumiert werden, andere Objekte werden weggefiltert. Positiv ausgedrückt: Es wird mit der Zuordnung eines Modells zu einem Darstellungsfenster eine Menge von Prädikaten festgelegt, die die darzustellenden Objekte erfüllen müssen.

Ein Modell als Informationsfilter ist also gedeutet auf der Wissensebene eine *Menge von Prädikaten*, die für die darzustellenden Objekte zur Entwicklungszeit erfüllt sein müssen. Die Bedingungen sind durch den Objektzerlegungsgraphen definiert und gehen auf die Definition des Begriffs Basiskonzept zurück. Es muß zur Entwicklungszeit auf der Konzeptebene beweisbar sein, daß die Bedingungen zur Laufzeit für alle Instanzen gelten.

---

84. Es wäre natürlich nachteilig, wenn schon im HAMVIS-Grundmodell bei der Definition von `spatial-object` als Rollenrestriktion `(at-most 1 has-projective-form)` angegeben worden wäre (siehe Abbildung 56). Dann könnte es nur eine einzige projektive Form geben!

3.1.10 Berechnung von Rollenfüllern: Modellspezifische zweidimensionale geometrische Formen

In den vorigen Abschnitten wurde die Charakterisierung von Konzepten als Basiskonzepte bzw. Primärkonzepte mit Hilfe des Konzeptschemas `(at-least` *`n-fillers`* `(k has-projective-form /` `(k` *`role-name1`*`) ...))` ausgenutzt. Es wird zur Entwicklungszeit zugesichert, daß konkrete Füller zur Laufzeit auch tatsächlich berechnet werden. Hierzu werden intern geeignete Regeln deklariert. Die manuelle Definition der Regeln wäre recht komplex, da z.B. Zyklen bei dem Feuern der Regeln vermieden werden müssen usw. Die Erweiterungen von KRSS-CLASSIC stellen jedoch sicher, daß zur Berechnung der Rollenfüller für `has-projective-form` eine spezielle generische Funktion `compute-role-fillers` aufgerufen wird, sobald bekannt wird, daß ein Individuum von einem Basiskonzept subsumiert wird. Falls Objekte schon zur Entwicklungszeit bekannt sind (wie z.B. der Flugzeugrumpf), wird die Funktion auch schon zur Entwicklungszeit aufgerufen, sonst erfolgt der Aufruf zur Laufzeit der Anwendung.

Als Parameter für die Funktion `compute-role-fillers` werden das Objekt, für das ein Rollenfüller ermittelt werden soll, der Name der betreffenden Rolle aus dem K-Operator und das Modellobjekt selbst übergeben. Falls bei der Evaluierung dieser Funktion nicht genügend Werte geliefert werden, wird eine Warnung bezüglich der Unvollständigkeit des Objekts ausgegeben.[85]

HAMVIS stellt für die Modellklassen des HAMVIS-Grundmodells schon geeignete Methoden für `compute-role-fillers` bereit. Ein Beispiel wird in Abbildung 81 gezeigt.

```
(define-method compute-role-fillers ((ind spatial-object)
                                     (role-name (eql 'has-projective-form))
                                     (model geometry-model-class))
  (let ((result (compute-projective-form
                 (map-geometric-representation
                   (first (get-fillers ind
                            'has-canonical-geometric-form-representation
                            model))
                   (cl-name ind)
                   model)
                 ind
                 model)))
    (if (listp result)
        result
        (list result))))
```

Abbildung 81. Basismethode für die Berechnung von Rollenfüllern.

Für die innerhalb dieser Methode für `compute-role-fillers` evaluierte generischen Funktion `compute-projective-form` sind u.a. die Methoden aus Abbildung 82 definiert, die wiederum Dienste von VANTAGE verwenden. Das Konzept `projective-form` stammt aus dem HAMVIS-Grundmodell und wurde in Abbildung 64 eingeführt.

85. Auf Wunsch kann auch ein Fehler signalisiert werden.

```
(define-method compute-projective-form ((geometric-representation t)
                                        (object spatial-object)
                                        (model hamvis-model-class))
  nil)

(define-method compute-projective-form
               ((geometric-representation vantage:csgnode)
                (spatial-object spatial-object)
                (model 2d-view-model-class))
  (let* ((csgnode-name (vantage:vantage-object-name geometric-representation))
         (scene-name (gensym (symbol-name csgnode-name)))
         (image-name (gensym "IMAGE-"))
         (projective-form-ind-name (gensym "PROJECTIVE-FORM")))
    (vantage:scene* scene-name (list csgnode-name))
    (vantage:image* scene-name
                    (model-camera model)
                    :image-name image-name)
    (classic:cl-create-ind projective-form-ind-name 'projective-form)
    (classic:cl-ind-add `(fills has-2d-form-representation
                                ,(vantage:find-instance image-name)))
    (classic:cl-named-ind projective-form-ind-name)))
```

Abbildung 82. Methoden für `compute-projective-form` mit Zugriff aus die Algorithmen von VANTAGE.

Die Funktion `map-geometric-representation` berechnet geeignete Abbildungen der kanonischen geometrischen Repräsentation. In Abbildung 83 sind einige Methoden für die XKL-Anwendung gezeigt. Diese anwendungsspezifischen Methoden müssen durch das Systementwicklungsteam definiert werden. Es wird hier u.a. festgelegt, durch welche Ebene der Grundriß des Flugzeugs verläuft.

```
(define-method map-geometric-representation
               ((canonical-geometric-representation vantage:csgnode)
                (spatial-object (eql 'uac))
                (model plan-area-section-model-class))
  (vantage:find-instance (find-geometric-object 'aircraft-body-floor)))

(define-method map-geometric-representation
               ((canonical-geometric-representation vantage:csgnode)
                (spatial-object (eql 'uac))
                (model (eql (find-model 'construction-floor-section-model))))
  (vantage:find-instance (find-geometric-object 'aircraft-body-floor-fuselage)))
```

Abbildung 83. Anwendungsspezifische Methoden für die Berechnung von Schnitten.

Für die Verwaltung von geometrischen Daten werden also wiederum generische Funktionen und ein vererbungsbasiertes Aufrufen von Methoden durch objektorientierte Techniken ausgenutzt. Bei der Erzeugung von Instanzen werden die entsprechenden Methoden durch die K-Deklarationen automatisch aufgerufen, und es werden die berechneten Füller in die jeweilige Rolle `has-projective-form` in den betreffenden Modellen eingetragen.

.

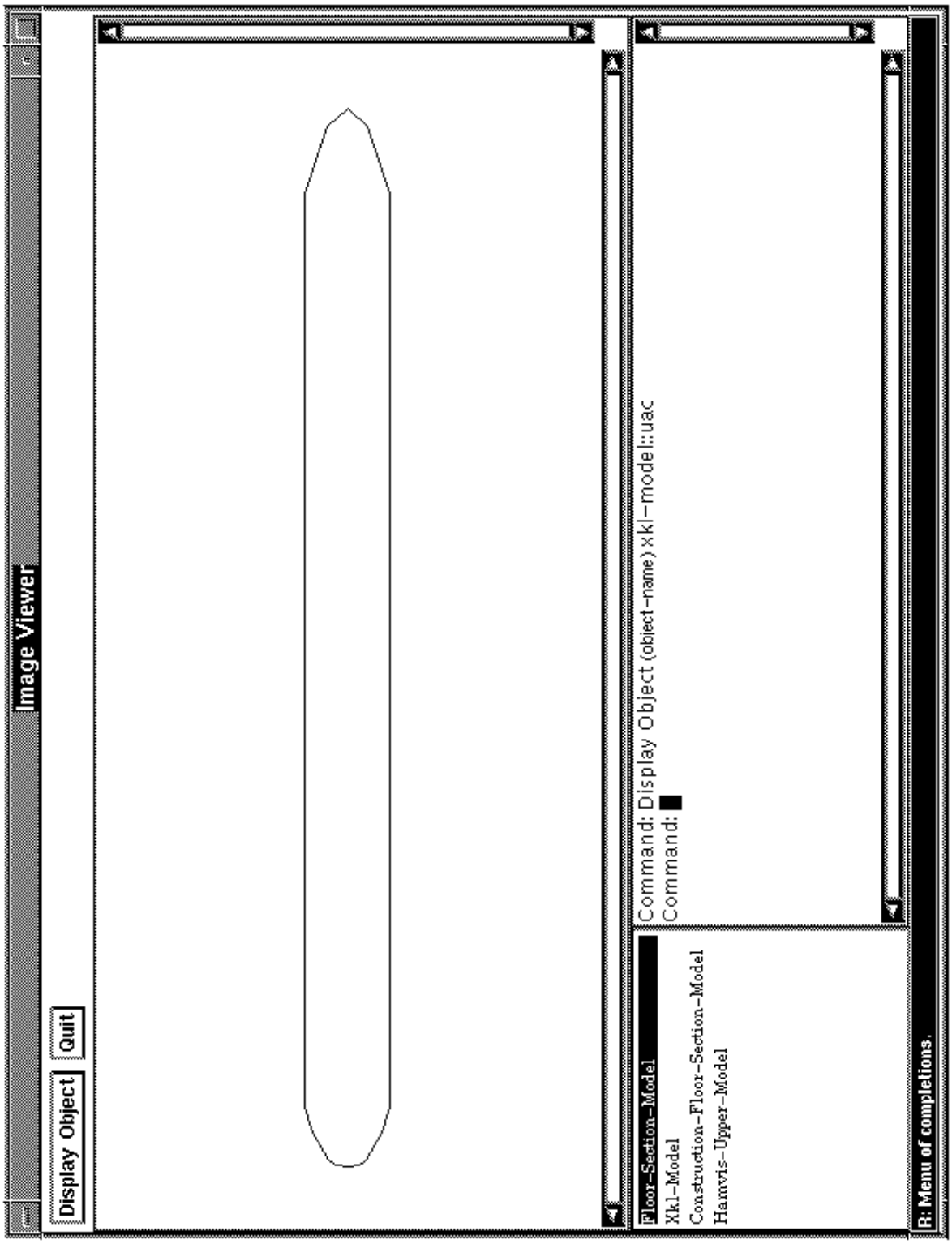


Abbildung 84. HAMVIS-Inspektor für visuelle Repräsentationen. Als aktuelles Modell ist `floor-section-model` ausgewählt.

HAMVIS ermöglicht also die modellspezifische Repräsentation der zweidimensionalen geometrischen Form (projektive Form). Bei der Entwicklung einer Anwendung können die vordefinierten Dienste von HAMVIS zur Berechnung von geometrischen Repräsentationen verwendet werden. Der Entwickler kann sich auf die inhaltliche Gestaltung von Visualisierungen konzentrieren und braucht nicht die Einzelheiten auf der Programmierebene zu beachten.

Für die Instanzen, die schon zur Entwicklungszeit bekannt sind (siehe Abbildung 69 für die Deklaration von `uac`), stellt HAMVIS einen Inspektor für die Darstellung der visuellen Repräsentation in verschiedenen Modellen bereit. Ich demonstriere hier einige Beispiele zur Illustration der von HAMVIS in der XKL-Anwendung bestimmten geometrischen ObjekteIn ähnlicher Weise wie bei dem oben gezeigten Modellinspektor kann das „aktuelle Modell“ z.B. mit der Maus aus der Tabelle im linken unteren Teil des Fensters ausgewählt werden. Die geometrischen Daten von Objekten werden im großen Teilfenster mit einer geeigneten Skalierung und Transformation angezeigt. Abbildung 85 zeigt das gleiche Objekt in einem anderen Modell.

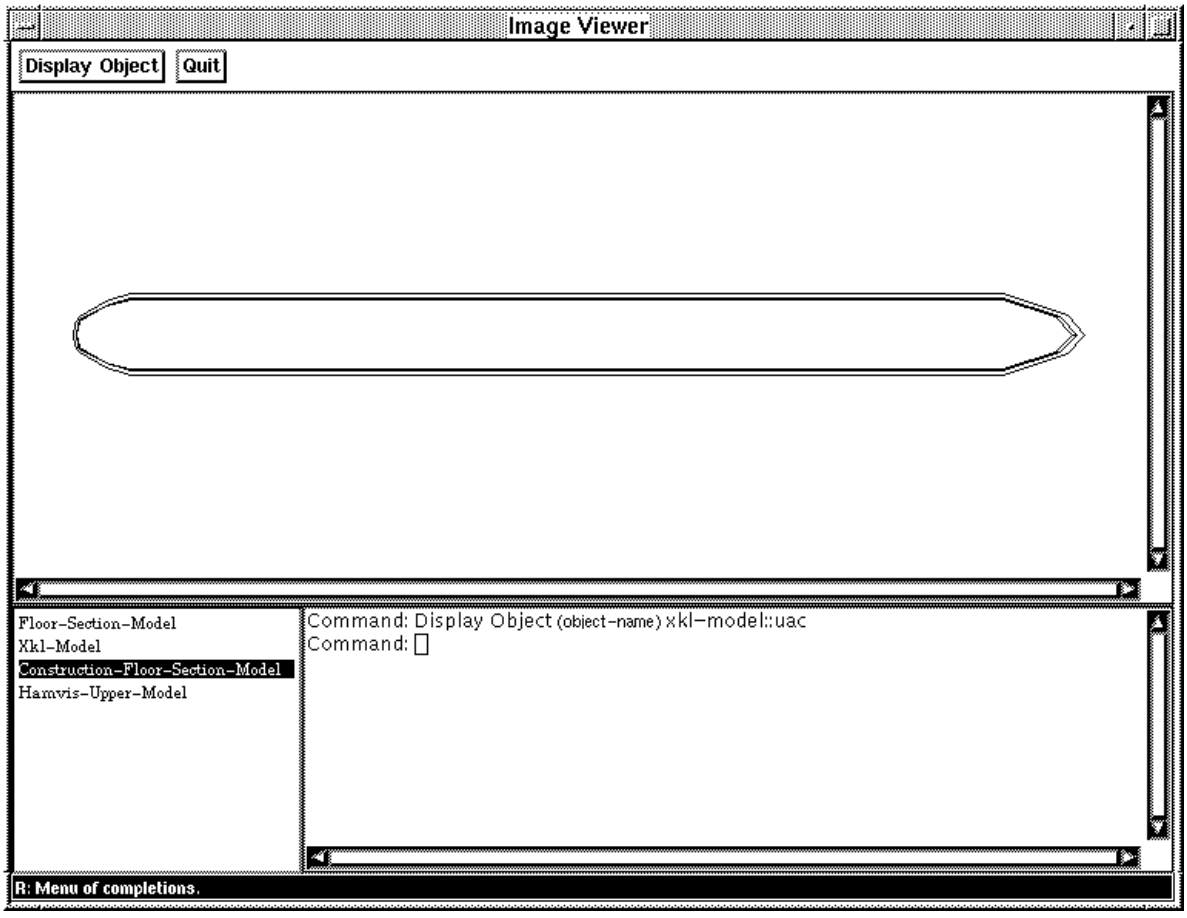


Abbildung 85. HAMVIS-Inspektor für visuelle Repräsentationen: Das gleiche Objekt im Modell `construction-floor-section-model`.

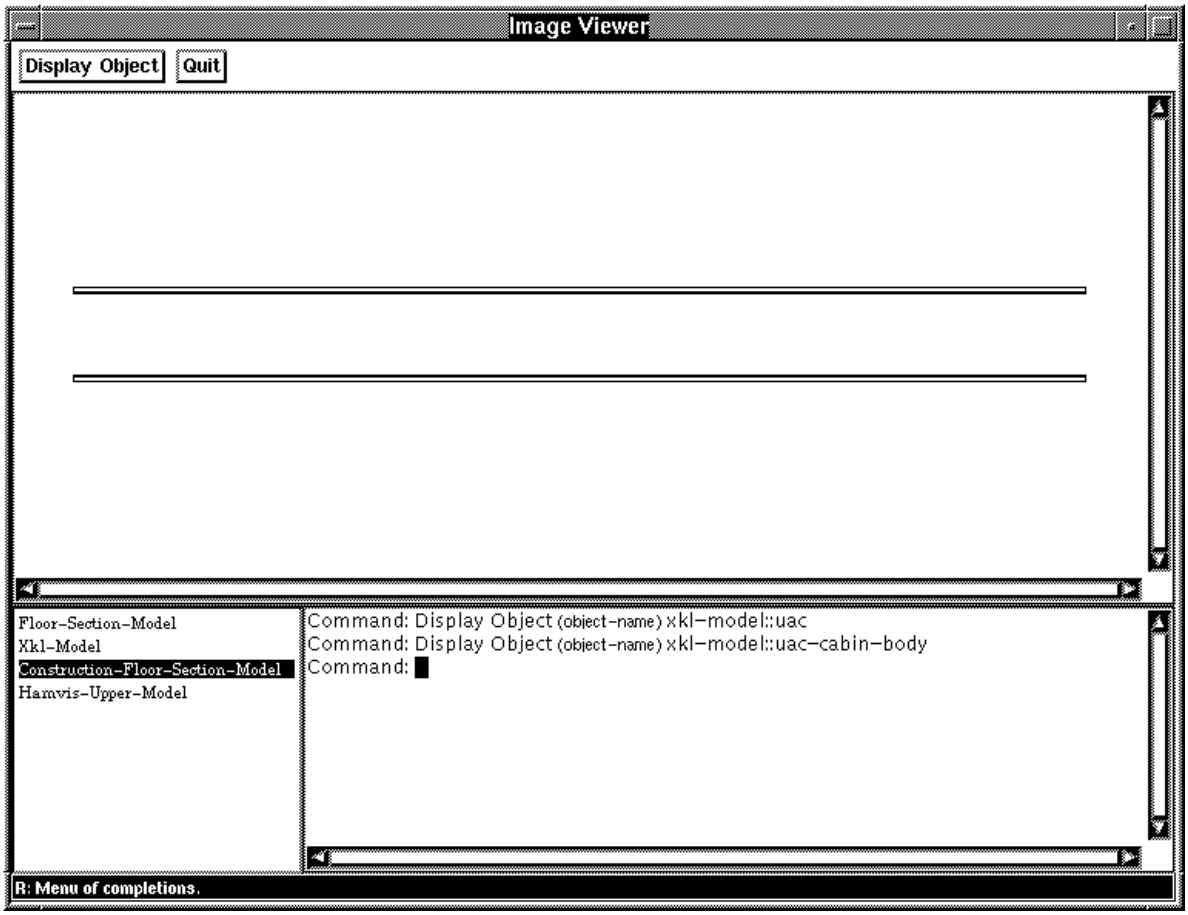


Abbildung 86. Darstellung eines Teilobjektes des in Abbildung 85 inspizierten Objekts.

In Abbildung 86 wird ein Objekt der Zerlegung von `uac` gezeigt: `uac-cabin-body`. Es handelt sich um den Mittelteil des Flugzeugrumpfes aus Abbildung 84. Siehe hierzu auch den Objektzerlegungsgraphen des Modells `construction-floor-section-model` aus Abbildung 80. Abbildung 87 verdeutlicht, daß die geometrische Repräsentation eines Objekts in einem anderen Modell durchaus unterschiedlich sein kann.

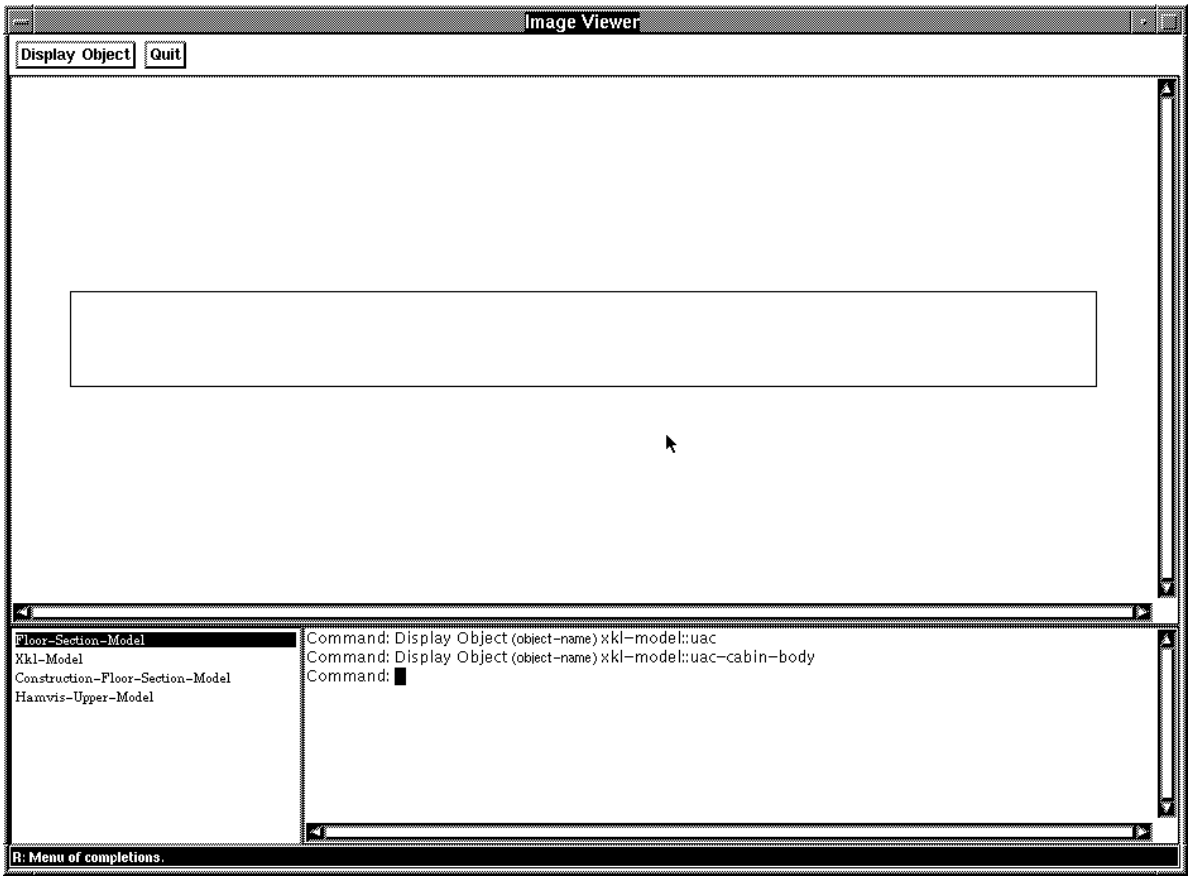


Abbildung 87. Darstellung des Objekts aus Abbildung 86 in dem Modell `floor-section-model`.

Es ist zu beachten, daß die visuelle Repräsentation des Ganzen (Abbildung 85) nicht durch einfache „Aneinandersetzung“ der Teilrepräsentationen entsteht (siehe die Abbildungen 86 und 87). Dies wäre auch wenig sinnvoll, da sonst die „Nahtstellen“ in der Gesamtdarstellung sichtbar wären. Es wird also für das Ganze ein spezielles graphisches Objekt berechnet, das aus der Flächen-Repräsentation des zugeordneten CSG-Knotens bestimmt wird (siehe Abbildung 60) und nicht aus der Komposition der zweidimensionalen Teilvisualisierungen hervorgeht.

3.1.11 Visualisierung mit geometrischen und nicht-geometrischen Modellen

Wenn bestimmte Objekte in einem Teilfenster dargestellt werden sollen, so muß diesem Fenster (zur Entwicklungszeit) ein bestimmtes Modell zugeordnet werden. Wir werden in späteren Kapiteln sehen, wie ein solches Modell ausgewählt wird. Mögliche Modelle sind durch die Aktionen festgelegt, die der Benutzer der Anwendung mit den visuellen Darstellungen durchführen soll.

In Kapitel 3.1.2 haben wir gesehen, wie durch den Namensraummechanismus für Modelle unterschiedliche geometrische Repräsentationen verwaltet werden können. Für geometrische Modelle (Modelle der Klasse `geometry-model-class`) wurde in Kapitel auch schon eine Zeichenmethode skizziert, in der ein Zugriff auf geometrische Informationen erfolgt. Allerdings kann einem Fenster nur dann ein bestimmtes Modell zugeordnet werden, wenn in dem Modell zur Entwicklungszeit garantiert werden kann, daß geometrische Informationen zur Laufzeit auch tatsächlich vorliegen. Zur

Laufzeit soll in einer bestimmten Interaktionssituation kein Fehler bezüglich fehlender Rollenfüller auftreten. Mit anderen Worten: Objekte müssen visualisierbar sein.

Definition: Direkte Visualisierbarkeit

> Anwendungsobjekte sind *direkt visualisierbar*, wenn es mindestens ein Modell gibt, in dem sie von einem Basiskonzept subsumiert werden. Genau die Modelle in denen eine zu präsentierende Instanz von einem (in dem Modell sichtbaren) Basiskonzept subsumiert wird, sind dann *Kandidaten* für die Gestaltung der Visualisierung (Ein- und Ausgaben).

Für Basiskonzepte müssen dann (zur Laufzeit) noch die entsprechenden Transformationen und Skalierungen berechnet werden, so daß die entsprechenden Objekte im Sichtfenster erscheinen. Dieses erfolgt durch vordefinierte Methoden für UIMS-spezifische Fensterklassen (siehe die HAMVIS-Ergänzungen zum UIMS in der Übersicht in Abbildung 52).

Nicht jede Aktion definiert aber Bedingungen, die die Kandidatenmenge der Modelle so weit einschränken, daß zur Darstellung von Objekten nur geometrische Modelle in Frage kommen (s.u.). Es können auch Modelle definiert werden, in denen Konzepte und Relationen nicht auf eine geometrische Modellierung ausgerichtet sind, sondern eher funktionale Zusammenhänge zwischen Objekten beschreiben (sogenannte funktionale Modelle). Relationen wie z.B. `has-projective-form` aus dem HAMVIS-Grundmodell können in einem solchen Modell eventuell nicht sichtbar sein, und damit kann es in einem solchen Modell auch keine Basiskonzepte geben.

Nehmen wir an, für die Darstellung einer Menge von Objekten O stehen zwei Modelle M_1 und M_2 als Kandidaten zur Verfügung, wobei M_1 eine Instanz von `geometry-model-class` und M_2 Instanz einer Modellklasse `functional-model-class` ist. Nehmen wir weiterhin an, daß durch Inferenzen über die Konzepte der Aktionen hergeleitet wird (s.u.), daß für die Darstellung der Objekte z.B. ein räumliches Referenzsystem R benötigt wird. Für das räumliche Referenzsystem R wird das Modell M_1 verwendet und daher wird es auch dem Fenster als Modell zugeordnet. Für die Darstellung der Objekte O kann auch das Modell M_2 verwendet werden, wenn es eine Abbildung von dem Modell M_2 in das Modell M_1 gibt. Die Objekte der Menge O sind *indirekt visualisierbar*, wenn bestimmte Bedingungen gelten.

Definition: Indirekte Visualisierbarkeit

> Sei ein Objekt in einem Modell M_2 nicht von einem Basiskonzept subsumiert. Ein Anwendungsobjekt ist *indirekt visualisierbar* in einem solchen Modell M_2, wenn es eine Abbildung der geometrischen Informationen von einem Modell M_1, in dem das Objekt von einem Basiskonzept subsumiert wird, auf das Modell M_2 gibt.

Abbildungen von geometrischen Modellen auf nicht-geometrische Modelle können auf der Ebene der Modellklasse definiert werden, z.B.:

```
(defmethod map-geometric-data ((2d-image vantage:image)
                               (model2 functional-model-class)
                               (model1 geometry-model-class))
  (get-bounding-box 2d-image)))
```

Eine Zeichenmethode für `present` kann auch für nicht-geometrische Modelle definiert werden.

```
(define-presentation-method present (individual
                                     (presentation-type spatial-object)
                                     (model functional-model-class)
                                     stream)
  (let* ((geometric-model-of-stream (get-model stream))
         (2d-image (map-geometric-data
                    (get-fillers individual
                                 'has-projective-form
                                 geometric-model-of-stream)
                    model
                    geometric-model-of-stream)))
    (draw-image stream individual 2d-image)))
```

Abbildung 88 zeigt ein Beispiel für eine solche Abbildung, in der statt der geometrischen Repräsentation nur das achsenparallele umschließende Rechteck für die Darstellung verwendet wird.[86]

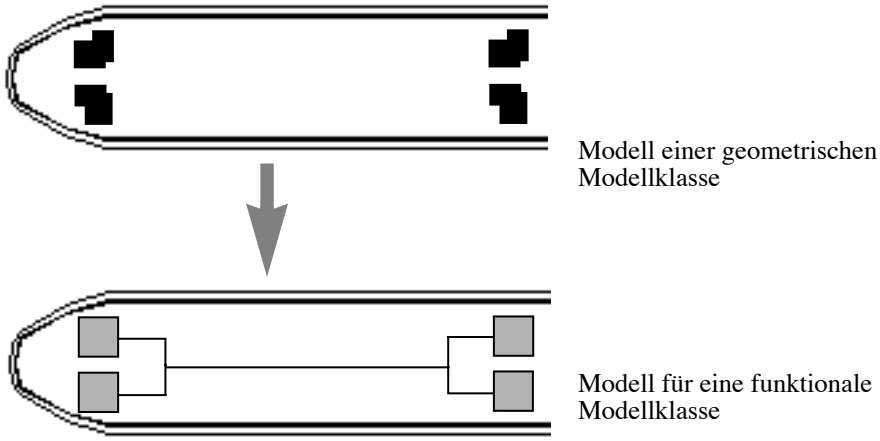


Abbildung 88. Beispiel für eine Abbildung zwischen Modellen: Übergang von geometrisch korrekter Darstellung zu einer schematischen Darstellung mit Umrißrechtecken und Verbindungslinien zur Darstellung von funktionalen Zusammenhängen.

In Kapitel 3.4 wird kurz angedeutet, wie auch anwendungspezifische visuelle Notationen in die Darstellung eingebettet werden können. Grundlage für die Darstellung im Kontext von HAMVIS sind aber die geometrischen Daten. Ich werde daher in den folgenden Kapiteln nur geometrische Modelle betrachten.

3.1.12 Zusammenfassung

In diesem Unterkapitel wurde der Begriff des *Modells* eingeführt. Modelle dienen als *Namensräume* für Konzeptdefinitionen, d.h. mit Hilfe von Modellen können gleichnamige Konzeptdefinitionen jeweils so detailliert modelliert werden, wie für die Aufgabe in dem Modell nötig. Es besteht nicht die Notwendigkeit, bei der Definition eines Konzeptes jede mögliche Verwendungsform vorherzusehen.

86. Umrißrechtecke werden automatisch für jede projektive Abbildung von VANTAGE verwaltet und können daher trivialerweise auch die Grundlage für die Implementierung von Berechnungs- und Speicherfunktionen bilden. Für viele Anwendungsprobleme wird aber eine adäquate, nicht-approximierende zweidimensionale oder sogar dreidimensionale Repräsentation der Geometriedaten benötigt, so daß ich hierauf nicht weiter eingehen möchte. Das HAMVIS-Grundmodell unterstützt durch VANTAGE eine dreidimensionale Modellierung und verwaltet über den Modellmechanismus auch zweidimensionale Projektionen.

Ein Modell kann durch die automatische Deklaration von Unterrollen (zu jeder in den Namensraum importierten Rolle) als abgegrenzter *Kontext* interpretiert werden. Die Bedeutung der aus dem HAM-VIS-Grundmodell stammenden Rollen für die Generierung von Visualisierungen wurde in diesem Abschnitt anhand von Beispielen erläutert. In Bezug auf die Definition von Basiskonzepten ist z.B. die Rolle `has-projective-form` besonders wichtig.

Motiviert durch kognitionswissenschaftliche Erkenntnisse wurden verschiedene Konzeptcharakterisierungen eingeführt. Die Berechnung von unterscheidbaren Standardwerten (defaults) für Zeichenattribute wird auf der konzeptuellen Ebene durch Konzeptcharakterisierungen wie „kategoriales Oberkonzept“ unterstützt. Über die Begriffe „Basiskonzept“ und „Primärkonzept“ kann ein Modell auch als Mechanismus zur Charakterisierung von Konzepten gedeutet werden. Es wurde gezeigt, wie diese Charakterisierungen mit Hilfe von beschreibungslogischen Repräsentationen formal definiert werden können.

Es wurde darauf hingewiesen, daß ein beschreibungslogisches Konzept wie `(at-least 1 r)` zwar die Existenz eines Rollenfüllers als konsistent deklariert, nicht aber dessen tatsächliche Bestimmung auch forciert.[87] Zur Beschreibung dieses Phänomens wurde der Begriff der Vollständigkeit bzw. Unvollständigkeit eines Objekts bzgl. einer Rolle eingeführt. Genau dieses ist der Dreh- und Angelpunkt für die Definition der Charakterisierung eines Konzepts als Basiskonzept. Obwohl beispielsweise schon bei dem Konzept `spatial-object` die Existenz eines Rollenfüllers für `has-projective-form` als konsistent deklariert wird, ist erst bei einem Basiskonzept wie `floor-section-model:cabin` durch die Verwendung einer entsprechenden K-Deklaration auch dessen Berechnung sichergestellt. Ein Objekt, das von einem Basiskonzept subsumiert wird, ist vollständig bezüglich der `Rolle has-projective-form`. Eine Unvollständigkeit führt in ähnlicher Weise wie eine Inkonsistenz zur Laufzeit zur Durchführung einer anwendungsspezifischen Ausnahmebehandlung.

Ich habe auf die Schwierigkeiten hingewiesen, die entstehen, wenn man auf der logischen Ebene das In-Beziehung-Setzen von Objekten über Relationen generell vermeiden möchte, indem man dieses als Inkonsistenz deklariert (umständliche Kardinalitätsrestriktionen). Mit der Einführung der K-Operatoren wird eine adäquatere Modellierungsform ermöglicht.

Zu beachten ist, daß schon zur Entwicklungszeit der Anwendung, also bevor die meisten Objekte als konkrete Individuen bekannt sind, sichergestellt werden muß, daß die zur Visualisierung benötigten Informationen zur Laufzeit auch vorliegen. Anders ausgedrückt, wenn nicht vorher hergeleitet werden kann, daß die Rollenfüller z.B. der Relation `spatially-encloses` von einem Basiskonzept subsumiert werden, so wird angenommen, daß sie nicht dargestellt werden *sollen*. Im Modell `floor-section-model` wird z.B. davon ausgegangen, daß eine Kabine keine Objekte räumlich enthält. Wenn zur Laufzeit die entsprechenden Rollenfüller dennoch vorliegen, wird dieses als „Zufall“ betrachtet und nicht berücksichtigt. Dieses wurde auf der Wissensebene als „bewußtes Ignorieren“ gedeutet. Ein Modell wirkt also genau dann als „Informationsfilter“, wenn zur Entwicklungszeit die Konzeptbeschreibungen nicht stark genug eingeschränkt sind.

87. Dieses gilt zumindest nicht für jedes Beweisverfahren, siehe auch [76]. Zu beachten ist auch, daß bei einer automatischen Generierung von Individuen die Zusatzannahme eingeht, daß neu generierte Individuen nicht identisch mit vorher definierten sind. Mit anderen Worten: Es muß möglich sein, Individuen im Nachhinein zu identifizieren.

In diesem Unterkapitel wurde die Konzeption des HAMVIS-Grundmodells aus der Perspektive der Visualisierungsgenerierung diskutiert. Anhand einiger Beispiele wurde die Anbindung domänenspezifischer Modelle an das Grundmodell besprochen. Im nächsten Unterkapitel wird die Aufgabe des Grundmodells bei der Definition einer anwendungsspezifischen Aktionenzerlegung erläutert.

3.2 Aktionenzerlegung

Die zweite Teilphase zur Erstellung einer Anwendung mit den Konzepten von HAMVIS dient zur Strukturierung der Interaktion und zur Festlegung der Benutzeraktionen sowie zur Spezifikation der zur Unterstützung dieser Aktionen notwendigen Berechnungs- und Speicherfunktionen.

Ausgangspunkt für die Visualisierungsgenerierung sind die Aktionen, die der Anwendungsbenutzer zur Laufzeit mit der Oberfläche durchführen will oder muß. In der in dieser Arbeit betrachteten Anwendungsklasse werden direktmanipulative Interaktionsformen mit visuellen Darstellungen benötigt (siehe z.B. die Abbildungen 1 und 14). Zur Realisierung dieser Interaktionstechniken für eine bestimmte Anwendungsklasse muß eine Menge von Standard-Interaktionsbausteinen bereitgestellt werden. Es wird klar, daß sehr schnell eine große und unübersichtliche Bibliothek entstehen kann. Anstatt aus dieser Bibliothek einen Baustein für einen gewissen Zweck von Hand herauszusuchen, ist es vorteilhafter, den Zweck zu beschreiben und sich die zur Realisierung notwendigen Bausteine automatisch zusammenstellen zu lassen. Dieses ist eine wesentliche Aufgabe der Aktionenmodellierungswerkzeuge von HAMVIS.

Den Zweck, den ein Interaktionsbaustein erfüllen soll, kann man über eine Beschreibung der Aktion des Anwendungsbenutzers spezifizieren, die der Baustein unterstützen soll. Damit die Konsequenzen der Auswahl eines speziellen Interaktionsbausteins für die Gestaltung der Darstellungen repräsentierbar sind, sollen Aktionen nicht auf der Ebene der Interaktionsgesten beschrieben werden (Mausknopf drücken, Maus bewegen, Mausknopf loslassen usw.), sondern auf der Ebene der Handhabung von Anwendungsobjekten.

3.2.1 Repräsentationen von Benutzeraktionen auf der Wissensebene

Wissen über die Handhabung von Domänenobjekten, also z.B. Selektion oder Manipulation, kann in Form von Kasusstrukturen repräsentiert werden. Für die in der XKL-Anwendungsklasse relevanten Aktionen wie z.B. Verschiebung von räumlichen Objekten, Lokalisierung von unplazierten räumlichen Objekten, Beschreibung von räumlichen Bereichen usw. lassen sich entsprechende Kasusrollen wie „manipuliertes Objekt“, „referenziertes Objekt“ definieren. Die Gesamtstruktur der XKL-Anwendung wurde schematisch in Abbildung 6 skizziert. Im Kontext von HAMVIS ist es wichtig, daß sich die Effekte von Benutzeraktionen durch aufgerufene Speicherfunktionen manifestieren. Bei einer Lokalisierung eines neuen Kabinenobjekts wird anschließend die neue Position bzw. Positionseinschränkung durch eine Speicherfunktion am Objekt vermerkt und ggf. an ein CAD-System übermittelt. Nicht die Aktionen selbst verändern die Welt, sondern die „nachgeschalteten“ Speicherfunktionen. Im Kontext von HAMVIS können also Benutzeraktionen als Funktionen aufgefaßt werden, die bestimmte Werte liefern, aber nicht die Welt destruktiv verändern. Die möglichen Aktionen des Benutzers der Anwendung werden durch das Systementwicklungsteam zur Entwicklungszeit spezifiziert, wobei die notwendigen automatischen Berechnungs- und Speicherfunktionen gleich mit angegeben werden.

Betrachten wir hierzu wieder das XKL-Beispiel aus der Einleitung: Die Aktion zur Erzeugung eines neuen Kabinenobjektes für die Festlegung der Inneneinrichtung des Flugzeugs kann zusammengesetzt werden aus einer Auswahlhandlung und einer Lokalisierungshandlung. Die erste Handlung bestimmt aus einer Menge von noch fehlenden Einrichtungsgegenständen das „nächste“ Objekt, das in die Inneneinrichtung aufgenommen wird, die zweite Teilhandlung legt dann für genau dieses Objekt gleich die Position fest. Weiterhin werden Berechnungsfunktionen benötigt: eine Funktion zur Berechnung der im aktuellen Designzustand jeweils noch fehlenden Einrichtungsobjekte und eine Funktion zur Ermittlung der jeweils für das gewählte Objekt noch möglichen Plazierungsbereiche. Die noch fehlenden Einrichtungsgegenstände werden in Form von sog. Prototypobjekten geliefert, d.h. beim Einfügungen eines Objekts in die Inneneinrichtung wird der gewählte Prototyp kopiert. Nach Ausführung der Lokalisierungshandlung muß die Positionseinschränkung, die sich durch den gewählten Plazierungsbereich ergibt, am lokalisierten Objekt vermerkt werden (Speicherfunktion). Aus Sicht des Endbenutzers der Anwendung kann diese Sequenz von Teilhandlungen auch als *zusammengesetzte Handlung* interpretiert werden. Die Berechnungs- und Speicherfunktionen sind damit transparent. Die zugrundeliegenden, nicht zerlegbaren Benutzeraktionen auf der niedrigeren Ebene nenne ich *elementare Benutzeraktionen*.

In einer Anwendung wie z.B. XKL lassen sich verschiedene zusammengesetzte Handlungen aufzählen (Erzeugen von Objekten, Dimensionieren, Löschen, Verschieben und Ausrichten von Objekten). Sie stellen Handlungsalternativen dar, die jeweils innerhalb eines Durchgangs durch den Interaktionszyklus ausgeführt werden können. Dem Benutzer muß eine graphische Oberfläche präsentiert werden, die es ihm erlaubt, im Interaktionszyklus jeweils eine der möglichen Handlungen auszuwählen (siehe die Diskussion der Interaktionstheorie nach Norman in Kapitel 2.2.1). Der Oberflächenentwickler kann mehrere zusammengesetzte Handlungen zu einer *Aktivität* gruppieren, in der jeweils ein gewisses Teilproblem der Anwendung bearbeitet wird. Anders ausgedrückt: Eine Aktivität kann in verschiedene Handlungsalternativen (zusammengesetzte Handlungen) zerlegt werden (ODER-Zerlegung). Die Menge der Aktivitäten legt dann die Struktur der Gesamtanwendung (*Applikation*) fest. In Kapitel 2.2.3 wurden verschiedene Ansätze vorgestellt, die das Wissen über die Struktur einer Anwendung in ähnlicher Weise in vier Ebenen repräsentieren.

Das Ziel der Visualisierungsgenerierung für eine Aktivität ist es, alle Anforderungen der zusammengesetzten und elementaren Benutzerhandlungen zu erfüllen, so daß eine „benutzerfreundliche“ Interaktionsform innerhalb der Aktivität entsteht. Wichtig ist, daß die möglichen zusammengesetzten Aktionen einer Aktivität echte Alternativen sind, d.h. es soll zur Laufzeit innerhalb einer Aktivität eine nicht-modale Interaktionsform unterstützt werden (siehe die vorausgegangene Diskussion eingangs von Kapitel 2). Statt einer permanenten Reaktion auf eine vom „System“ gegebene Sequenz von Aufforderungen oder Fragen, wird vom Benutzer erwartet, daß er die möglichen Handlungen kennt und selbständig eine Aktion auswählt, die zum Ziel der Aktivität führt. Es liegt das Standard-Interaktionsschema für interaktive Anwendungen vor, das schon von Norman diskutiert wurde (siehe Kapitel 2.2.1, Abbildung 16). Seine „Aktionsfolge“ innerhalb eines Durchgangs durch den Interaktionszyklus entspricht einer zusammengesetzten Handlung in der von HAMVIS verwendeten Terminologie. Innerhalb des Interaktionszyklus einer Aktivität können zur Laufzeit beliebige Folgen von (zusammengesetzten) Benutzerhandlungen auftreten. Eine konkrete Sequenz von zusammengesetzten Benutzerhandlungen liegt also zur Entwicklungszeit nicht vor. Eine Aktionenmodellierung kennzeichnet für jede Aktivität also alle möglichen Handlungssequenzen. Ich habe schon erwähnt, daß anstelle der Begriffs „Benutzerhandlung“ oder auch „Benutzeraktion“ eigentlich immer der umständlichere

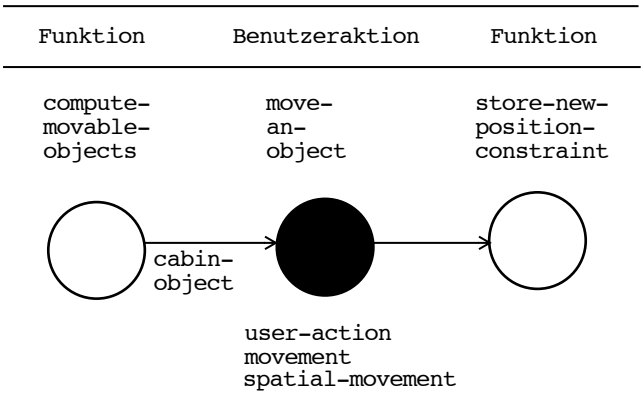
Begriff des Handlungsschemas oder der Begriff der Aktionsmöglichkeit verwendet werden müßte. Eine Zerlegung einer Aktivität in Teilhandlungen (zusammengesetzte Handlungen) bedeutet also, daß die Teilhandlungen Handlungsalternativen sind, zwischen denen der Benutzer innerhalb des Interaktionszyklus frei wählen kann. Handlungsalternativen werden innerhalb dieser „Schleife“ nicht direkt „angekündigt“ (man stelle sich einem Texteditor vor, bei dem alle möglichen Editieroperationen dem Benutzer textuell angezeigt werden). Wenn der Benutzer aber z.B. den Mauszeiger über ein maussensitives graphisches Objekt positioniert, können je nach UIMS die möglichen *objektspezifischen* Handlungen ggf. in der Mausdokumentationszeile angezeigt werden. Falls eine Handlungsalternative nicht möglich ist, so sind die entsprechenden Mausgesten nicht aktiviert.

Im Zusammenhang mit der Auswahl der für die Durchführung einer Handlung zu präsentierenden Informationen ist der oben verwendete Begriff „benutzerfreundlich“ allerdings i.a. schwer faßbar. Schon bei der Diskussion der Ansätze der kognitiven Psychologie (siehe Kapitel 2.3.4 und Kapitel 2.3.5) und der IMMP-Systeme in Kapitel 2.4 wurden Schwierigkeiten bei der Beschreibung der Qualität einer Präsentation aufgezeigt. Auch auf der Ebene der Aktionenmodellierung sind bei den zusammengesetzten Handlungen Bedürfnisse des Benutzers zu berücksichtigen. Dieses möchte ich am Beispiel der zusammengesetzten Handlung für die Erzeugung eines neuen Kabinenobjekts diskutieren. Nehmen wir an, statt durch eine Auswahlhandlung würde das jeweils nächste zu lokalisierende Prototypobjekt durch eine Berechnungsfunktion bestimmt. Wenn nicht der Benutzer dieses auswählt, sondern eine automatische Berechnungsfunktion, dann muß der Benutzer über den Typ des zu lokalisierenden Objekts zumindest informiert werden. Es ist nicht adäquat, plötzlich im Kabinenrumpf irgendwelche Plazierungsbereiche zu zeigen und zu warten, bis der Benutzer eines davon angeklickt hat. Durch die Auswahlaktion wird ein aktuelles Objekt im Diskurs „fokussiert“ (in Sinne eines attentionalen Diskurszustands). Das Objekt wird dann in der Lokalisierungsaktion weiter „bearbeitet“. Die Lokalisierungsaktion selbst führt die Diskursfokussierung nicht durch, sondern benötigt eine vorhergehende Aktion, mit der ein Objekt im Diskurs fokussiert wird. Damit wird auch deutlich, warum sich eine Lokalisierung dieser Art prinzipiell von einer Auswahl unterscheidet. Es wird deutlich, daß sich Aspekte der Diskursmodellierung auch in der Aktionendekomposition niederschlagen müssen. Ich komme in Abschnitt 3.2.9 wieder auf Aspekte der Diskurs- und Dialogmodellierung zurück. Zunächst möchte ich das Zusammenspiel von generischem Aktionswissen, das von HAMVIS bereitgestellt wird, und domänenspezifischem Aktionswissen beleuchten, welches vom Systementwicklungsteam für die spezielle Anwendung modelliert werden muß.

Zur Entwicklungszeit kann die Modellierung von Wissen über mögliche Benutzerhandlungen nur in konzeptueller Form erfolgen. Eine Aktionsmöglichkeit ist – wie oben geschildert – durch entsprechende Kasusrollen gekennzeichnet. Die möglichen Füller dieser Kasusrollen werden durch Konzepte des HAMVIS-Grundmodells beschrieben. Weiterhin gehört zum Wissen über Benutzerhandlungen das Wissen über die zur Realisierung der Handlung zu präsentierenden Objekte und Interaktionswerkzeuge. Mit der Handlung ist auch Wissen über Anforderungen an die Darstellung von Objekten verbunden.

Nehmen wir als Beispiel eine Verschiebung von räumlichen Objekten. In diesem Beispiel sind die Füller der Kasusrolle „Objekt“ bzw. „manipuliertes Objekt“ zu präsentieren. Zu beachten ist, daß diese zur Entwicklungszeit der Anwendung nicht bekannt sind. Es kann nur eine Funktion definiert werden, die die verschiebbaren Objekte zur Laufzeit liefern wird (z.B. eine Funktion `compute-movable-objects` die eine Menge von Kabinenobjekten liefert, siehe Abbildung). Auf jedes ein-

zelne Kabinenobjekt kann dann die angesprochene Verschiebehandlung angewendet werden. Anders ausgedrückt: Jedes dieser Objekte könnte zur Laufzeit der Anwendung Füller der Objekt-Kasusrolle der Verschiebehandlung werden. Die Eigenschaften der zur Laufzeit manipulierbaren Objekte sind aber durch die Konzeptbeschreibung `cabin-object` schon zur Entwicklungszeit bekannt.



Schematische Darstellung der Zerlegung der Aktion `move-cabin-object` aus Abbildung 6 (siehe Text).

Durch das Systementwicklungsteam werden Berechnungs- und Speicherfunktionen über die Konzepte (Typen) der Parameter und Werte spezifiziert. Allerdings sind – wie oben ausführlich geschildert – Domänenkonzepte über das Vererbungsprinzip durch Konzepte des HAMVIS-Grundmodells definiert. Die für XKL betrachteten Kabinenobjekte sind z.B. räumliche Objekte, ein Konzept des HAMVIS-Grundmodells.

Aktionenkonzepte, wie z.B. eine Verschiebung, werden ebenfalls in einer Vererbungshierarchie repräsentiert. Eine spezielle Verschiebung `spatial-movement` ist z.B. für die Manipulation von räumlichen Objekten vorgesehen. Nehmen wir an, daß in Abbildung das Konzept der vorgesehenen Handlung `move-an-object` noch nicht vollständig spezifiziert sei, z.B. sei nur bekannt, daß es sich um eine Verschiebung handelt (`movement`), aber noch offen, daß es eine Verschiebung von räumlichen Objekten sein soll (`spatial-movement`). Wird nun durch die Anbindung der Berechnungsfunktion für die zu verschiebenden Objekte bekannt, daß es sich bei den verschiebbaren Objekten in jedem Fall um Kabinenobjekte handelt, dann läßt sich schließen, daß als mögliche Verschiebehandlung nur eine räumliche Verschiebung in Frage kommen kann, da Kabinenobjekte ja laut Modell räumliche Objekte sind.

Auch in die andere Richtung kann die Schlußfolgerung gehen. Wenn als Handlung eine spezielle räumliche Verschiebehandlung vorgesehen wird und noch nicht spezifiziert ist, von welchem Typ der Wert der Berechnungsfunktion sein wird, die die verschiebbaren Objekte liefert, dann kann der Typ des Wertes zumindest auf räumliche Objekte eingeschränkt werden. Auch bezüglich der Komposition von Handlungen lassen sich Schlußfolgerungen erkennen. Bei einer zusammengesetzten Handlung kann über die Konzepte der Teilhandlungen geschlossen werden, ob sich ggf. eine Realisierung mittels einer speziellen Interaktionstechnik wie z.B. eine Ziehen-und-Fallenlassen-Geste anbietet.

In dem Beispiel aus Abbildung geht HAMVIS davon aus, daß der Entwickler für jedes Objekt die gleiche Handlung vorsieht. Falls jedoch z.B. zur Laufzeit unterschiedliche elementare Handlungen auf Sitze und Küchen (beides seien Unterkonzepte von Kabinenobjekt) angewendet werden sollen, so muß der Entwickler zwei verschiedene zusammengesetzte Benutzeraktionen als Handlungsalternati-

ven einer Aktivität definieren und entsprechende Berechnungsfunktionen angeben, die jeweils nur Sitze und Küchen liefern. Für Sitze wäre beispielsweise eine speziellere Verschiebehandlung denkbar, mit der eine ganze Sitzreihe verschoben werden kann und sich auch der Abstand der anderen Sitzreihen entsprechend ändert usw. Es wäre weiterhin möglich, auch zur Laufzeit noch Inferenztechniken einzusetzen, um z.B. spezielle Handlungen für spezielle geometrische Situationen zu bestimmen. Zu beachten ist jedoch, daß zur Entwicklungszeit erst einmal ein Rahmen festgelegt werden muß, so daß die grundlegende Gestaltung der Oberfläche festgelegt werden kann. Die Gestaltung und Strukturierung der Oberfläche soll sich nicht ständig ändern, sondern für alle Eventualitäten geeignet sein. Die Beschreibung von elementaren Aktionen muß also zur Entwicklungszeit mit Konzepten (aus dem HAMVIS-Grundmodell) erfolgen, die ausreichend speziell sind, so daß der Inhalt von notwendigen Visualisierungen bestimmt und die Grundstruktur der Oberfläche festgelegt werden kann (HAMVIS zeigt dem Entwickler an, wenn Handlungen ausreichend speziell beschrieben sind).

In den nachfolgenden Abschnitt gehe ich davon aus, daß notwendige Spezialisierungen zur Entwicklungszeit erfolgen. Auf eine Spezialisierung von Aktionen zur Laufzeit mit entsprechender Fallunterscheidung komme ich später zurück (siehe Kapitel 4.2).

Während bei der Modellierung von Domänenkonzepten durch das Systementwicklungsteam Konzepte zur Spezialisierung des HAMVIS-Grundmodells definiert werden, wird bei der Aktionenmodellierung nur noch die „Komposition“ festgelegt, d.h. es werden keine neuen Aktionenkonzepte eingeführt, sondern bestehende in einer Aktionenzerlegung kombiniert. Die möglichen Inferenzen beziehen sich auf Objekte, die zwar noch nicht erzeugt, deren konzeptuelle Beschreibungen aber schon zur Entwicklungszeit der Anwendung festgelegt werden können.

Durch die Kopplung von konzeptuell beschriebenen Aktionenschemata mit Berechnungs- und Speicherfunktionen können also verschiedene Schlußfolgerungen durchgeführt werden. Diese und ähnliche Schlußfolgerungen können über Beschreibungslogiken formalisiert werden. In dem nachfolgenden Abschnitt wird geschildert, wie die von HAMVIS bereitgestellten Repräsentationsformen zur Handlungsmodellierung aussehen und wie die hier skizzierten Infererenzschemata auf der logischen Ebene formalisiert werden. Zu beachten ist allerdings wieder, daß eine Aktionenzerlegung von den Mitgliedern des Systementwicklungsteams durchgeführt wird. Zum Aufstellen der Modelle und zum Nachvollziehen der automatischen Ableitungen wird eine interaktive Oberfläche benötigt. Es werden in den nachfolgenden Abschnitten also wieder formale Modellierungsformen und das Zusammenspiel mit interaktiven Darstellungsformen gezeigt. Die nachfolgend gezeigte Ausprägung dieser Oberfläche ist ein möglicher Gestaltungsvorschlag, anhand dessen hier die auf der formalen Ebene stattfindenden Schlußfolgerungen aufgezeigt und diskutiert werden.

3.2.2 Erstellung einer Aktionenzerlegung mit einer interaktiven Oberfläche

Die Modellierung der Handlungsstruktur erfolgt in vier Ebenen: Applikation, Aktivitäten, zusammengesetzte Handlungen und elementare Benutzerhandlungen bzw. Berechnungs- oder Speicherfunktionen. Obwohl zusammengesetzte Handlungen innerhalb einer Aktivität Alternativen sind, die in einer Schleife ausgeführt werden, möchte ich auch hier von einer Zerlegung sprechen (ODER-Zerlegung). Es ergibt sich also für eine Anwendung insgesamt ein Zerlegungsbaum der Höhe vier. Nun hat es sich nicht als vorteilhaft herausgestellt, diese Zerlegungsstruktur auch graphisch als Baum zu repräsentieren. Aufgrund der Erfahrungen mit dem HAMVIS-Prototypen wurde die Oberflächenstruktur aus

Abbildung 89 entwickelt. Große Teile der interaktiven Oberfläche wurden von Brüning im Rahmen ihrer Diplomarbeit mit CLIM 2.0 (Motif) unter Allegro Common Lisp 4.2 erstellt [35].

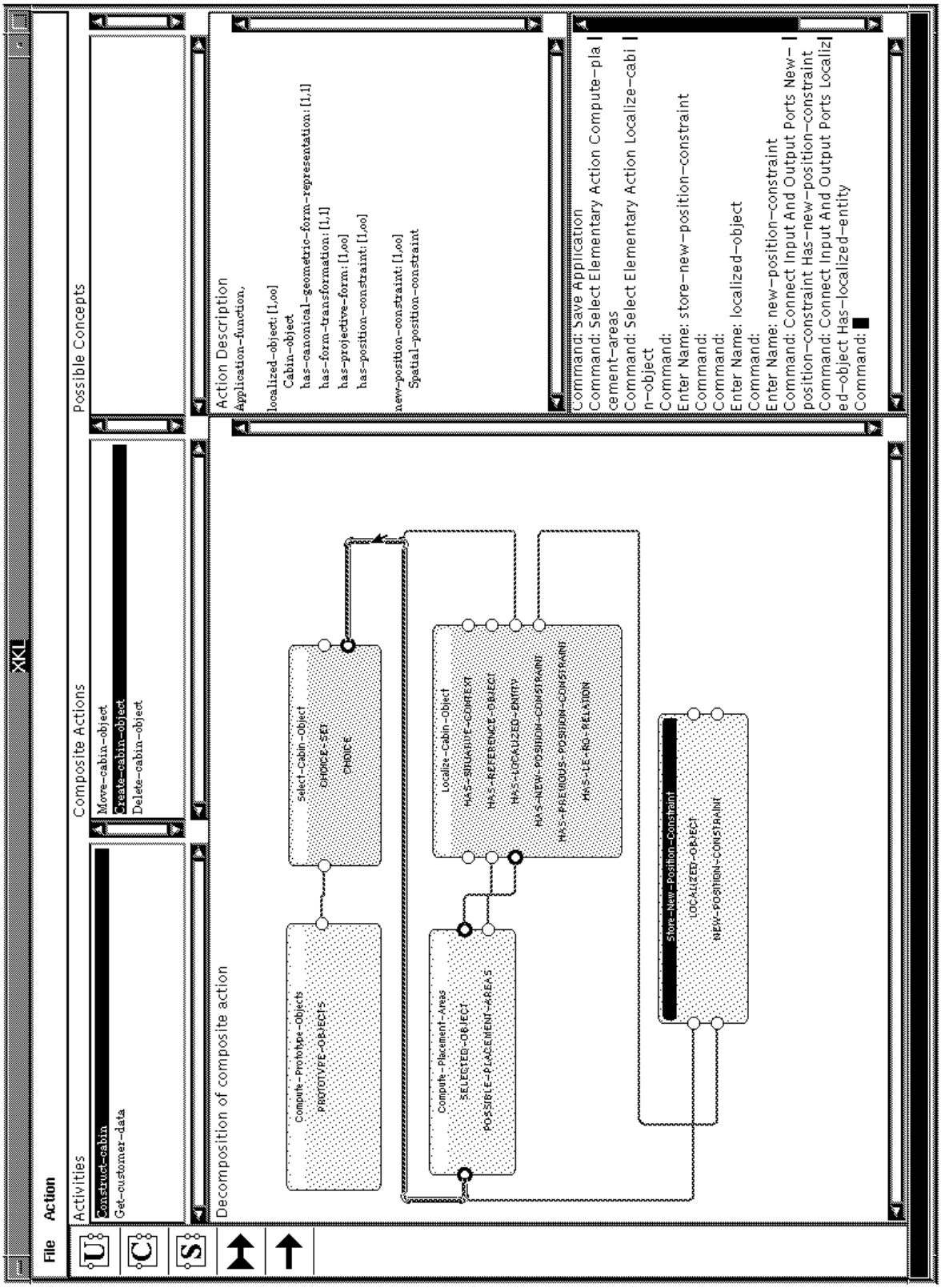


Abbildung 89. Die HAMVIS-Oberfläche für das Systemenwicklungsteam zur Erstellung einer Aktionenzerlegung.

Für die Aktionenzerlegung einer Applikation wird ein separates Interaktionsfenster bereitgestellt. Im Beispiel aus Abbildung 89 ist das Fenster für die XKL-Anwendung gezeigt (siehe die Titelzeile des Fensters). Die Oberfläche setzt sich zusammen aus drei Tabellen (oberer Bereich), einem Hauptfenster und zwei kleineren Teilfenstern (panes). Das obere der beiden kleinen quadratischen Teilfenster auf der rechten Seite dient zur Beschreibung von Aktionen, das untere ist ein Ein- und Ausgabefenster. Auf der linken Seite ist eine Palette von Objekten und Werkzeugen untergebracht, deren Bedeutung in den nachfolgenden Abschnitten noch genauer erläutert wird. Im unteren schmalen, invers dargestellten Fenster ist eine Zeile zur Dokumentation der Mausfunktionen untergebracht.

Die linke Tabelle im oberen Teil des Gesamtfensters zeigt die Aktivitäten, aus denen sich die Applikation XKL zusammensetzen soll. Für die markierte Aktivität werden in der mittleren Tabelle die Handlungsalternativen (zusammengesetzten Aktionen) gezeigt. Eine Markierung wird durch eine Invertierung des Namens angedeutet (schwarzes Rechteck). Die Zerlegung der markierten zusammengesetzten Aktion in Berechnungs- und Speicherfunktionen sowie elementare Benutzeraktionen wird dann im Hauptinteraktionsfenster definiert. Aktivitäten und zusammengesetzte Aktionen können interaktiv zu der Zerlegungshierarchie hinzugefügt bzw. gelöscht werden.

Die rechte obere Tabelle zeigt die noch möglichen Spezialisierungen einer markierten elementaren Benutzeraktion.

3.2.3 Formale Modellierung von Aktionen im HAMVIS-Grundmodell

Neben der oberflächennahen Präsentation muß die Zerlegungsstruktur noch formal repräsentiert werden. Hierzu enthält das HAMVIS-Grundmodell einen Kern von Konzept- und Rollenbeschreibungen für die Modellierung von Aktionen und ihrer Zerlegung. Dieser Kern wird nicht aus dem HAMVIS-Grundmodell exportiert, also nicht in Domänenmodellen erweitert. Für die Beschreibung der Handlungsstruktur einer Anwendung ist die Definition der Komposition der vorhandenen Aktionenkonzepte ausreichend.

Eine Teil-von-Zerlegung von Aktionen wird über die Rolle `has-substep` vorgenommen (vgl. Kapitel 3.1.2). Für die verschiedenen Ebenen der Aktionenmodellierung wurden jeweils eigene Aktionentypen eingeführt, wobei das Konzept der Füller der Rolle `has-substep` entsprechend eingeschränkt wird. Elementare Benutzeraktionen (`user-action`) sowie Berechnungs- und Speicherfunktionen (`application-function`) werden durch `atomic-action` zusammengefaßt mit einer entsprechenden Einschränkung für die Zerlegungsrelation versehen.

```
(in-model hamvis-upper-model)

(define-primitive-role has-substep has-decomposition)

(define-primitive-concept decomposition-component component)

(define-disjoint-primitive-concept atomic-action (action-hierarchy-level)
  (and decomposition-component
       (atmost 0 has-substep)))

(define-disjoint-primitive-concept composite-action (action-hierarchy-level)
    (and decomposition-component
         (atleast 1 has-substep)
         (all has-substep atomic-action)))

(define-disjoint-primitive-concept activity (action-hierarchy-level)
  (and decomposition-component
       (atleast 1 has-substep)
       (all has-substep
            composite-action)))

(define-disjoint-primitive-concept application (action-hierarchy-level)
  (and decomposition-component
       (atleast 1 has-substep)
       (all has-substep activity)))

(define-disjoint-primitive-concept user-action (action-agent)
  atomic-action)

(define-primitive-concept action-with-focused-input user-action)
(define-primitive-concept action-with-focussing-output user-action)

(define-disjoint-primitive-concept application-function (action-agent)
  atomic-action)
```

Abbildung 90. Kern des HAMVIS-Grundmodells zur Aktionenmodellierung.

Wir werden später einige speziellere Konzepte für zusammengesetzte Aktionen (`composite actions`, siehe Abbildung 90) betrachten, die mit notwendigen und hinreichenden Bedingungen definiert werden und über beschreibungslogische Schlußfolgerungen die Umsetzung von zusammengesetzten Aktionen durch Interaktionsbausteine und Interaktionsgesten ermöglichen (siehe Kapitel 3.3.11). Die Schlußfolgerungen hängen von den Konzepten der Unterhandlungen in der Aktionenzerlegung ab. Ich möchte daher zunächst detaillierter auf die Zerlegung von zusammengesetzten Aktionen zu sprechen kommen. Zur Illustration der Formalisierung von Schlußfolgerungen über Beziehungen zwischen Berechnungsfunktionen und elementaren Benutzeraktionen betrachten wir wieder die XKL-Anwendung. Die in Abbildung 6 gezeigte Struktur der XKL-Anwendung wird durch eine Menge von ABox-Individuen modelliert. Je nach Ebene werden die Konzepte aus Abbildung 90 zugewiesen. Man beachte, daß die ABox-Individuen *zur Entwicklungszeit Handlungsschemata* auf den verschiedenen Ebenen modellieren und nicht zur Laufzeit als konkrete Handlungen bzw. Handlungsfolgen erzeugt werden.

3.2.4 Die Handlungsstruktur von XKL: ein vereinfachtes Beispiel

Das Systementwicklungsteam definiert die Grundstruktur mit Hilfe der interaktiven Oberfläche. Die notwendigen ABox-Aussagen werden intern generiert. Es wird beispielsweise festgelegt, daß die XKL-Anwendung aus zwei Aktivitäten besteht. Die eine Aktivität dient zur Erfassung der Kundendaten (`get-customer-data`), die andere soll die Einrichtung des Innenraums der Flugzeugkabine (`construct-cabin`) ermöglichen (auf die Reihenfolge bzw. die Aktivierungsbedingungen der Aktivitäten innerhalb der Applikation gehe ich in Abschnitt 3.3.12 ein.). In Abbildung 91 ist der relevante Teil des Interaktionsfenster mit einer „Lupe“ hervorgehoben.

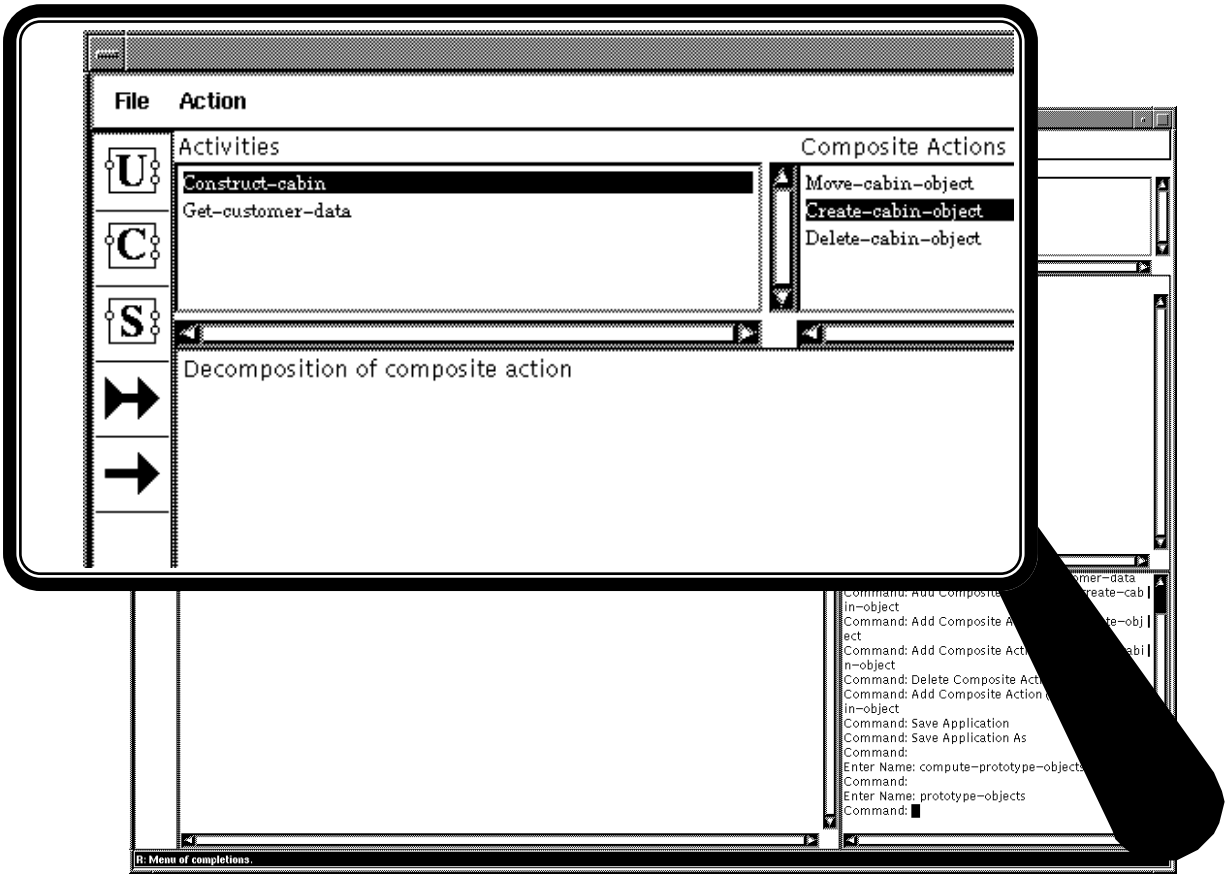


Abbildung 91. Beispielstrukturierung der XKL-Anwendung.

Da die Aktivität `construct-cabin` markiert ist, wird in der mittleren Tabelle deren „Zerlegung“ gezeigt. Die Zerlegung enthält die Handlungsalternativen innerhalb der Aktivität. Wir betrachten hier drei zusammengesetzte Handlungen `move-cabin-object`, `create-cabin-object` und `delete-cabin-object`. Die Namen sind jedoch nur für die Oberfläche relevant. Das Wissen über diese Aktionen wird auf der Logikebene durch entsprechende Assertionen modelliert. Für jede Aktion der verschiedenen Hierarchiestufen wird ein Individuum erzeugt. Den jeweiligen Individuen werden die für die Hierarchiestufe vorgesehenen Konzepte zugeordnet (siehe Abbildung 91). Weiterhin wird über die Relation `has-substep` eine entsprechende Zerlegungsstruktur aufgebaut.

Vom epistemologischen Standpunkt her dienen die hier verwendeten Aktionenindividuen als „Stellvertreter“ für die zur Laufzeit zu erzeugenden Handlungsobjekte. Sie ermöglichen schon zur Entwicklungszeit der Anwendung Schlußfolgerungsprozesse über die zu unterstützenden Vorgänge. Durch die interaktive Oberfläche tritt diese „interne“ Sicht jedoch in den Hintergrund. Das Systementwicklungsteam spezifiziert die möglichen Handlungssequenzen des Endbenutzers der XKL-Anwendung. Man beachte, daß die drei zusammengesetzten Aktionen der Aktivität `construct-cabin` als Alternativen aufzufassen sind, die solange angewendet werden, bis ein bestimmtes Ziel erreicht ist, oder die Aktivität abgebrochen wird (vgl. das Diagramm von Norman in Abbildung 16).

3.2.5 Spezifikation von Berechnungs- und Speicherfunktionen

Auf der untersten Ebene muß die Zerlegung von zusammengesetzten Aktionen in atomare Aktionen (`atomic-actions`, siehe Abbildung 90) erfolgen. Wir betrachten zunächst die Modellierung von `create-cabin-object` (in Abbildung 91 als „aktuelle“ zusammengesetzte Handlung markiert).

Zur Einfügung von atomaren Aktionen dient die Auswahlpalette auf der linken Seite des HAMVIS-Aktionenmodellierungsfensters. Die Piktogramme können mit der Maus in das Hauptinteraktionsfenster in der Mitte gezogen werden. „U“ steht Benutzeraktion (user action), „C“ kennzeichnet eine Berechnungsfunktion (computation function). „S“ steht für eine spezielle Berechnungsfunktion zur Erkennung von räumlichen Situationen. In Abbildung 92 wurde eine Berechnungsfunktion mit Namen `compute-prototype-objects` in das Fenster gezogen (siehe die linke Lupe).[88] Der Name ist frei wählbar und dient zur Dokumentation und Kommentierung.

Berechnungsfunktionen sind durch die Parametertypen und die Typen des Wertes bzw. der Werte, die sie liefern, gekennzeichnet. In der graphischen Darstellung von Funktionen werden die Eingangsparameter durch kleine Anschlüsse auf der linken Seite, die Werte durch Anschlüsse auf der rechten Seite repräsentiert. Anschlüsse sind jeweils benannt. Das hat zur Folge, daß auch der Wert einer Funktion einen Namen bekommt, ich spreche dann auch von einem *Ausgangsparameter.*[89]

In der Werkzeugpalette auf der linken Seite des Interaktionsfenster sind unter den Piktogrammen für Aktionentypen auch Piktogramme zur Beschreibung von Ein- und Ausgangparametern von Berechnungsfunktionen plaziert. Ein Parameter bzw. Wert einer Funktion wird dann durch das Ziehen eines der beiden Piktogrammsymbole auf das graphische Objekt der Funktion zu dieser hinzugefügt.[90]

88. Die Größe des beschreibenden graphischen Objekts wird bei der Erzeugung der Funktion durch ein Gummibandrechteck bestimmt.

89. Vgl. die relationale Modellierung von Funktionen in PROLOG.

90. Durch die Wiedergabe im Schwarz-Weiß-Druck gehen die Farbinformationen verloren. Berechnungsfunktionen sind mit einem blaßgelben Rechteck dargestellt. Für Benutzeraktionen wird ein hellblaues Rechteck verwendet. In den Bildschirmabzügen wird ein Punktmuster mit einem entsprechenden Grauwert gezeigt.

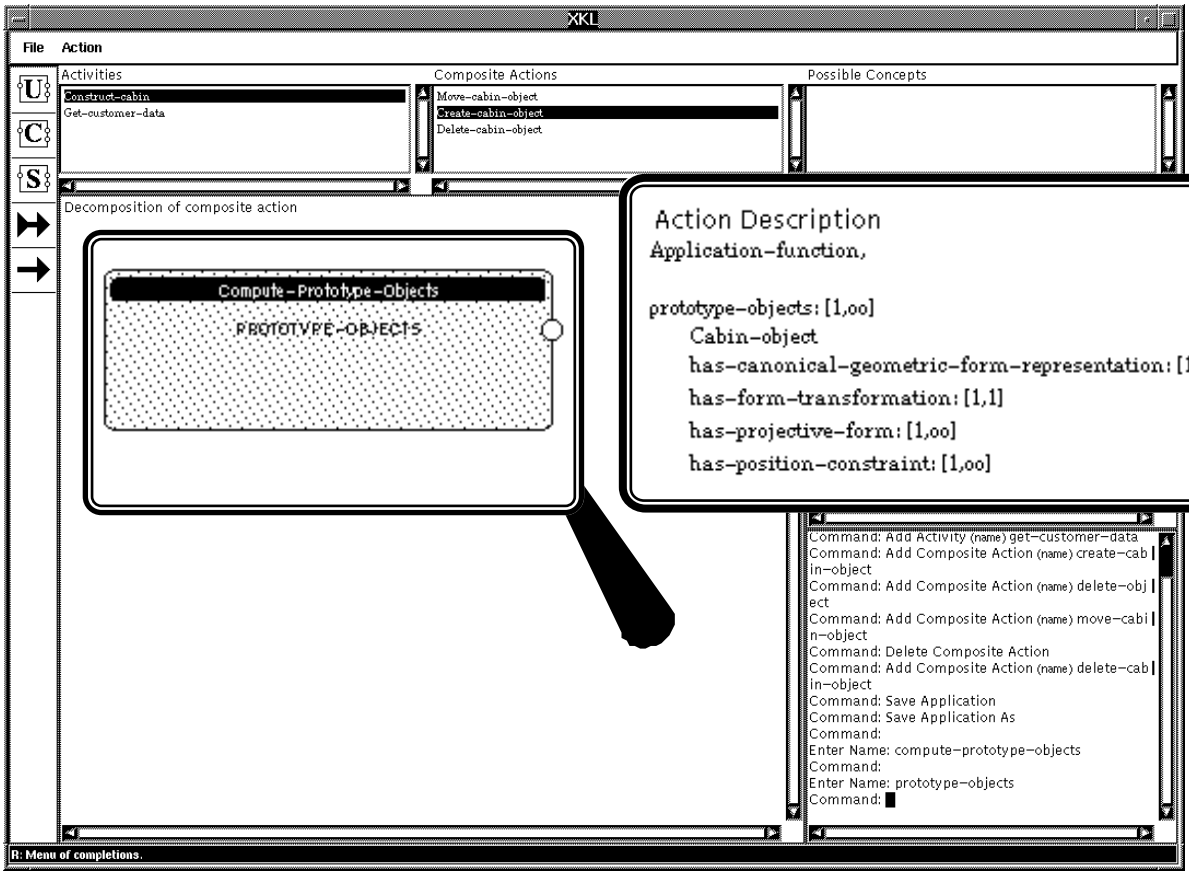


Abbildung 92. Einfügung einer Berechnungsfunktion (linke Lupe) mit Darstellung der Spezifikation der zur Laufzeit der Anwendung zu liefernden Werte (rechte Lupe).

Die Beschreibung der Typspezifikation der markierten Aktion erscheint auf der rechten Seite im oberen der beiden quadratischen Fenster. In Abbildung 92 wird wiederum die Lupe zur Vergrößerung des Inhalts verwendet. Durch das Systementwicklungsteam können hier Konzepte aus dem Basismodell der Anwendung (hier `xkl-model`) angegeben werden. Ich gehe hier nicht auf die realisierte menüorientierte Interaktionsform ein und verweise auf [35]. Es werden die aus der Konzeptangabe abgeleiteten Rolleneinschränkungen angezeigt. Für den Ausgabeparameter `prototype-objects` gilt z.B., daß bei der Evaluierung der Funktion `compute-prototype-objects` innerhalb des Interaktionszyklus jeweils mindestens ein Objekt berechnet wird (Kardinalitätsangaben sind in der Darstellung in eckigen Klammern angegeben).[91] In Abbildung 92 wurde auch das Konzept dieser Objekte angegeben (`cabin-object`), und damit ergeben sich über die Vererbung von `spatial-object` die entsprechenden Rolleneinschränkungen, die unter dem Konzeptnamen ausgegeben worden sind. Weitere Einschränkungen können durch Anklicken der Ausgaben vorgegeben werden.

91. `oo` steht für „unendlich“

3.2.6 Spezifikation von elementaren Benutzeraktionen

Aktionsmöglichkeiten des Benutzers der zu erstellenden XKL-Anwendung können in ähnlicher Weise deklariert werden. In Abbildung 93 wurde schon das Piktogramm „U“ in das Hauptinteraktionsfenster gezogen und eine Benutzeraktion `select-cabin-object` erzeugt. Der Name ist frei wählbar und dient nur zur Identifikation der Aktion.

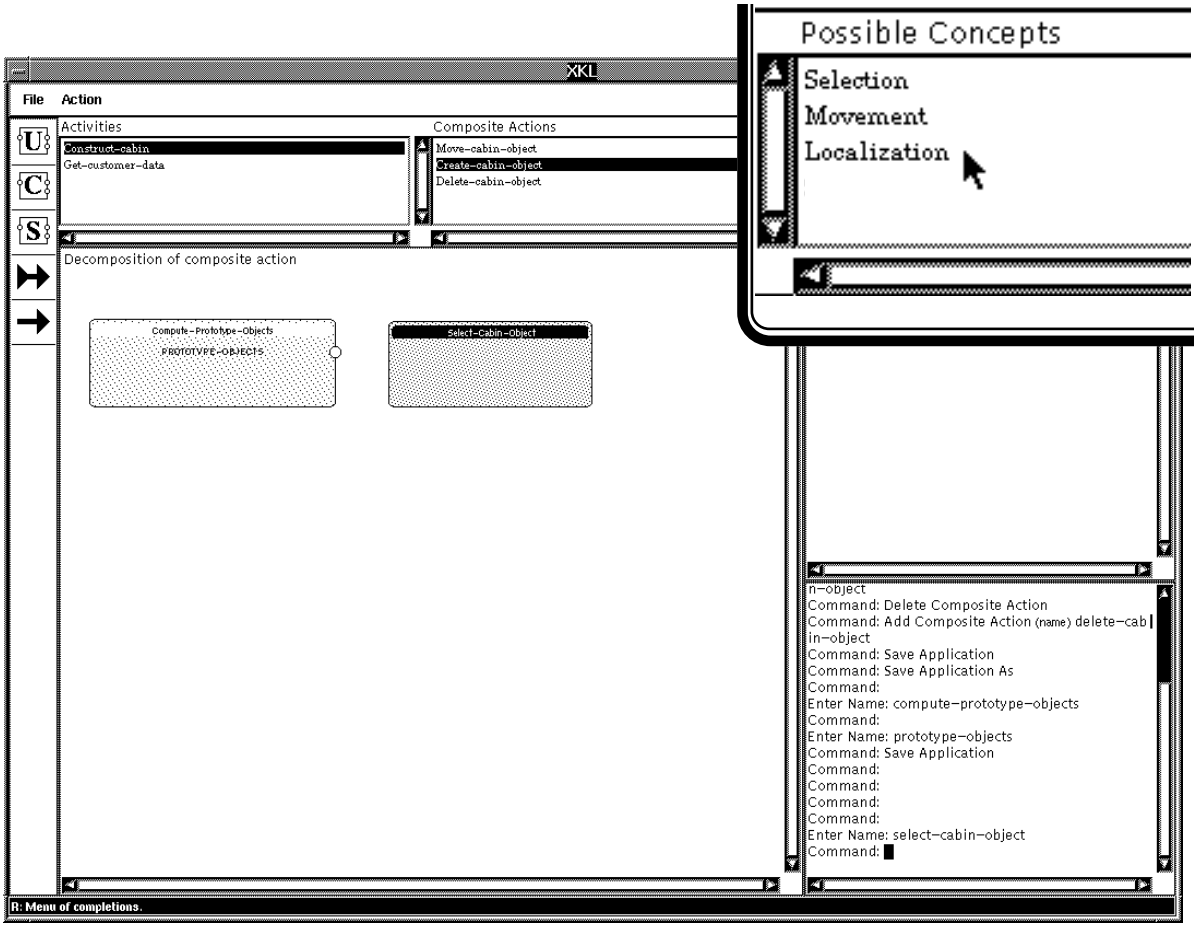


Abbildung 93. Einfügung einer Benutzeraktion mit Darstellung der möglichen Konzepte.

Benutzeraktionen müssen näher beschrieben werden. Hierzu stellt HAMVIS ein Menge von generischen Aktionenkonzepten für eine Anwendungsklasse bereit. In Abbildung 93 werden mit der Lupe die möglichen Konzepte im Fenster `Possible Concepts` gezeigt. Nehmen wir an, der HAMVIS-Benutzer wählt `Selection` aus. Durch die Wahl von `Selection` ist die grobe Charakterisierung der Handlung durch Kasusrollen vorgegeben. HAMVIS leitet aus seinen Aktionenmodellen ab, daß eine Handlung vom Konzept `Selection` durch zwei Kasusrollen `choice-set` und `choice` gekennzeichnet ist (siehe Abbildung 94).[92] In der Tabelle `Possible Concepts` stehen jetzt speziellere Konzepte zur Auswahl bereit.

92. Parameter für Benutzeraktionen werden nicht durch den Entwickler vorgegeben.

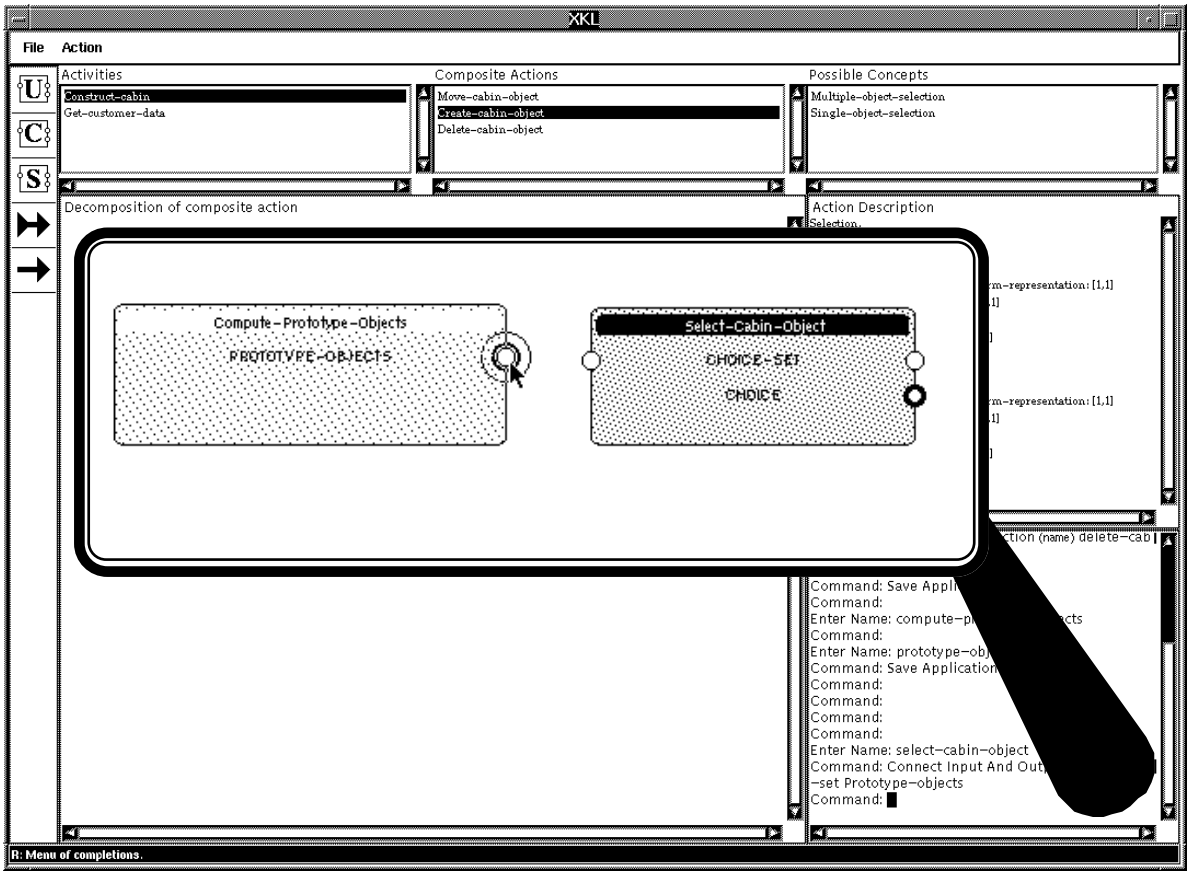


Abbildung 94. Kopplung von Berechnungsfunktionen und Benutzeraktionen.

Die Menge der zur Auswahl stehenden Objekte der Handlung `select-cabin-object` wird zur Laufzeit der Anwendung bestimmt. Die Berechnung soll durch die zuvor definierte Funktion `compute-prototype-objects` erfolgen. Der Zusammenhang von Berechnungsfunktion und Benutzerhandlung kann deklariert werden, indem die entsprechenden Anschlüsse in der graphischen Darstellung verbunden werden. Dieses erfolgt mit Hilfe der Maus (siehe Abbildung 94). Wie schon erwähnt, können auch Benutzeraktionen als Funktionen interpretiert werden. Die Kasusrolle `choice-set` aus der Aktionenbeschreibung des Konzeptes `Selection` wird in diesem Sinne zu einem Eingangsparameter während `choice` als Ausgangsparameter kenntlich gemacht wird.

3.2.7 Modellierung von Aktionen mit Beschreibungslogiken

Aktionenkonzepte werden durch HAMVIS für die entsprechende Anwendungsklasse bereitgestellt. Sie werden als Erweiterung des Kernmodells aus Abbildung 90 mit beschreibungslogischen Formeln beschrieben. Für die in dieser Arbeit als Leitbeispiel verwendete Anwendungsklasse der Layoutsysteme habe ich die Rollen und Konzepte aus Abbildung 95 und Abbildung 96 definiert. Die Definition erfolgt unter Rückgriff auf die Konzepte des HAMVIS-Grundmodells aus den Abbildungen 56 und 57.

Definition von Aktionenkonzepten im HAMVIS-Grundmodell

HAMVIS stellt die Deklarationsformen `define-action-concept` bzw. `define-primitive-action-concept` bereit.

```
(define-[primitive-]action-concept concept-name
    concept-expression
  [ (:asserted-concepts concept-expression ...) ]
  [ (:input role-name ...) ]
  [ (:output role-name ...) ]
  [ (:focused-input role-name ...) ]
  [ (:focussing-output role-name ...) ]
  [ (:derivation-relations ((role-name role-name ...) role-name) ...) ]
  [ (:gesture-realization-function function-name)
  [ (:parameters-to-present role-name ...) ]
  [ (:results-to-present role-name ...) ]
  [ (:pane-types (role-name pane-class) ...) ]
  [ (:required-methods generic-function-name ...) ]
  [ (:required-model-classes (role-name model-class) ...) ] )
```

Mit dieser Deklarationsform können Konzepte in ähnlicher Weise wie mit `define[-primitive]-concept` deklariert werden. Es werden jedoch noch weitere Optionen zugelassen, die im folgenden erläutert werden.

- `:asserted-concepts`

  Dient zur Definition einer Regel (`define-rule` *rule-name concept-name* (`and` *concept-expression* `...`)) in der KRSS-Syntax. Die Vergabe eines Regelnamens geschieht automatisch.

- `:input`, `:output`, `:focused-input`, `:focussing-output`

  Mit `:input` und `:output` werden die Kasusrollen gekennzeichnet, die entsprechend als Eingabe- und Ausgabeparameter einer Benutzeraktion gelten sollen.

  Durch die spezielleren Angaben `:focused-input` und `:focussing-output` wird festgelegt, daß bei der Durchführung der Aktion zur Laufzeit die in der angegebenen Kasusrolle stehenden Objekte fokussiert werden bzw. fokussiert sein müssen (vgl. die Definition von `selection` und siehe Abschnitt 3.2.9).

- `:derivation-relations`

  In einigen Fällen, ist es vorteilhaft, eine Aktion nicht durch das direkte Konzept eines der Rollenfüller zu beschreiben, sondern indirekt durch das Konzept eines Objektes, daß über eine Rollenkette von einem Rollenfüller einer Kasusrolle aus erreicht werden kann. Diese Objekte stehen nur indirekt mit der Aktion in Beziehung. Mit einer abgeleiteten Relation kann diese Beziehung explizit gemacht werden. Unter `:derivation-relations` wird die Rollenkette (Liste von Rollennamen) und die Rolle des Zielobjektes angegeben. Eine solche Rolle erscheint dann in der graphischen Darstellung als Rolle ohne Eingangs- und Ausgangsanschlüsse (für ein Beispiel siehe Abbildung 103).

- `:gesture-realization-function`

  Zur Anbindung von UIMS-Diensten wird hier eine Funktion angegeben (siehe Kapitel 3.5).

Die folgenden Optionen dienen schon zur Angabe von Visualisierungsinformationen für die Aktionenkonzepte.

- `:parameters-to-present`

  Hiermit werden diejenigen Kasusrollen gekennzeichnet, die mit einer Eingangs-Berechnungsfunktion verbunden werden müssen und zur Laufzeit mit Objekten gefüllt werden, die auf dem Bildschirm in einem Fenster dargestellt werden müssen.

- `:results-to-present`

  Hiermit werden diejenigen Kasusrollen gekennzeichnet, die mit einer Ausgangs-Berechnungsfunktion verbunden werden müssen und zur Laufzeit nach Ausführung der Aktion ebenfalls Objekte enthalten, die auf dem Bildschirm in einem Fenster dargestellt werden müssen.

- `:pane-types`

  Unter `:pane-types` werden die Fenstertypen angegeben, die zur Darstellung benötigt werden. Für die spezielle Auswahlaktion `selection-from-small-palette` ist dieses z.B. die Klasse `singe-choice-palette-pane`.

- `:required-methods`

  Für bestimmte Klassen von Teilfenstern müssen zur Darstellung von Objekten noch Methoden für bestimmte generischen Funktionen angegeben werden, deren Name hier spezifiziert wird. Für `selection-from-small-palette` ist dieses z.B. die Funktion `draw-palette-icon`.

- `:required-model-classes`

  Mit dieser Option können bei der Aktionsdefinition Anforderungen an das bei der Darstellung von Objekten zu verwendende Modell angegeben werden.

Die Optionen sind nicht für die Individuen relevant, sondern können als Attribute des Konzepts aufgefaßt werden.

Betrachten wir zunächst als Beispiel die Definition von `selection` in Abbildung 96. Eine Auswahl kann (unter normalen Bedingungen) durchgeführt werden, wenn mindestens ein Element in der Kasusrolle `choice-set` steht. Dieses muß vom Konzept `um-object` (`um` steht für Upper Model) sein. Die ausgewählten Objekte – eine Teilmenge der Rollenfüller von `choice-set` – werden durch die Rolle `choice` beschrieben. `choice-set` stellt die Eingabe und `choice` die Ausgabe der Aktion dar. Eine Auswahlhandlung, bei der nur ein einziges Element zur Auswahl steht, kann in gewissem Sinne als Bestätigung interpretiert werden.

```
(in-model hamvis-upper-model)

(define-primitive-role choice-set nil)
(define-primitive-role choice choice-set)

(define-primitve-action-concept selection
    (and user-action
         (at-least 1 choice-set) (all choice-set um-object)
         (at-least 1 choice) (all choice um-object))
  (:asserted-concepts action-with-focussing-output)
  (:input choice-set)
  (:focussing-output choice))

(define-action-concept single-object-selection
    (and selection
         (at-most 1 choice)))

(define-primitive-action-concept selection-from-small-palette
    (and single-object-selection
         (at-most 10 choice-set))
  (:parameters-to-present choice-set)
  (:pane-types (choice-set singe-choice-palette-pane))
  (:required-methods draw-palette-icon))

(define-primitive-role has-manipulated-object nil)

(define-concept spatial-action
    (and user-action
         (all has-manipulated-object spatial-object)))

(define-primitive-role has-localized-entity has-manipulated-object)
(define-primitive-role has-reference-object nil)
(define-primitive-role has-le-ro-relation nil)
(define-primitive-role has-previous-position-constraint nil)
(define-primitive-role has-new-position-constraint nil)
(define-primitive-role has-situative-context nil)

(define-primitive-action-concept localization
    (and user-action
         (at-least 1 has-localized-entity)
         (all has-localized-entity um-object)
         (at-least 1 has-reference-object)
         (all has-reference-object um-object)
         (at-least 1 has-le-ro-relation)
         (at-most 1 has-le-ro-relation)
         (all has-le-ro-relation static)
         (at-least 1 has-new-position-constraint)
         (all has-new-position-constraint position-constraint)
         (at-least 1 has-situative-context)
         (all has-situative-context input))
   (:asserted-concepts action-with-focused-input)
   (:focused-input has-localized-entity)
   (:input has-reference-object)
   (:output has-new-position-constraint))
```

Abbildung 95. Aktionenkonzepte für die Anwendungsklasse der Layoutsysteme (Teil 1).

In Abbildung 95 sind noch einige weitere Unterkonzepte zu `selection` aufgeführt. Bei einer Einfachauswahl (`single-object-selection`) ist die Maximalkardinalität der `choice`-Relation z.B. auf eins eingeschränkt. Bei einer Aktion vom Konzept `selection-from-small-palette` ist auch die Kandidatenmenge (Rolle `choice-set`) auf zehn Elemente beschränkt. In diesem Fall ist eine Realisierung an der Oberfläche möglich. Hierzu muß im zu erzeugenden Interaktionsfenster der XKL-Anwendung ein Fenster vom Typ `single-choice-palette-pane` untergebracht werden, in dem die jeweiligen Füller der Rolle `choice-set` präsentiert werden. Hierzu muß noch eine Methode für die generische Funktion `draw-palette-icon` zur Darstellung eines Piktogramms formuliert werden. Die Klasse `single-choice-palette-pane` wird als Standard-Interaktionsbaustein von HAMVIS in einer Bibliothek bereitgestellt. Zur Darstellung der Auswahlelemente der Palette wird die Funktion `draw-palette-icon` verwendet.

Die schon angesprochenen Lokalisierungshandlung soll genauer besprochen werden. Abbildung 95 enthält die zur Formalisierung notwendigen Deklarationen. Auch hier werden wiederum die Kasusrollen deklariert und in der Konzeptbeschreibung mit entsprechenden Einschränkungen versehen. Als Eingänge sind `has-localized-entity` und `has-reference-object` vorgesehen. Die Aktion liefert in `has-new-position-constraint` ein Ausgangsobjekt. Die mit `has-localized-entity` beschriebenen Anwendungsobjekte müssen durch eine vorhergehende fokussierende Aktion geliefert werden. Dieses ist in dem kleinen Beispiel aus Abbildung 95 vorzugsweise die oben diskutierte Auswahlaktion.

In Abbildung 96 werden speziellere Lokalisierungskonzepte deklariert. Hierzu werden definierte Konzepte mit notwendigen und hinreichenden Bedingungen verwendet. Wenn für eine Aktion bekannt ist, daß sie eine Lokalisierungsaktion ist, deren Rollen `has-localized-entity` und `has-reference-object` mit räumlichen Objekten gefüllt werden, genau dann handelt es sich auch um eine Aktion vom Konzept `spatial-localization`. Damit gelten auch die unter `:asserted-concepts` angegebenen Konzepte, wobei die Konzeptbeschreibung von `has-previous-position-constraints` aus der Rollenkette `(has-localized-entity · has-position-constraint)` abgeleitet wird. Da `has-localized-entity` eine Unterrolle von `has-manipulated-object` ist, gilt auch `spatial-action`, obwohl dieses nicht direkt angegeben worden ist, sondern durch Subsumtionsinferenzen auf der TBox-Ebene abgeleitet wird. In einem komplexeren Modell wird der Vorteil der Beschreibungslogiken deutlich.

```
(define-action-concept spatial-localization
    (and user-action
         localization
         (all has-localized-entity spatial-object)
         (all has-reference-object spatial-object))
  (:asserted-concepts (at-least 1 has-previous-position-constraint)
                      (all has-previous-position-constraint
                           spatial-position-constraint)
                      (all has-new-position-constraint
                           spatial-position-constraint))
  (:derivation-relations ((has-localized-entity has-position-constraint)
                          has-previous-position-constraint)))

(define-action-concept spatial-localization-in-xy-bounding-rectangle
    (and spatial-localization
         (at-most 1 has-localized-entity)
         (all has-le-ro-relation in)
         (all has-previous-position-constraint
              xy-bounding-rectangle-position-constraint))

  (:gesture-realization-function spatial-localization-in-xy-bounding-rectangle)
  (:asserted-concepts (all has-new-position-constraint
                           xy-bounding-rectangle-position-constraint)
                      (all has-reference-object non-overlapped-spatial-object))
  (:parameters-to-present has-reference-object)
  (:results-to-present has-localized-entity)
  (:required-model-classes (has-reference-object top-view-section-model-class)
                           (has-localized-entity geometry-model-class)))

(define-primitive-role has-moved-entity has-manipulated-object)

(define-primitive-action-concept movement
    (and user-action
         (at-least 1 has-moved-entity)
         (all has-moved-entity um-object)
         (at-least 1 has-new-position-constraint)
         (all has-new-position-constraint
              position-constraint-with-default))
  (:input has-moved-entity)
  (:output has-new-position-constraint-with-default))

(define-action-concept spatial-movement
    (and user-action
         movement
         (all has-moved-entity spatial-object))
  (:asserted-concepts spatial-action
                      (at-least 1 has-previous-position-constraint)
                      (all has-previous-position-constraint
                           spatial-position-constraint)
                      (all has-new-position-constraint
                           spatial-position-constraint))
  (:required-model-classes (has-moved-entity geometry-model-class))
  (:derivation-relations ((has-moved-entity has-position-constraint)
                          has-previous-position-constraint)))
```

Abbildung 96. Aktionenkonzepte für die Anwendungsklasse der Layoutsysteme (Teil 2).

Bei spezielleren Lokalisierungshandlungen ist dann bekannt, welche Einschränkungen für die Gestaltung der Visualisierungen gelten müssen (siehe `spatial-localization-in-xy-bounding-rectangle`). Etwas weiter unter wird die Definition von speziellen Verschiebehandlungen aufgezeigt. In Abbildung 97 wird ein spezielles Aktionenkonzept zur Bewegung von Objekten in einem rechteckigen Bereich gezeigt. Für dieses Konzept können dann entsprechende Visualisierungen und Interaktionswerkzeuge erzeugt werden.

```
(define-action-concept
    spatial-movement-within-bounding-rectangle-position-constraint-with-xy-default
  (and spatial-movement
       (all has-previous-position-constraint
            bounding-rectangle-position-constraint-with-xy-default))
  (:asserted-concepts
   (all has-new-position-constraint
        bounding-rectangle-position-constraint-with-xy-default)
   (all has-moved-entity non-overlapped-spatial-object))
  (:gesture-realization-function
     move-spatial-object-within-xy-bounding-rectangle-position-constraint)
  (:parameters-to-present has-moved-entity)
  (:presentation-types (has-moved-entity movable-object)))
```

Abbildung 97. Modellierung eines speziellen Aktionenkonzeptes zur Bewegung von Objekten.

Vergleich mit anderen Ansätzen zur Modellierung von Aktionen und Plänen

In der Literatur wurden vielfältige Ansätze zur Modellierung von Aktionen aufgezeigt. Eine Zusammenfassung von mehreren Aktionen wird auf höherer Ebene auch als Plan bezeichnet.

Für den Zugriff auf gespeicherte Pläne mit beschreibungslogischen Schlußmechanismen wurden Ansätze von Köhler [157] und Mittal et al. [208] entwickelt. In STRIPS-ähnlicher Weise werden Aktionen bzw. Pläne durch Vor- und Nachbedingungen beschrieben, wobei zur Beschreibung der Bedingungen Konzeptterme konstruiert werden. Beispielsweise können für zu erreichende Nachbedingungen durch Subsumtionsalgorithmen mögliche Pläne aus einer Bibliothek bestimmte werden. Auch Di Eugenio verwendet einen ähnlichen Ansatz zur Interpretation von speziellen Relativsätzen [70]. In dieser Anwendung kann die Reihenfolge von Teilaktionen durch speziell benannte Teil-von-Relationen (`has-substep-1`, `has-substep-2`) repräsentiert werden. In Anwendungen, in denen dieses nicht ausreichend ist, müssen mächtigere Konstrukte verwendet werden. Devanbu und Litman stellen mit dem System CLASP eine Zerlegungsbeschreibung durch reguläre Ausdrücke vor [68] (siehe auch die Rekonstruktion von Borgida in [26]). Die Zerlegung einer Aktivität A in Handlungsalternativen a1,a2,a3,… kann im Sinne von CLASP als regulärer Ausdruck gedeutet werden. Es werden alle Handlungsfolgen (a1 | a2 | a3 | …)* beschrieben. Devanbu und Litman erweitern den Subsumtionsbegriff von CLASSIC durch die Teilmengenbeziehungen von regulären Mengen, die durch die regulären Ausdrücke der Zerlegungsbeschreibung definiert sind. Im Zusammenhang mit der Realisierung von zusammengesetzten Handlungen durch Interaktionsgesten werden ich auf dieses Thema zurückkommen.

Sequentielle Zerlegungen können in RAT modelliert werden [122]. Obwohl RAT nur einfache sequentielle Zerlegungsformen unterstützt, können Pläne simulativ durchgeführt werden, wodurch eine zeitliche Projektion der Effekte von Aktionen untersucht werden kann [331]. Zeitliche Beziehun-

gen zwischen Aktionen und Plänen wurden ebenfalls von Weida und Litman [335] sowie Artale und Franconi [13] untersucht, sind aber für HAMVIS zur Zeit nicht relevant.

Bei der Auswahl eines Repräsentationskonstrukts sind die für eine Anwendung zu modellierenden Inferenzdienste wichtig. Zur Realisierung der in Abschnitt 3.2.1 aufgezeigten Inferenzschemata können Handlungsmöglichkeiten des Endbenutzers als ABox-Instanzen abgebildet werden. Auf die Verwendung von komplexeren Zerlegungsbeschreibungen für zusammengesetzte Aktionen, wie sie z.B. CLASP ermöglicht, komme ich in Abschnitt 3.3.11 zurück.

Ein Inferenzdienst, der für die Beziehungen zwischen den Kasusrollen von Berechnungsfunktionen und Benutzeraktionen besonders wichtig ist und der es gestattet, Inferenz über den „Datenfluß“ zwischen Rollen zu modellieren, wird von keinem der oben aufgeführten Systeme angeboten. Ich möchte den Inferenzdienst „Propagierung von Typinformationen“ nennen und ihn zunächst wieder anhand der interaktiven HAMVIS-Oberfläche erläutern.

3.2.8 Propagierung von Typinformationen

Betrachten wir die Verwendung der Aktionenmodelle bei der Definition der Aktionenzerlegung einer konkreten Applikation. In Abbildung 94 wurden die Anschlüsse der Aktionen `compute-prototype-objects` und `select-cabin-object` mit der Maus verbunden. Eine Verbindung von Anschlüssen wird durch eine Pipeline visualisiert (siehe Abbildung 98).[93]

93. Die Linienführung der Pipeline erfolgt automatisch (siehe hierzu auch Möller [211] und Haarslev und Möller [108]).

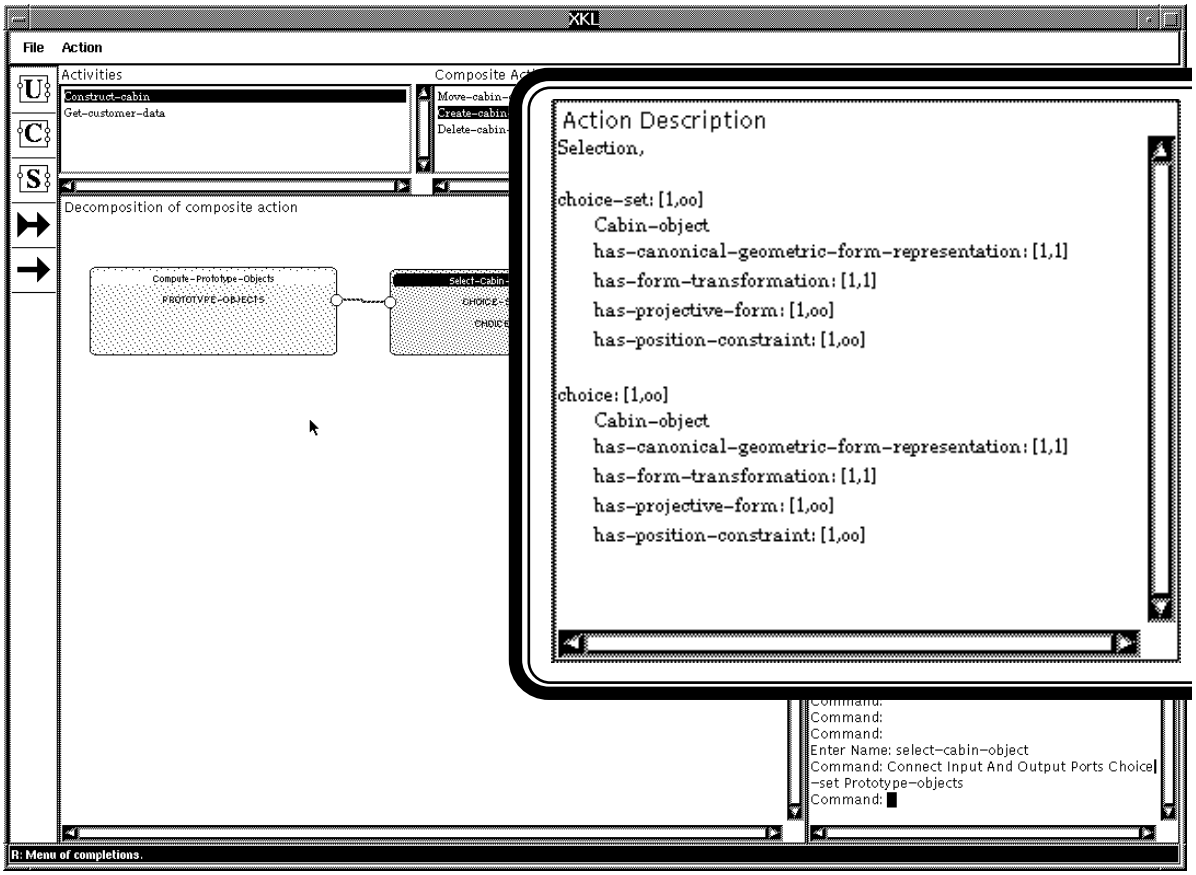


Abbildung 98. Kopplung von Berechnungsfunktionen und Benutzeraktionen mit Ableitung von Typinformationen für Anschlüsse.

Da die Objekte, die von `compute-prototype-objects` zur Laufzeit berechnet werden, auch durch die Benutzeraktion manipuliert werden, müssen die Konzeptinformationen „propagiert“ werden. Die Einschränkungen für den Ausgabeparameter `prototype-objects` von `compute-prototype-objects` gelten also auch für die Kasusrolle `choice-set` und umgekehrt. In Abbildung 98 sind die im Fenster `Action Description` präsentierten Typinformationen mit der Lupe hervorgehoben. Sie werden von HAMVIS automatisch hergeleitet. Zu beachten ist, daß in der Konzeptbeschreibung von `selection` die Füller der Rolle `choice` als eine Untermenge von `choice-set` deklariert wurden. Daher werden die Konzepteinschränkungen auch für `choice` übernommen (siehe Abbildung 98). Welche Rolle spielt nun die Beschreibungslogik bei dieser Art von Typpropagierung?

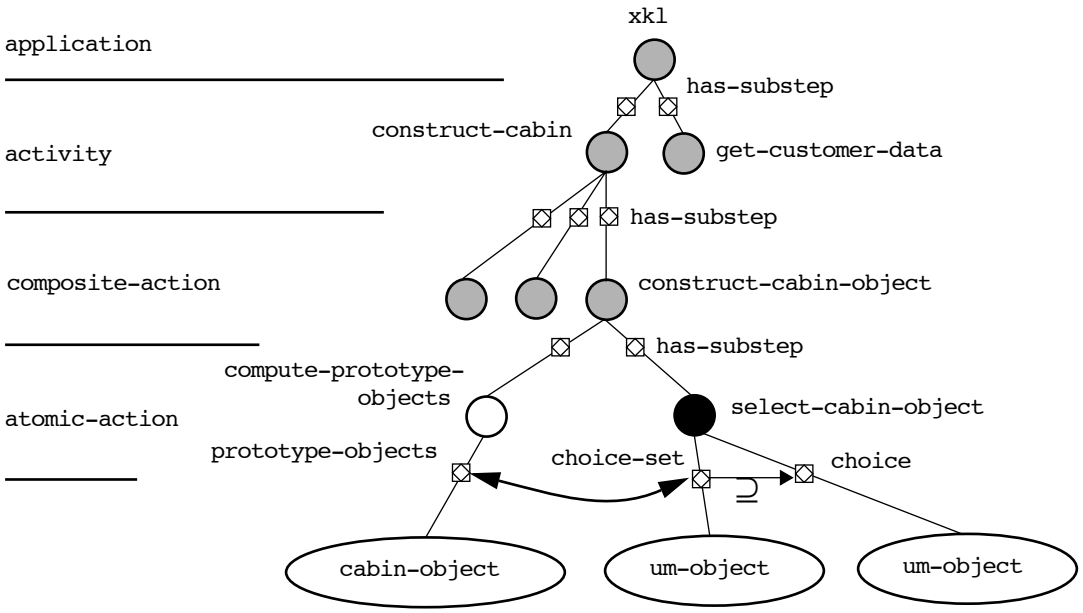


Abbildung 99. Überblick über die Beziehungen zwischen Individuen in der Beschreibungslogik
graue Kreise stehen für Individuen, Quadrate mit Karos symbolisieren Rolle und Ellipsen stellen Konzepte dar (um steht für Upper Model). Durch den Doppelpfeil ist die Pipeline gekennzeichnet.

In Abbildung 99 werden die in der ABox verwalteten Relationen zwischen den prototypischen Aktionen dargestellt. Die durch die Pipeline ausgedrückte Information läßt sich in Termen der Beschreibungslogik als ABox-Aussagen interpretieren:

```
(state (instance select-cabin-object (all choice-set cabin-object)))
(state (instance compute-prototype-objects (all prototype-objects um-object)))
```

Die erste Aussage propagiert die Typinformationen von links nach rechts und modelliert damit auf der konzeptuellen Ebene die Verwendung der von der Berechnungsfunktion gelieferten Objekte in einer Benutzeraktion `select-cabin-object`. Die zweite Aussage sorgt dafür, daß die Berechnungsfunktion `compute-prototype-object` nur solche Objekte liefert, die in der Aktion auch verwendet werden können (Propagierung von rechts nach links).

Durch die Propagierung werden die „Typinformationen“ für die Parameter und Werte von Berechnungsfunktionen eingeschränkt. Benutzeraktionen können durch Rollenfüllereinschränkungen ggf. automatisch spezialisiert werden.

Wenn immer sich die Rolleneinschränkungen der hier angesprochenen Individuen ändern oder neue Pipelines hinzukommen oder gelöscht werden, müssen entsprechende ABox-Aussagen zur „Propagierung“ der Informationen generiert werden. Beim Löschen von Pipelines müssen die Aussagen entsprechend wieder zurückgezogen werden.

Da die in der Aktionendekomposition zur Entwicklungszeit verwalteten Modelle auf UIMS-Dienste abgebildet werden, also zur Laufzeit nicht mehr vorhanden zu sein brauchen, ist es für HAMVIS nicht notwendig, zu deklarieren, daß die Rollenfüller *gleich* sind. Beschreibungslogiken stellen auf der Ebene der Konzeptbeschreibungen auch kein Repräsentationskonstrukt für die Gleichheit von Rollen-

füllern bereit (hier ist insbesondere die Diskussion um das „Role-Value-Map“-Konstrukt relevant[94]). Es können jedoch keine Füller der Rollen `prototype-objects` bzw. `choice-set` eingesetzt werden. Zur Entwicklungszeit sind die konkreten Objekte ja noch nicht bekannt. Bei dem Beispiel `choice-set` wird deutlich, daß eine Einschränkung der Modellierungssprache auf Attribute auch aus Sicht der Aktionenmodellierung wenig sinnvoll ist.

Die im Kontext von HAMVIS benötigten Inferenzschemata lassen sich wie folgt beschreiben. Wenn jeweils zwei Rollen `r1` und `r2` zweier Instanzen `a` und `b` mit einer Pipeline verbunden werden, so muß sichergestellt werden, daß folgende Formel zweiter Ordnung gilt:

$$\forall a, b, x, y \in D(r1(a, x) \wedge r2(b, y) \Rightarrow (\forall P(P\langle x\rangle \Leftrightarrow P(y))))$$

Es ist also nur wichtig, daß die gleichen *Rollenfüllereinschränkungen* gelten. Dieses wird durch entsprechend generierte ABox-Aussagen nach dem obigen Schema ausgedrückt. Zur Erzeugung entsprechender ABox-Aussagen bei Änderung der Rollenfüllereinschränkungen war ein Zusatzmodul zu CLASSIC notwendig, der in der Arbeit von Brüning [35] mit Hilfe der Erweiterungsschnittstelle von CLASSIC realisiert wurde.

Als nächstes möchte ich auf die oben schon angesprochene Bedeutung der Fokussierung von Anwendungsobjekten durch Aktionen zu sprechen kommen. Anhand der zusammengesetzten Handlung `create-cabin-object` wird erläutert, in welcher Weise eine Fokussierung von Objekten im Diskurs zur Laufzeit bei der Aktionenmodellierung zur Entwicklungszeit berücksichtigt werden kann und wie das hierzu notwendige Wissen formalisiert wird.

3.2.9 Attentionale Diskursaspekte in der Handlungsstruktur: Fokussierung und deren Propagierung

In Abbildung 100 wird die weitere Modellierung der zusammengesetzten Handlung `create-cabin-object` gezeigt. Es sind folgende Komponenten vorgesehen: eine Berechnungsfunktion `compute-placement-areas` und eine Benutzeraktion `localize-cabin-object`. Für die Funktion `compute-placement-areas` wurden zwei Parameter `selected-object` (Eingang) und `possible-placement-areas` (Ausgang) definiert. Für die Typbeschreibung des Parameters `possible-placement-areas` wurde schon das Konzept `placement-area` angegeben (siehe das Fenster `Action Description` in Abbildung 100), da nur Objekte dieses Typs geliefert werden.

94. Das eventuell naheliegende Konzept der Gleichheit von Rollenfüllern wie es aus Beschreibungslogiken bekannt ist, erfordert die Einschränkung der Rollenkardinalität auf eins, d.h. die Rollen müssen Attribute sein, da die Sprache sonst unentscheidbar wird [275]. Auch für andere Rolleninteraktionsspezifikationen kann für das Subsumtionsproblem bei Verwendung von Standardrepräsentationskonstrukten aus der Beschreibungslogik Unentscheidbarkeit gezeigt werden [114].

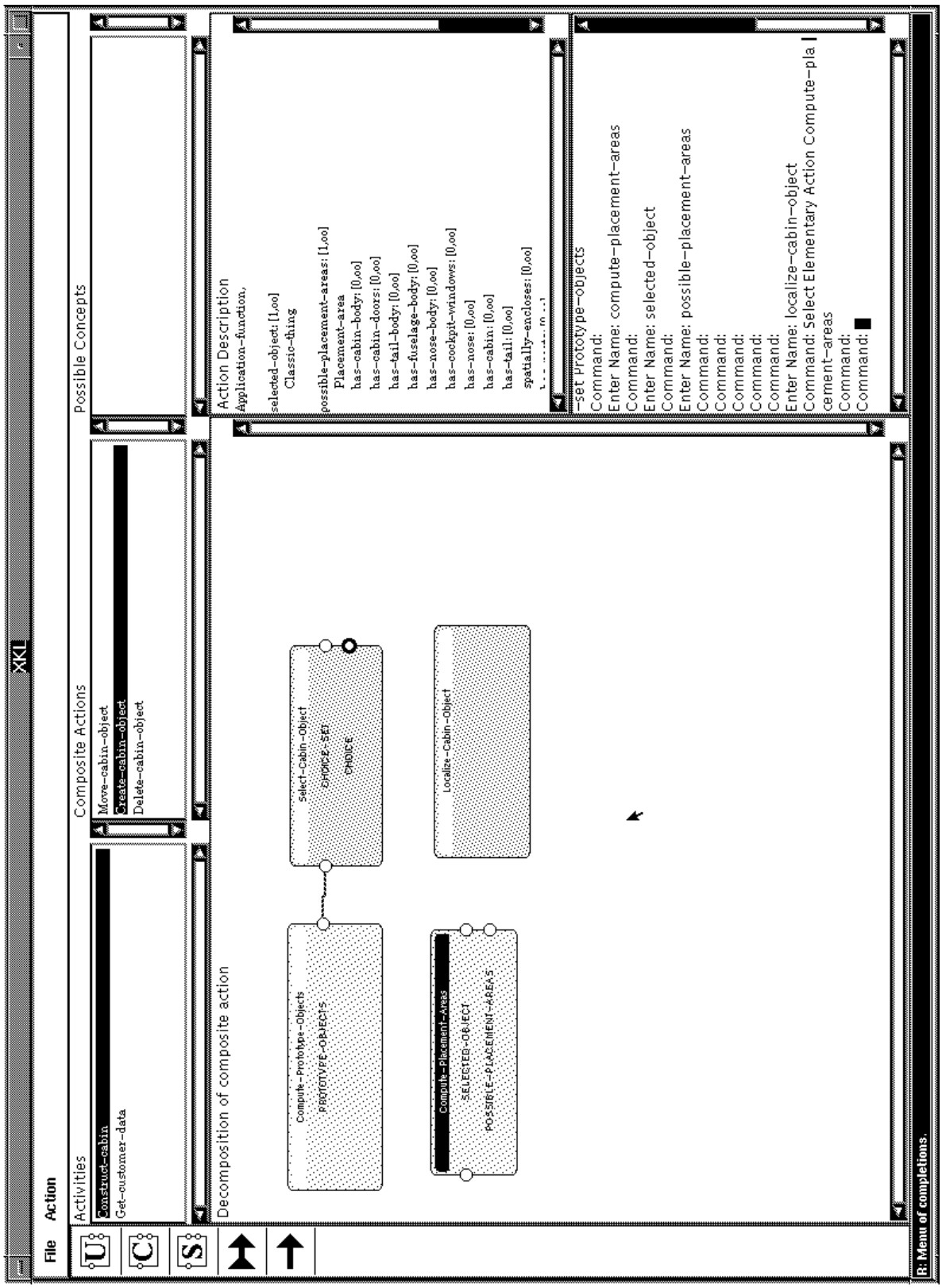


Abbildung 100. Fortsetzung der Modellierung der zusammengesetzten Handlung `create-cabin-object`.

Die Idee hinter der Funktion `compute-placement-areas` ist, daß zu jedem gewählten Objekt die möglichen Plazierungsbereiche zur Laufzeit ausgerechnet werden. Durch die Typpropagierung wird beim Verbinden von `choice` (Benutzeraktion `select-cabin-object`) und `selected-object` (Funktion `compute-placement-areas`) auch für `selected-object` der Typ `cabin-object` abgeleitet. Die Einrichtung einer Pipeline wird in Abbildung 101 skizziert.

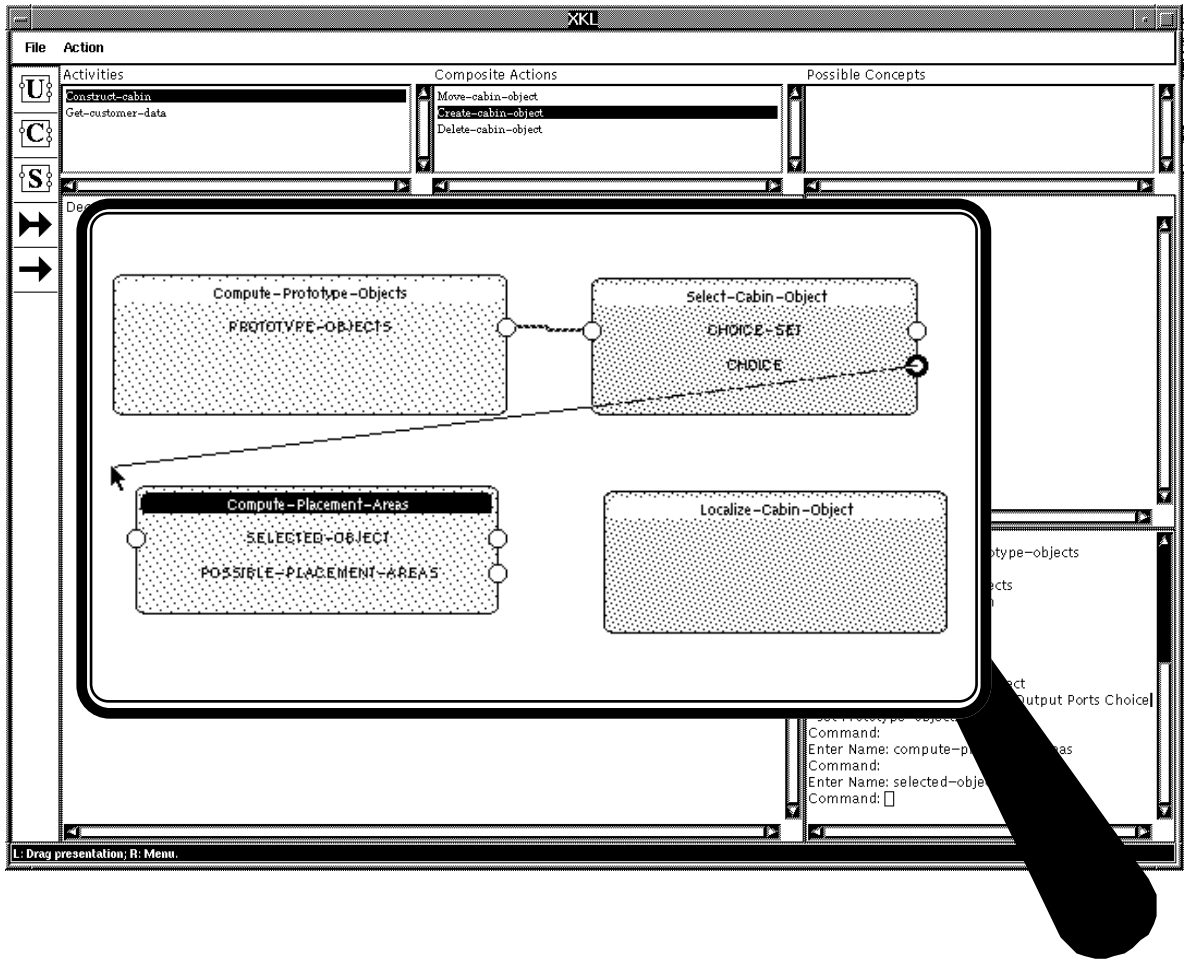


Abbildung 101. Kopplung der Benutzeraktion `select-cabin-object` mit der Berechnungsfunktion `compute-placement-areas`.

Zu beachten ist die Markierung des Anschlusses des Ausgangsparameters `choice` von `select-cabin-object` durch einen dickeren Kreis. Aus dem Aktionenmodell geht hervor, daß die durch `choice` beschriebenen Objekte fokussiert sind, d.h. es können sich Aktionen anschließen, die auf diesen Objekten agieren können, ohne daß dem Benutzer noch einmal Information über die Objekte mitgeteilt werden müssen. In der Aktionenmodellierungsoberfläche wird ein „fokussierender“ Parameter durch einen verdickten Anschlußkreis dargestellt. Nach der Verbindung der Anschlüsse wie in Abbildung 101 angedeutet, wird auch die Fokussierungsinformation weiterpropagiert, d.h. auch die Funktion `compute-placement-areas` zeichnet sich durch entsprechende Anschlüsse aus (siehe Abbildung 102).

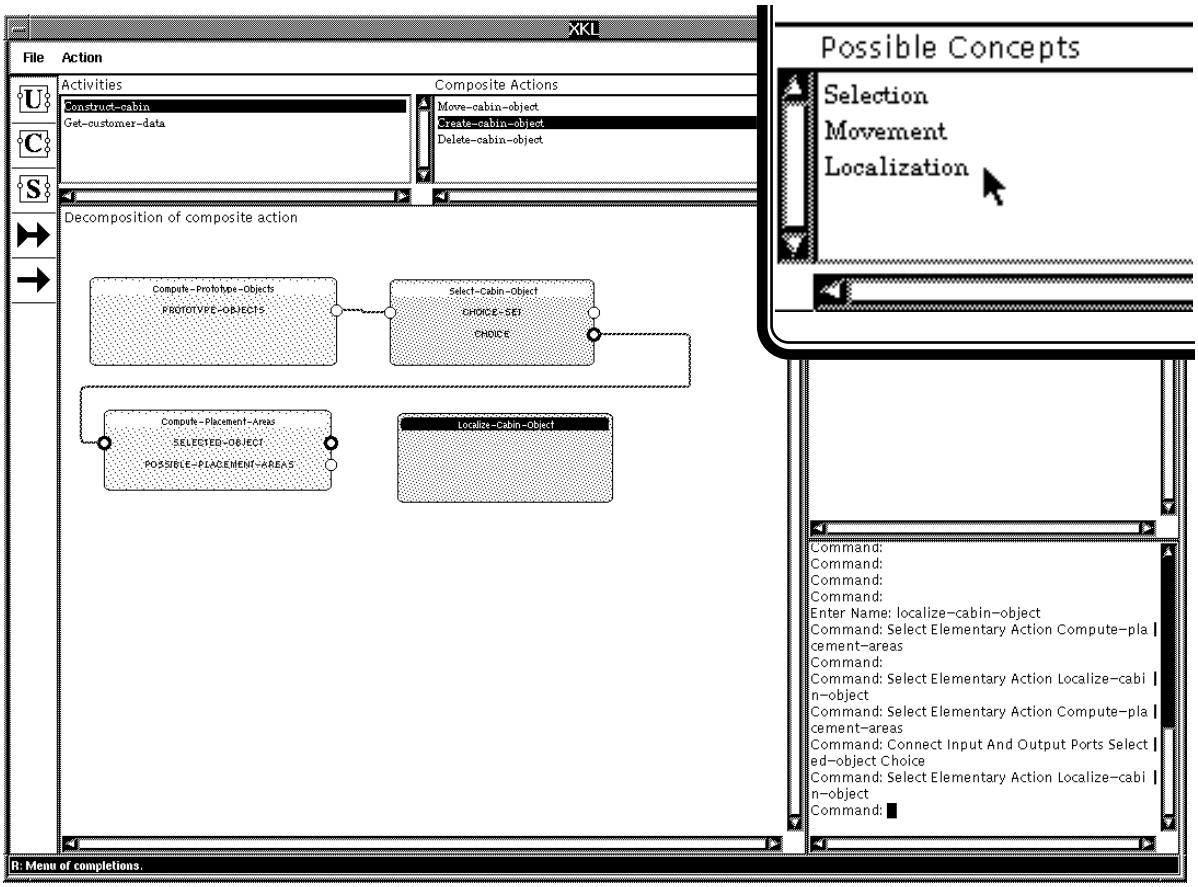


Abbildung 102. Propagierung von Fokussierungsinformationen.

Ein fokussierender Anschluß kann nur mit einem Anschluß verbunden werden, der fokussierte Objekte fordert. Dieses wird an dem Beispiel der Lokalisierungsaktion `localize-cabin-object` diskutiert. Diese Aktion ist in Abbildung 102 markiert, und es kann in der Tabelle rechts oben ein Konzept ausgewählt werden (siehe die Lupendarstellung). Nehmen wir an, es wird `Localization` gewählt. HAMVIS ermittelt die Rollen zur Kennzeichnung der Lokalisierungsaktion aus seinem Aktionenmodell. Sie sind in Abbildung 103 innerhalb des Aktionenrechtecks eingezeichnet.[95] Der Parameter `has-localized-entity` von `localize-cabin-object` erfordert eine vorherige Fokussierung (siehe die Deklaration von `localization` im Aktionenmodell in Abbildung 95).

Die Funktion `compute-placement-areas` liefert zwei Objekte, erstens das selektierte Objekt und zweitens die für dieses Objekt möglichen Plazierungsbereiche. Der Anschluß für `selected-object` wird mit dem Anschluß für `has-localized-entity` und der Anschluß für `possible-placement-areas` wird mit `has-reference-object` verbunden. Die entsprechenden Pipelines überkreuzen sich (siehe Abbildung 103).

Für die Verlegung von Pipelines ist es vorteilhaft, bei jeder Funktion oder Benutzeraktion alle Eingänge gleich weiterzuleiten, d.h. Eingangsparameter sind immer auch Ausgangsparameter. Die Umkehrung gilt jedoch nicht.

95. Das vorherige Rechteck wird hierzu automatisch vergrößert.

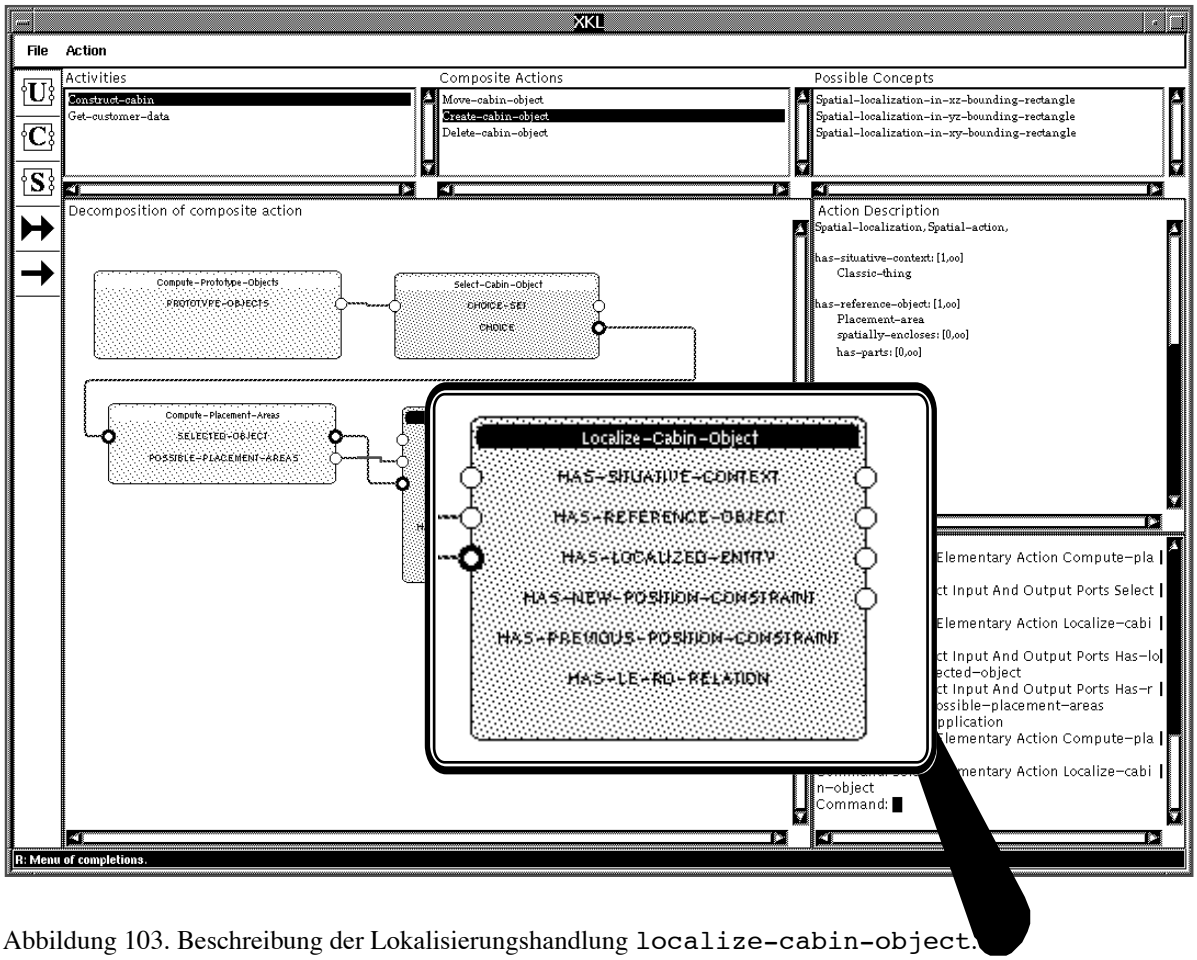


Abbildung 103. Beschreibung der Lokalisierungshandlung `localize-cabin-object`.

Eine Verlegung einer Pipeline zwischen einem fokussierenden Parameter einer Funktion oder einer Benutzeraktion und einem nicht fokussierten Anschluß einer Benutzeraktion ist nicht möglich. Betrachtet man die hier vorgestellte Form der Aktionenmodellierung als visuelle Sprache, so wird deutlich, daß Aspekte der Gestaltung der Benutzungsschnittstelle hier durch syntaktische Beschränkungen der visuellen Aktionenspezifikationssprache von HAMVIS repräsentiert werden.

3.2.10 Ableitung impliziter Information

Im Fenster `Action Description` aus Abbildung 103 ist zu erkennen, daß durch die Typpropagierung z.B. die Kasusrolle `has-reference-object` schon als Füllereinschränkung das Konzept `placement-area` bekommen hat. Die Rolle `has-localized-entity` ist durch die Pipeline auf `cabin-object` eingeschränkt. Sowohl `placement-area` als auch `cabin-object` sind Unterkonzepte von `spatial-object` (siehe das Basismodel der Anwendung `xkl-model` aus Abbildung 67). Damit sind die hinreichenden Bedingungen für `spatial-localization` erfüllt, ein definiertes Aktionenkonzept aus Abbildung 96. Die Aktion `localize-cabin-object` ist also nicht nur eine Aktion vom Konzept `localization`, sondern sogar von `spatial-localization`. Durch die Implikationen von `spatial-localization` ergeben sich weitere Einschränkungen. In Abbildung 103 stehen in der Tabelle `Possible Concepts` schon die Unterkonzepte von `spatial-`

localization zur weiteren Beschreibung der Lokalisierungsaktion zur Auswahl bereit. Die Stärke der Beschreibungslogiken kommt bei der Realisierung dieser Art von Inferenzen voll zur Geltung. Mit Hilfe der beschreibungslogischen Formalisierung können Schlußfolgerungen über Aktionen modelliert werden, auch wenn die konkreten, manipulierten Objekte noch gar nicht als Instanzen bekannt sind, sondern nur über konzeptuelle Informationen beschrieben werden.

Beschreibung von Aktionenkonzepten

Bei sehr umfangreichen Modellen für Aktionenkonzepte kann die Auswahl von Konzepten über die Tabelle Possible Concepts unübersichtlich werden. Es ist daher auch möglich, die Aktion im Fenster Action Description zu beschreiben. Hierzu können auch die abgeleiteten und statischen Parameter einer Aktion verwendet werden. In Abbildung 103 wird bei der Aktion localize-cabin-object der Parameter has-le-ro-relation[96] aufgeführt. Durch diese Relation ist die Beziehung zwischen den lokalisierten Entitäten und den Referenzobjekten beschrieben. Wird hier z.B. interaktiv durch Mausklick auf die Beschreibung der Rolle im Fenster Action Description die Einschränkung „in“ vorgegeben, so wird auch die Menge der möglichen Spezialisierungen in der Tabelle Possible Concepts entsprechend eingeschränkt. Die Konzeptauswahl ist dann übersichtlicher. Unter Umständen reicht eine solche Angabe auch zur automatischen Spezialisierung einer Aktion (siehe den vorigen Abschnitt).

Einschränkung des Implementationsmodells durch Typpropagierung

Durch die Aktion localize-cabin-object wird zur Laufzeit der XKL-Anwendung ein Objekt vom Konzept spatial-position-constraint erzeugt. Diese Lokalisierungsinformation muß noch am zu lokalisierenden Objekt der Rolle has-localized-entity vermerkt werden. Hierzu dient eine „nachgeschaltete“ Speicherfunktion. Die vollständige Beschreibung von create-cabin-object wurde schon am Anfang dieses Abschnitts in Abbildung 89 gezeigt. In dieser Abbildung ist die Speicherfunktion schon eingefügt und entsprechend mit Pipelines „verdrahtet“.[97] Die für die Parameter der markierten Speicherfunktion store-new-position-constraint abgeleiteten Typinformationen sind im Teilfenster Action Description abzulesen. Sie brauchen nicht explizit eingegeben zu werden, sondern können durch automatische Ableitungsschritte inferiert werden. Dieses Beispiel verdeutlicht erneut die Wichtigkeit des Schließens über konzeptuelle Informationen. Bedingt durch die Wahl des Konzepts einer Benutzeraktion muß durch die nachfolgende Speicherfunktion zur Laufzeit der XKL-Anwendung ein bestimmter Dienst realisiert werden. Dieser Dienst ist durch die Konzeptangaben für die Eingangsparameter der Funktion gekennzeichnet. Im nächsten Abschnitt wird die Umsetzung der Aktionenmodellierung in ein Laufzeitsystem genauer beschrieben.

---

96. LE = Located Entity, RO= Reference Object.

97. Der Verlegealgorithmus für die Pipelines vermeidet nicht eine eventuelle Überlagerung von Teilsegmenten. Wie aus Abbildung 89 hervorgeht, kann eine Pipeline mit der Maus selektiert werden. Sie wird dann entsprechend hervorgehoben. Viele parallele, dicht zusammenliegende Leitungen lassen sich schwer verfolgen, so daß die Überlagerung auch ästhetische Vorteile hat. Eine einzelne Pipeline kann durch „Berühren“ mit der Maus hervorgehoben werden.

3.2.11 Umsetzung einer Aktionenmodellierung in ein Laufzeitsystem

Durch die Wahl der Gestaltung der graphischen Oberfläche wird die Umsetzung der Aktionenmodellierung in ein Datenflußnetz schon nahegelegt. Die Zerlegung einer zusammengesetzten Aktion mit den entsprechenden Pipelines kann als Petri-Netz repräsentiert werden. Eine Aktion entspricht einer Transition. Für eine Pipeline wird eine Stelle eingesetzt.

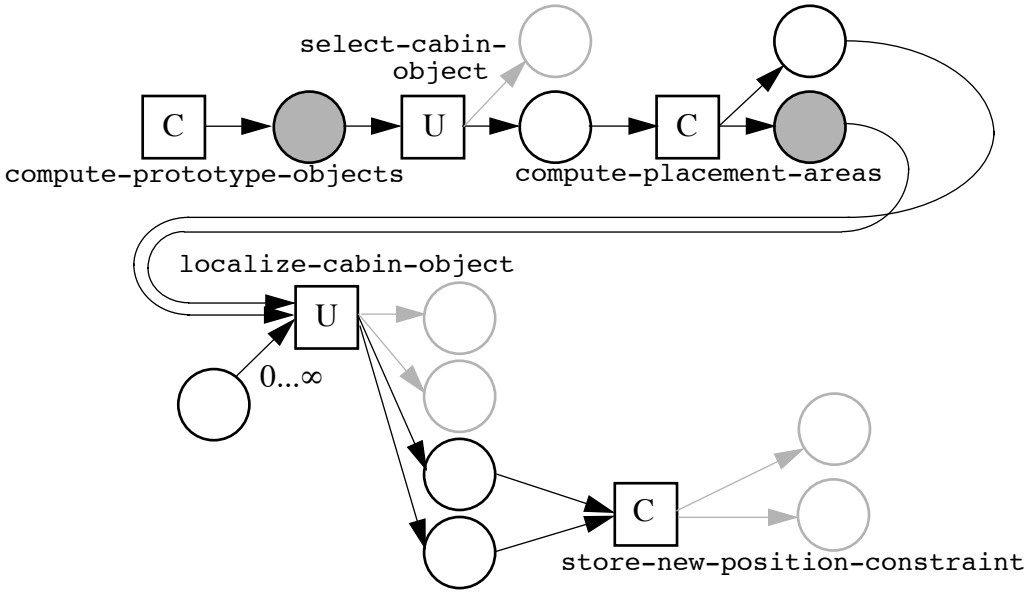


Abbildung 104. Repräsentation der Zerlegung von `create-cabin-object` als Petri-Netz.

In Abbildung 104 ist das Netz der zusammengesetzten Aktion `create-cabin-object` dargestellt. Transitionen mit „C“ bzw. „U“ stehen entsprechend für Berechnungsfunktion bzw. Benutzeraktion. Wenn eine Aktivität zur Laufzeit gestartet wird (das zugehörige Interaktionsfenster erscheint auf dem Bildschirm), werden die Transitionen ohne Eingangsstellen gefeuert, d.h. die zugehörigen Berechnungsfunktionen werden evaluiert. Das hat in dem konkreten Beispiel aus Abbildung 104 also zur Folge, daß die Berechnungsfunktion `compute-prototype-objects` evaluiert wird. Es wird die Menge der möglichen Prototypobjekte in die Ausgangsstelle abgelegt. Damit sind die Füller der Kasusrolle `choice-set` der Benutzeraktion `select-cabin-object` bekannt. Die Objekte, die sich in der Stelle befinden, können an der Oberfläche zur Auswahl dargestellt werden. Durch die Darstellung der Objekte an der Oberfläche wird der Benutzer in die Lage versetzt, eine Handlung durchzuführen. Nach der Ausführung der Benutzerhandlung werden die Ausgangsstellen der Transition `select-cabin-object` belegt. Die Kandidatenmenge (`choice-set`) wird in die obere Stelle weitergeleitet, das gewählte Objekt (`choice`) wird in die untere Stelle gelegt. Damit ist die Startbedingung für die Transition `compute-placement-areas` erfüllt. Die Funktion wird evaluiert und die Ergebnisstellen werden gefüllt. Oben erscheint wieder das Eingangsobjekt und unten wird eine Menge von Plazierungsbereichen ermittelt. Die Plazierungsbereiche werden auf dem Bildschirm dargestellt. Der Benutzer kann einen Bereich auswählen (Transition `localize-cabin-object`) und sorgt damit dafür, daß die vier Ausgangsstellen gefüllt werden. Am Ende wird die Funktion `store-new-position-constraint` evaluiert.

Aus Sicht eines UIMS ist damit ein Durchgang durch den Eingabe-Berechnungs-Zyklus beendet (vgl. Abbildung 9). Objekte, die noch nicht aus den Stellen „abgeholt“ wurden, werden entfernt. Der Zyklus beginnt von neuem, indem alle Marken entfernt und die Transitionen ohne Eingangsstelle neu evaluiert werden. Auf die Aktualisierung der graphischen Ausgabe wurde in Kapitel 2.1.4 schon kurz eingegangen, ich komme in Kapitel 3.3.10 darauf zurück.

Die in grau dargestellten Stellen enthalten zur Laufzeit Anwendungsobjekte, die an der Oberfläche in Form von graphischen Darstellungen präsentiert werden müssen. Die Art der Präsentation hängt von dem speziellen Aktionskonzept ab, mit dem die Aktion beschrieben wird. Das bedeutet wiederum, daß zur Generierung von Oberflächenelementen auch ein gewisses Spezialisierungsniveau in der Hierarchie der Aktionenkonzepte erreicht werden muß. Die Aktionenmodellierung ist also erst beendet, wenn die Benutzeraktionen genügend genau beschrieben worden sind.

3.2.12 Zwang zur Spezialisierung

Für die Generierung von Visualisierungen und Interaktionswerkzeugen müssen für Aktionen ausreichend spezielle Konzepte festgelegt werden. In Abbildung 103 standen schon in der Tabelle `Possible Concepts` die Spezialisierungen für `localize-cabin-object` bereit. Wenn wir die jeweiligen Definitionen im Aktionenmodell in Abbildung 96 betrachten, so wird deutlich, daß für `spatial-localization-in-xy-bounding-rectangle` eine Visualisierung generiert werden kann. Im Aktionenmodell sind Angaben zu den zu präsentierenden Parametern (Kasusrollen) vermerkt, die die entsprechenden Stellen im Laufzeitnetzwerk markieren (hier grau dargestellt). Außerdem sind Einschränkungen für die bei der Objektdarstellung zu verwendende Modellklasse eingetragen (siehe Abbildung 96).

3.2.13 Eine alternative Benutzeraktion

Im Aktionenmodell ist für die Aktivität `construct-cabin` noch die zusammengesetzte Aktion `move-cabin-object` vorgesehen. In Abbildung 106 ist deren Zerlegung dargestellt. Durch eine Funktion `compute-movable-objects` wird die Menge der verschiebbaren Objekte ermittelt. Es handelt sich wiederum um Instanzen von `cabin-object` (siehe die Beschreibung im Fenster `Action Description`)

Für die Benutzeraktion `move-an-object` wurde das Konzept `movement` gewählt, wodurch sich die entsprechenden Kasusrollen mit ihren Anschlüssen ergeben. In Abbildung 107 wird gerade die Pipeline zwischen `compute-movable-objects` und `move-an-object` verlegt. Da die Aktion `move-an-object` markiert ist, läßt sich nunmehr deren konzeptuelle Beschreibung im Teilfenster `Action Description` ablesen. Nach der Verbindung der Anschlüsse durch die Pipeline sind die Rollenfüller von `has-moved-entity` nicht mehr nur Instanzen von `um-object`, sondern `cabin-object`.

Aus diesen Angaben kann automatisch eine Spezialisierung hergeleitet werden. Es sind die hinreichenden Bedingung des definierten Konzepts `spatial-movement` erfüllt (siehe das HAMVIS-Aktionenmodell in Abbildung 96). In Abbildung 108 sind die resultierenden Implikationen dargestellt.

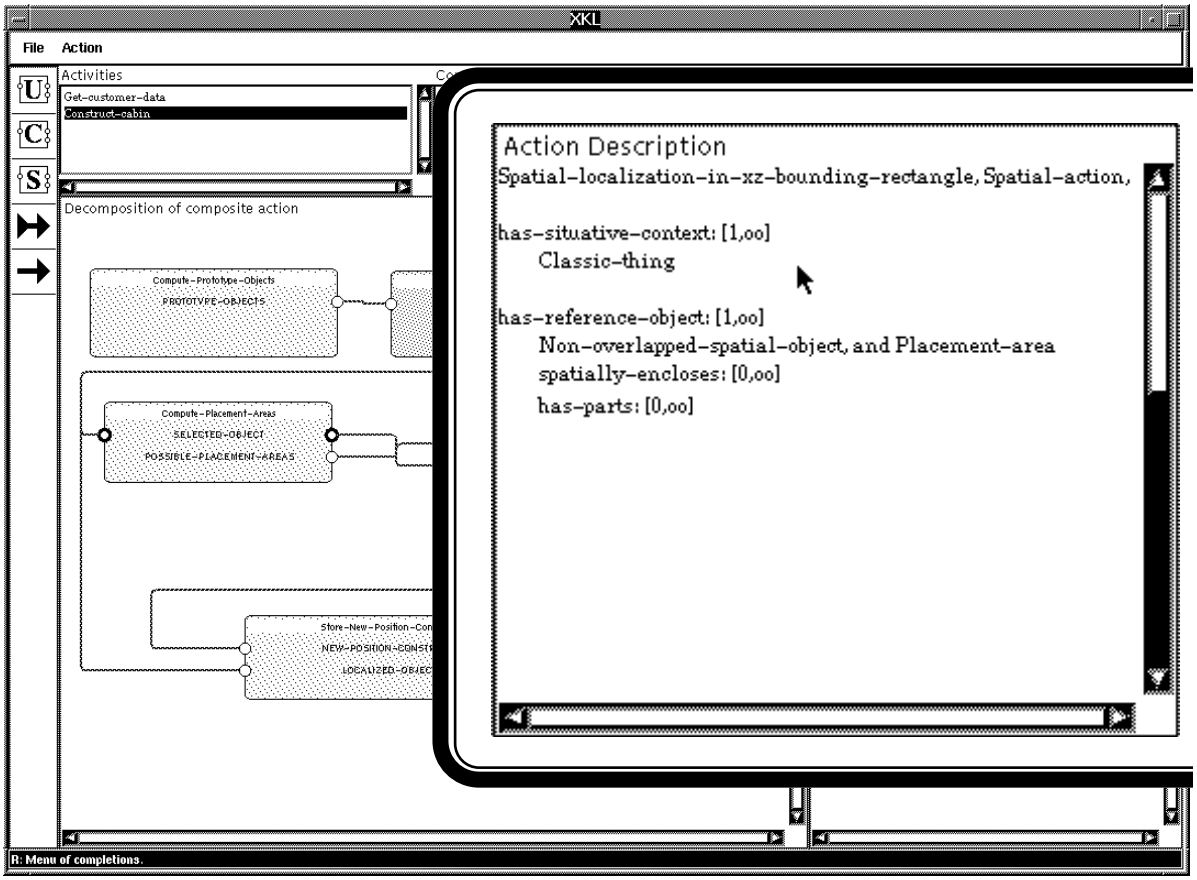


Abbildung 105. Auswahl der Spezialisierung `spatial-localization-in-xy-bounding-rectangle`.

Durch die Auswahl des spezielleren Konzeptes `spatial-localization-in-xy-bounding-rectangle` ist die Aktion `localize-cabin-object` ausreichend weit spezialisiert.[98] Durch die Spezialisierung wird gefordert, daß die Objekte der Kasusrolle `has-reference-object` vom Konzept `non-overlapped-spatial-object` sind (vgl. die Implikationen von `spatial-localization-in-xy-bounding-rectangle` in Abbildung 96). Dieses ist für die zur Realisierung der Aktion eingesetzte Interaktionsform notwendig.

Auch die Handlung `select-cabin-object` muß noch weiter spezialisiert werden. Nehmen wir an, aus der hier präsentierten Beispielaktionenmodellierung wird `selection-from-small-palette` gewählt (siehe Abbildung 95). Damit ist die Modellierung der zusammengesetzten Aktion `create-cabin-object` beendet. Dieses ist jedoch nicht die einzige Handlungsalternative innerhalb der Aktivität `construct-cabin`. Bevor auf die weitere Verarbeitung der Präsentationsinformationen, die sich durch das Modell von `create-cabin-object` ergeben, eingegangen wird, möchte ich im folgenden noch die Definition von `move-cabin-object` skizzieren.

98. Graphisch wird dieses hier durch einen dickeren Rand angezeigt. In der Implementierung für einen Farbbildschirm wechselt die Farbe von hellblau nach hellgrün.

.

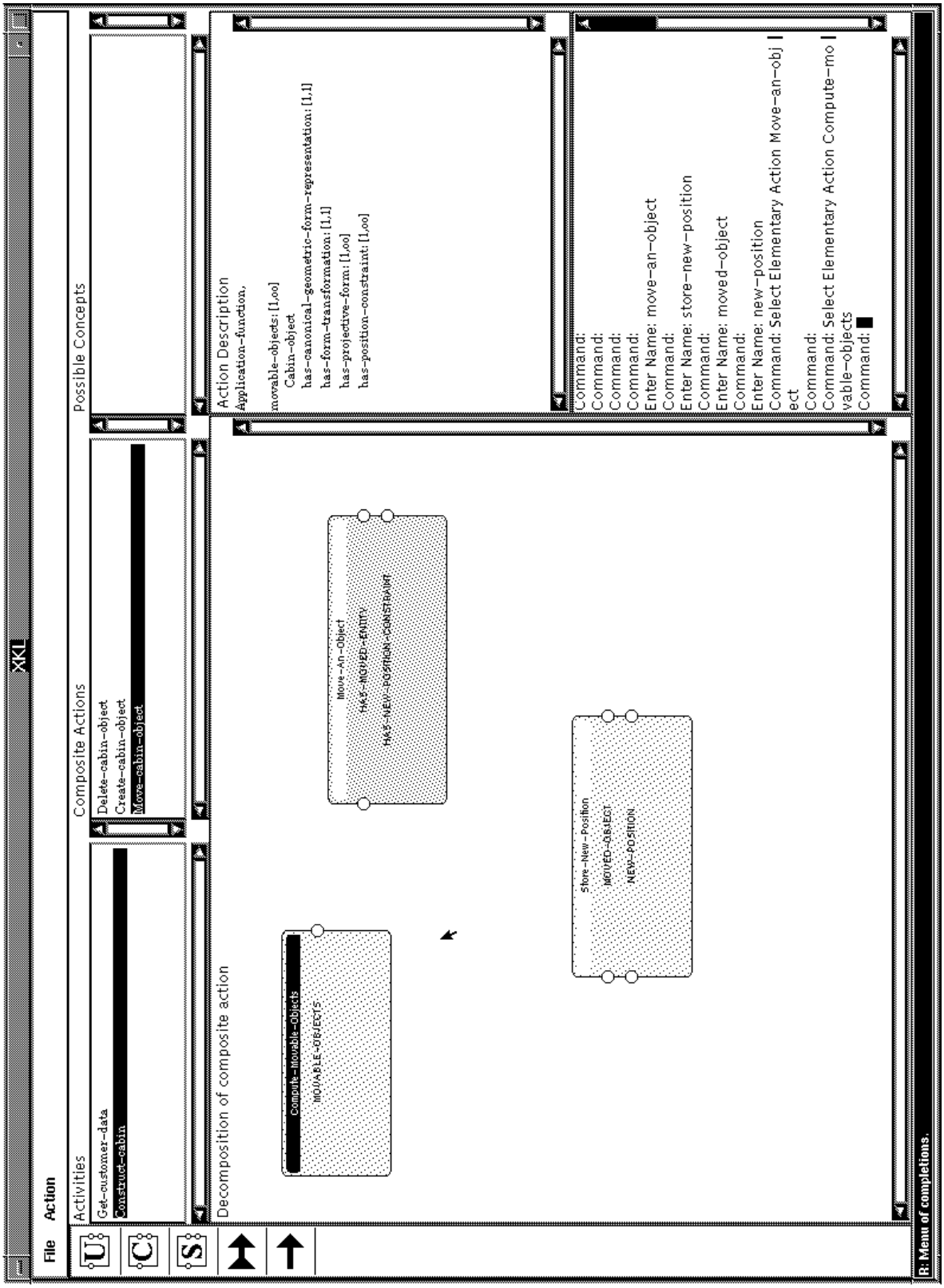


Abbildung 106. Definition der Aktion `move-cabin-object`.

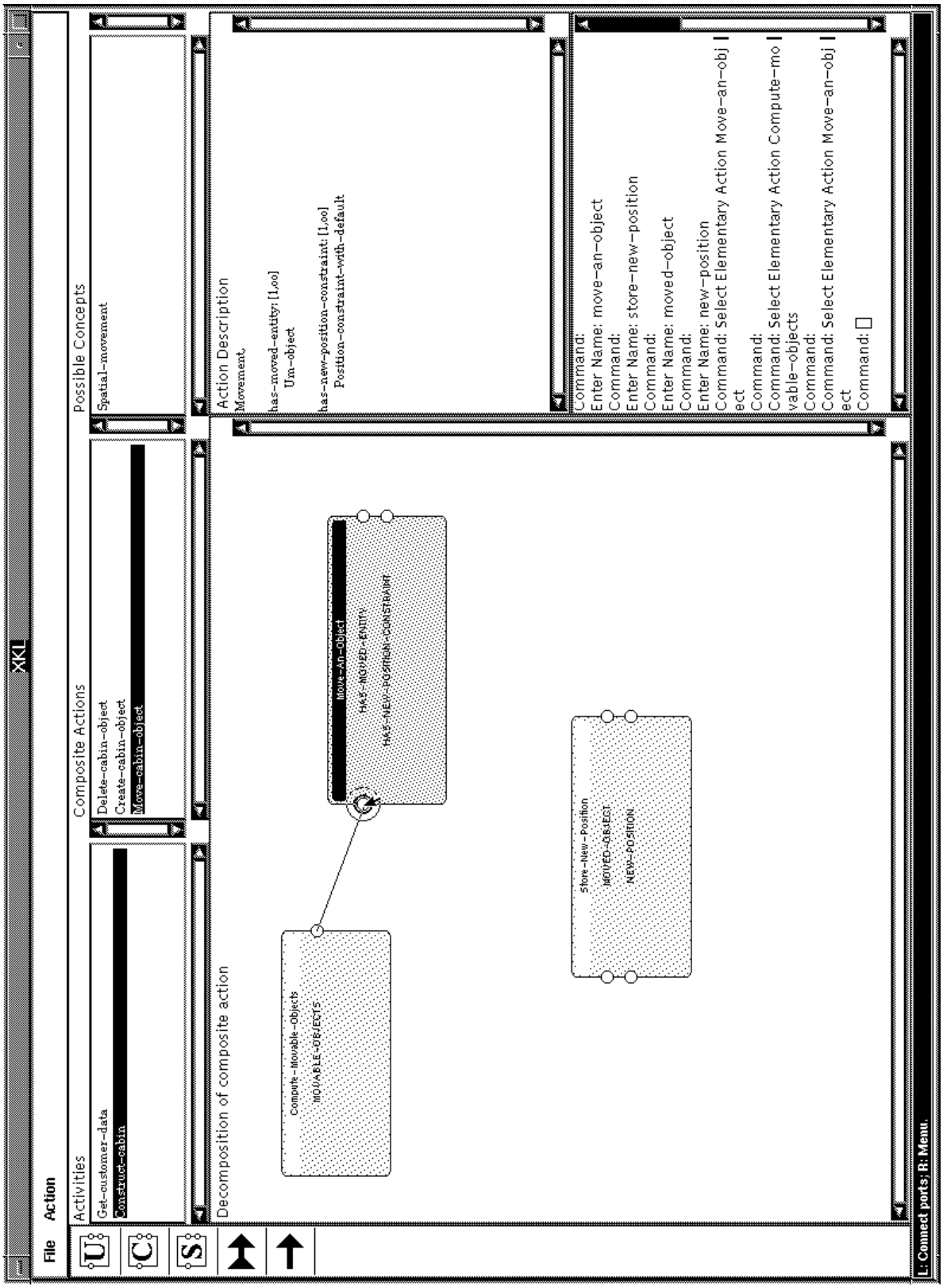


Abbildung 107. Verbindung von `compute-movable-objects` und `move-an-object` über eine Pipeline.

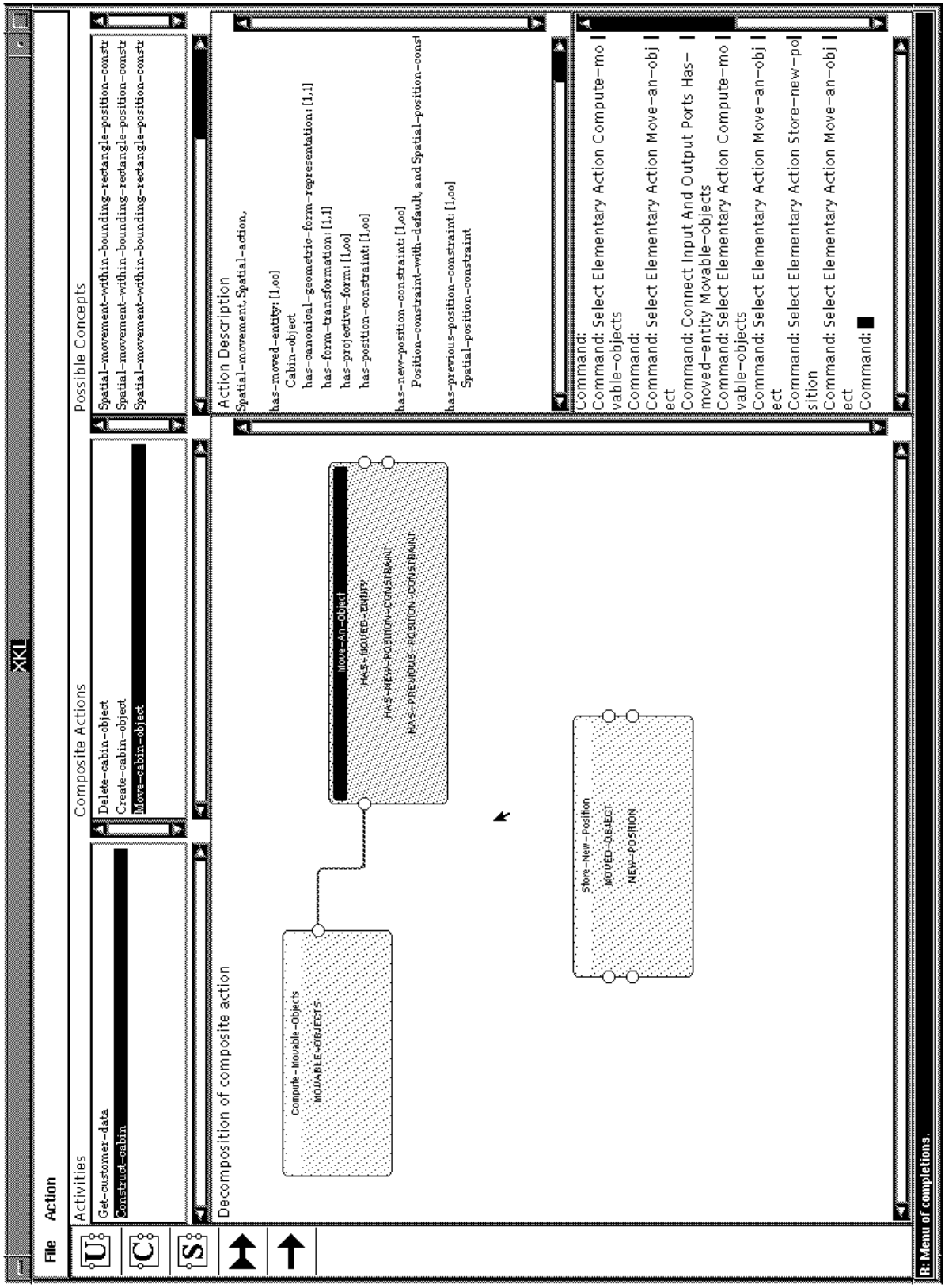


Abbildung 108. Automatische Spezialisierung von `move-an-object`.

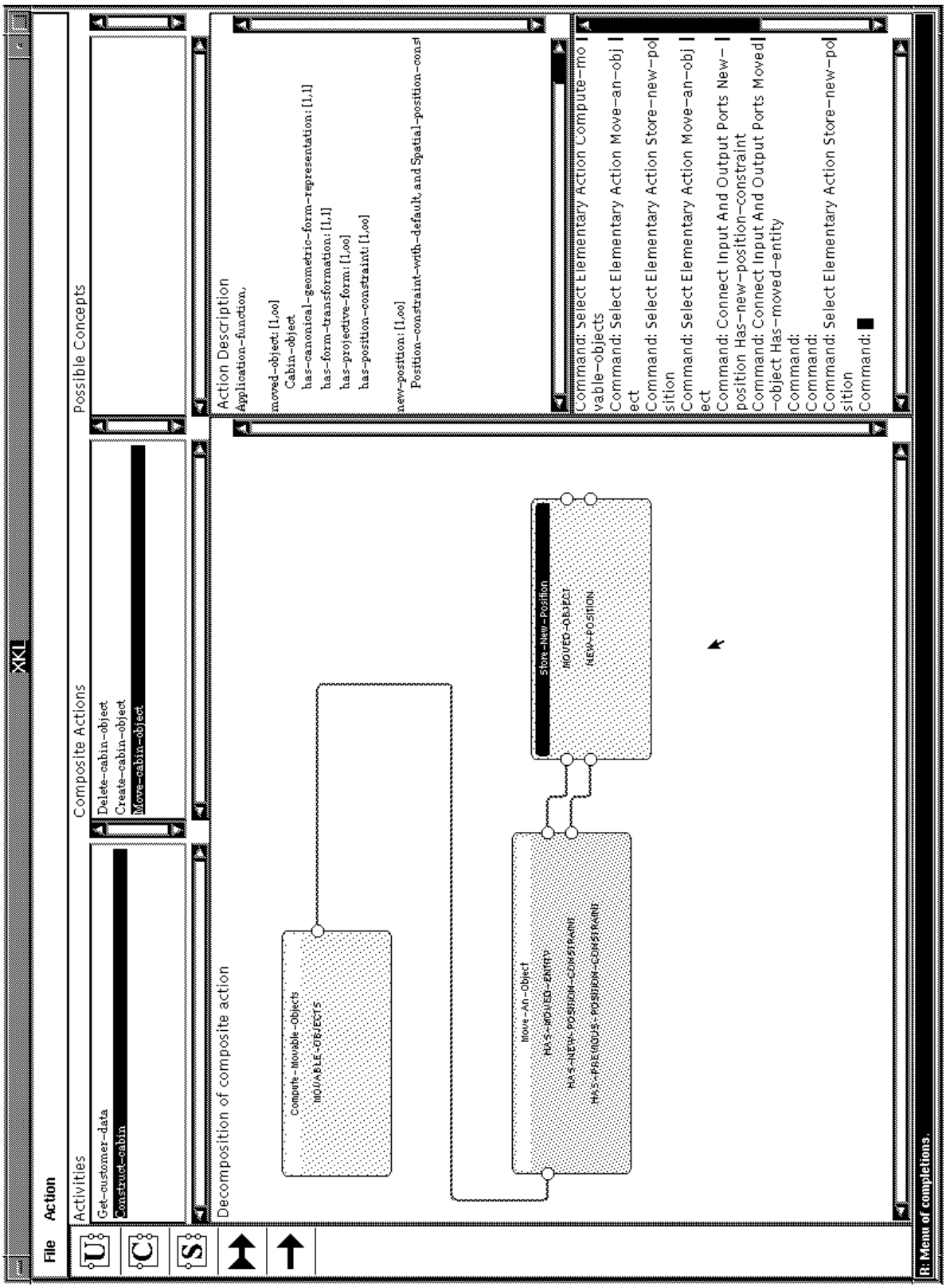


Abbildung 109. Automatisch inferierte Spezifikation von `store-new-position`.

Die Funktion `store-new-position` ist in Abbildung 109 durch Pipelines angebunden. Im Teilfenster `Action Description` steht die automatisch inferierte Typspezifikation der Funktion. Zur Illustration der Möglichkeiten der graphischen Oberfläche zur Aktionenmodellierung wurde die Benutzeraktion `move-an-object` an eine andere Stelle verschoben. Die Verbindungsstruktur durch Pipelines wird automatisch angepaßt.

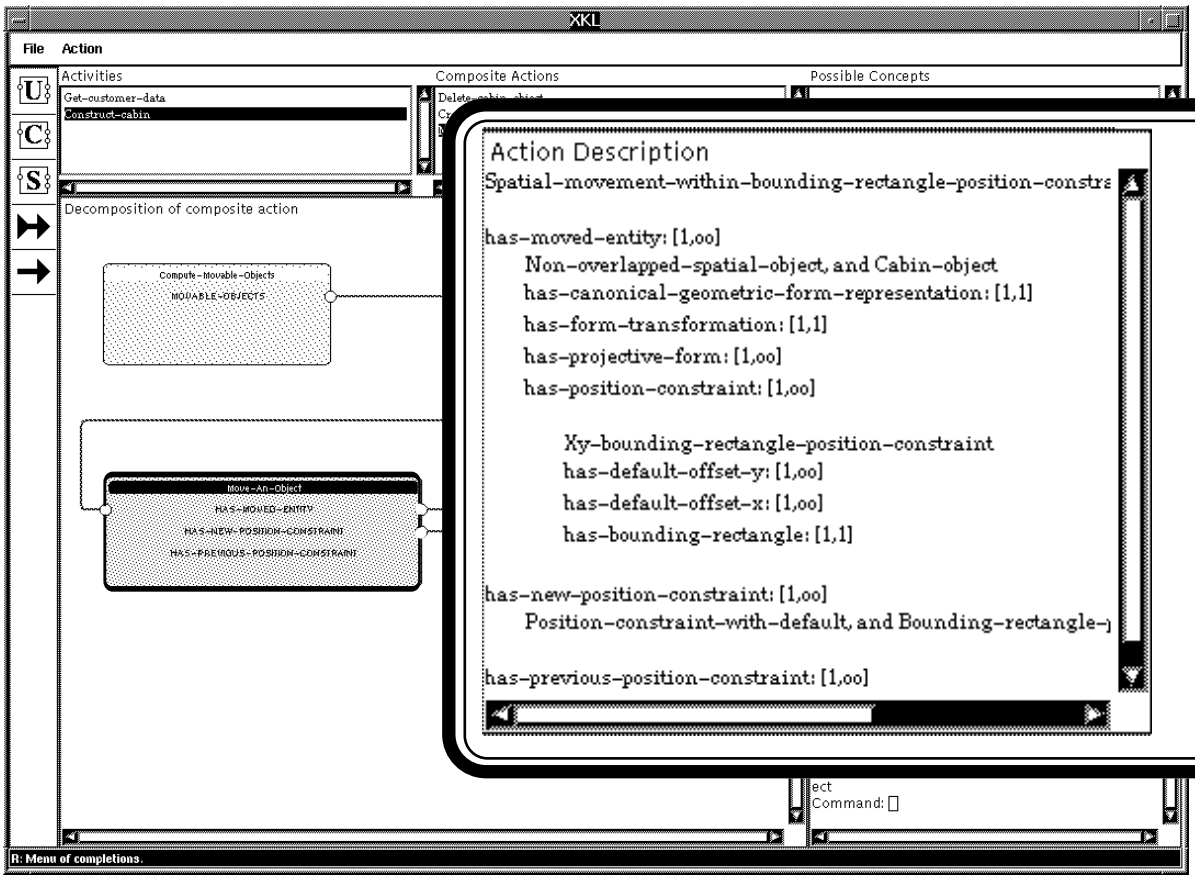


Abbildung 110. Spezialisierung von `move-an-object` zu `spatial-movement-within-bounding-rectangle-position-constraint.`

Auch `move-an-object` muß zur Entwicklungszeit soweit spezialisiert werden, daß genügend Informationen für die Oberflächengestaltung vorhanden sind. In Abbildung 110 wurde ein sehr spezielles Verschiebungskonzept gewählt. Mit der Lupe werden die daraus abgeleiteten Konzeptbeschreibungen hervorgehoben. Auch in diesem Beispiel wird `non-overlapped-spatial-object` für die manipulierten Objekte gefordert (Füllereinschränkungen von `has-moved-entity`).

In Abbildung 111 wird die Konsequenz der Spezialisierung gezeigt. Auch die Berechnungsfunktion `compute-movable-objects` muß Objekte von entsprechendem Konzepttyp liefern.

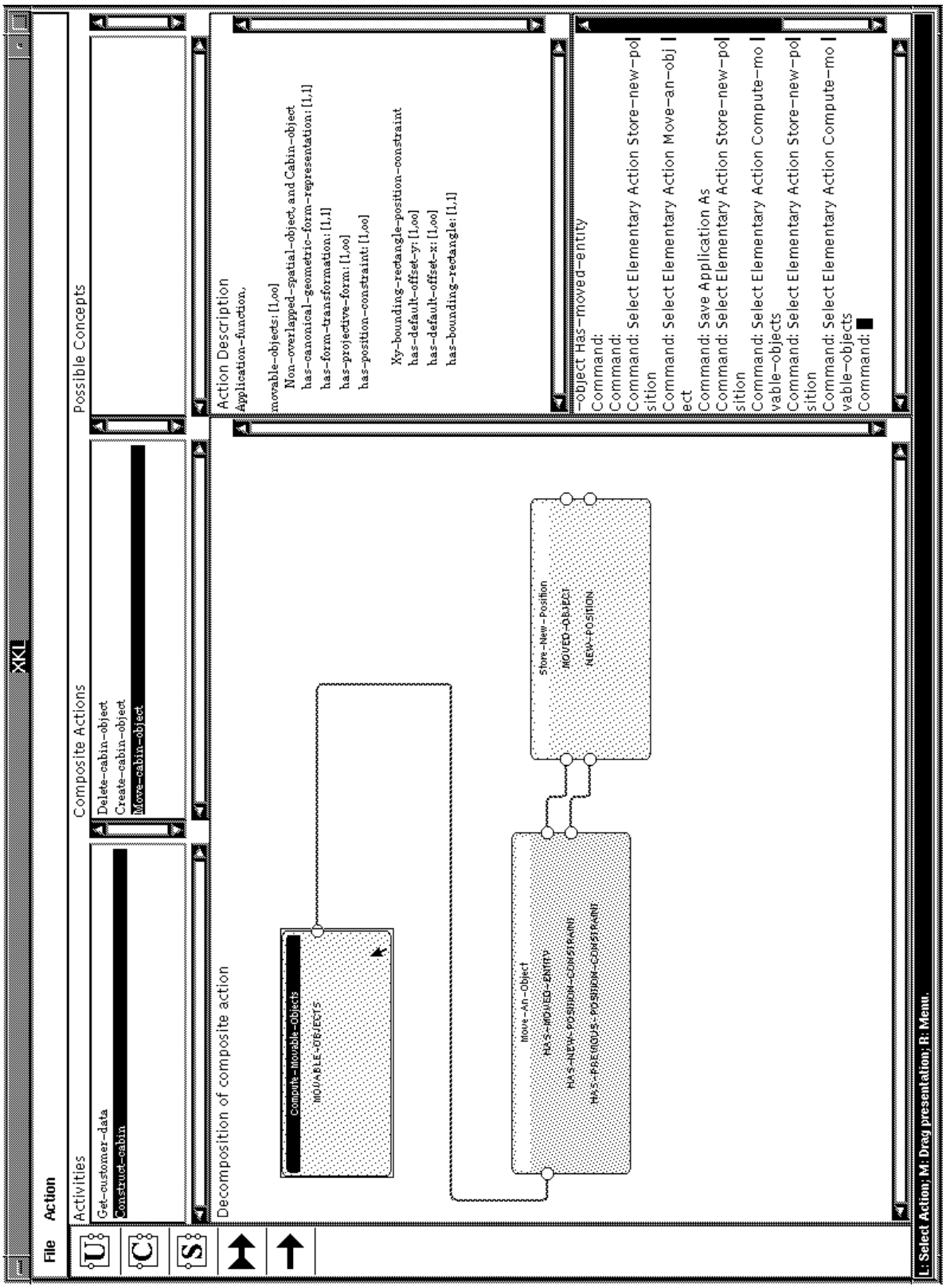


Abbildung 111. Beeinflussung der Spezifikation von Berechnungsfunktionen durch gewählte Benutzeraktionen.

Mit diesem Beispiel möchte ich die Illustration der Aktionenmodellierung mit HAMVIS beenden. Mit den vorgestellten Oberflächenkomponenten wurde verdeutlicht, daß es möglich ist, automatische Inferenzen in einem interaktiven System dem HAMVIS-Benutzer zu vermitteln.

3.2.14 Zusammenfassung

Während das HAMVIS-Grundmodell generisches Wissen über Aktionenkonzepte bereitstellt, muß die Komposition von Aktionen und Berechnungsfunktionen durch das Systementwicklungsteam als anwendungsspezifisches Wissen modelliert werden. Dieses Kapitel hat gezeigt, wie sich beschreibungslogische Wissensrepräsentationstechniken und Inferenzverfahren zur Aktionenmodellierung einsetzen lassen. Der Vorteil von Beschreibungslogiken bei der Repräsentation von unvollständigem Wissen durch Konzeptterme kommt im Kontext von HAMVIS besonders zur Wirkung. Obwohl konkrete Objekte als Füller von Kasusrollen von Aktionen zur Entwicklungszeit der Anwendung (i.a.) noch nicht bekannt sind, lassen sich doch durch beschreibungslogische Konzeptterme die Laufzeitobjekte so weit beschreiben, daß schon zur Entwicklungszeit einer Anwendung Schlußfolgerungsprozesse über zur Laufzeit anwendbare Aktionen realisiert werden können. Die in diesem Kapitel diskutierten Beispiele belegen, wie die zu betrachtenden Wechselwirkungen zwischen Wissen über Benutzeraktionen und Wissen über Berechnungsfunktionen formal modelliert werden können. Durch Subsumtionsinferenzen werden die bezüglich der zu manipulierenden Anwendungsobjekte möglichen Aktionen ermittelt. Spezialisierungen stehen innerhalb der interaktiven HAMVIS-Aktionenmodellierungsoberfläche zur Auswahl bereit. Durch definierte Konzepte mit notwendigen und hinreichenden Bedingungen kann HAMVIS (zur Entwicklungszeit) auch ohne Zutun des Systementwicklungsteams auf mögliche Handlungen schließen. Durch interaktive Auswahl (Spezialisierung) eines Aktionenkonzeptes werden die Typen der Berechnungs- und Speicherfunktionen der Anwendung ggf. eingeschränkt. Um Inkonsistenzen zu vermeiden und Aktionen mit Berechnungsfunktionen verbinden zu können, muß ggf. das Domänenmodell `xkl-model` entsprechend angepaßt werden. Es wird deutlich, daß das für die Anwendung zu erfassende Domänenwissen von der gewünschten Oberflächenrealisierung abhängt.

Die Aktionenmodellierung ist beendet, wenn die verwendeten elementaren Benutzeraktionen so weit spezialisiert sind, daß eine Abbildung auf UIMS-Dienste möglich ist (an der Oberfläche wird dieses durch graphische Anzeichen symbolisiert). Je spezieller ein Aktionenkonzept beschrieben wird, desto genauer kann die Darstellung auf die Erfordernisse der Anwendung abgestimmt werden. In anderen Worten: Dienste eines UIMS, die etwa in einer großen Softwarebibliothek für Interaktionsbausteine angeboten werden, lassen sich durch Aktionenkonzepte auf höherer Ebene zugängig machen, d.h. nicht jedes Standard-Interaktionselement muß dem Entwicklungsteam a priori bekannt sein. Notwendige Bausteine werden aufgrund der Charakterisierung der zu manipulierenden Anwendungsobjekte und eines gröberen Aktionenkonzeptes mit Hilfe von HAMVIS Schritt für Schritt hergeleitet.

Die Aktionenmodellierung definiert die grundlegende Interaktionsstruktur der Anwendung. Für jede Aktivität wird ein „Interaktionskontext“ aufgespannt. Bei den heute gebräuchlichen graphischen Systemen ist dieses üblicherweise ein Fenster. Innerhalb einer Aktivität stehen mehrere Handlungsalternativen (zusammengesetzte Aktionen) zur Auswahl bereit. Zusammengesetze Aktionen sind an der Oberfläche vorzugsweise durch Mausgesten umzusetzen (siehe Abbildung 1). Innerhalb des Interaktionszyklus zur Laufzeit werden direktmanipulative Interaktionsformen unterstützt, d.h. der Benutzer sagt nicht, daß z.B. ein Objekt verschoben werden soll und HAMVIS generiert erst dann die Graphik,

sondern der Benutzer klickt das zu verschiebende Objekt an und verwendet das dann erscheinende Interaktionselement (gadget) zur Verschiebung. Mit anderen Worten, die Auswahl einer Aktion aus den innerhalb einer Aktivität möglichen Alternativen (durch den Benutzer zur Laufzeit) erfolgt implizit z.B. durch die Mausgeste (siehe die Ausführung zu UIMS-Konzepten in Kapitel 2.1.4).

Mit dem Aktionenzerlegungsmodell einer Anwendung ist der Basisinhalt von Graphikfenstern für interaktive Visualisierungen festgelegt. Neben den Einschränkungen für die Darstellung der Anwendungsobjekte zur Laufzeit ist aber durch eine Aktionenzerlegung auch das zu generierende Laufzeitsystem beschrieben. Wir haben gesehen, daß das Laufzeitsystem abstrakt als Petri-Netz repräsentiert werden kann. Im Sinne eines Datenflußnetzes sind Berechnungs- und Speicherfunktionen immer dann anwendbar, wenn ihre Eingangsstellen gefüllt sind. Die Stellen enthalten zur Laufzeit jeweils Mengen von Objekten. Durch die Aktionenmodellierung ist das Konzept dieser Objekte vorgegeben. Im Petri-Netz sind Ausgangsstellen von Berechnungsfunktionen wiederum Eingangsstellen für Benutzeraktionen. Werden Stellen dieser Art gefüllt (jeweils mit einer Menge von Anwendungsobjekten), so wird die Graphikausgabe ggf. entsprechend aktualisiert. Danach sind Benutzeraktionen ausführbar. Sie liefern neue Anwendungsobjekte (oder leiten bestehende einfach weiter), die wiederum in den Ausgangsstellen im Laufzeitnetz abgelegt werden usw.

In Abbildung 112 wird das resultierende Laufzeitnetz für die XKL-Anwendung innerhalb der HAMVIS-Benutzungsschnittstelle gezeigt.

Es wird in Abbildung 112 im Vergleich zur schematischen Darstellung des Laufzeit-Petri-Netzes in Abbildung 104 auch das Netz für die zusammengesetzte Aktion `move-cabin-object` gezeigt. Das von HAMVIS ermittelte Laufzeitnetz enthält weiterhin noch Kardinalitätsangaben für die Stellen (bzw. die Objektmengen, siehe die Darstellung in Abbildung 112). Links an einer Stelle ist die Minimalkardinalität, rechts die Maximalkardinalität angegeben (eine fehlende Angabe steht für „unbegrenzt“). Die Stellenkardinalitäten leiten sich aus den Kasusrollenrestriktionen der Aktionen im Aktionenzerlegungsmodell ab.[99] Im Gegensatz zu Ansätzen aus der Mensch-Computer-Interaktion, die in Kapitel 2.2.4 erläutert wurden, braucht das Petri-Netz zur Beschreibung der Dialogmodellierung nicht extra spezifiziert zu werden, sondern es ergibt sich automatisch aus der Aktionenzerlegung.

99. Wenn durch die Startfunktion einer zusammengesetzten Aktion (hier `compute-movable-objects`) die geforderte Minimalanzahl von Objekten nicht geliefert wird, so ist die betreffende zusammengesetzte Aktion nicht durchführbar. Durch das Laufzeitsystem müssen entsprechende Anzeichen (z.B. graue Menüeinträge) präsentiert werden.

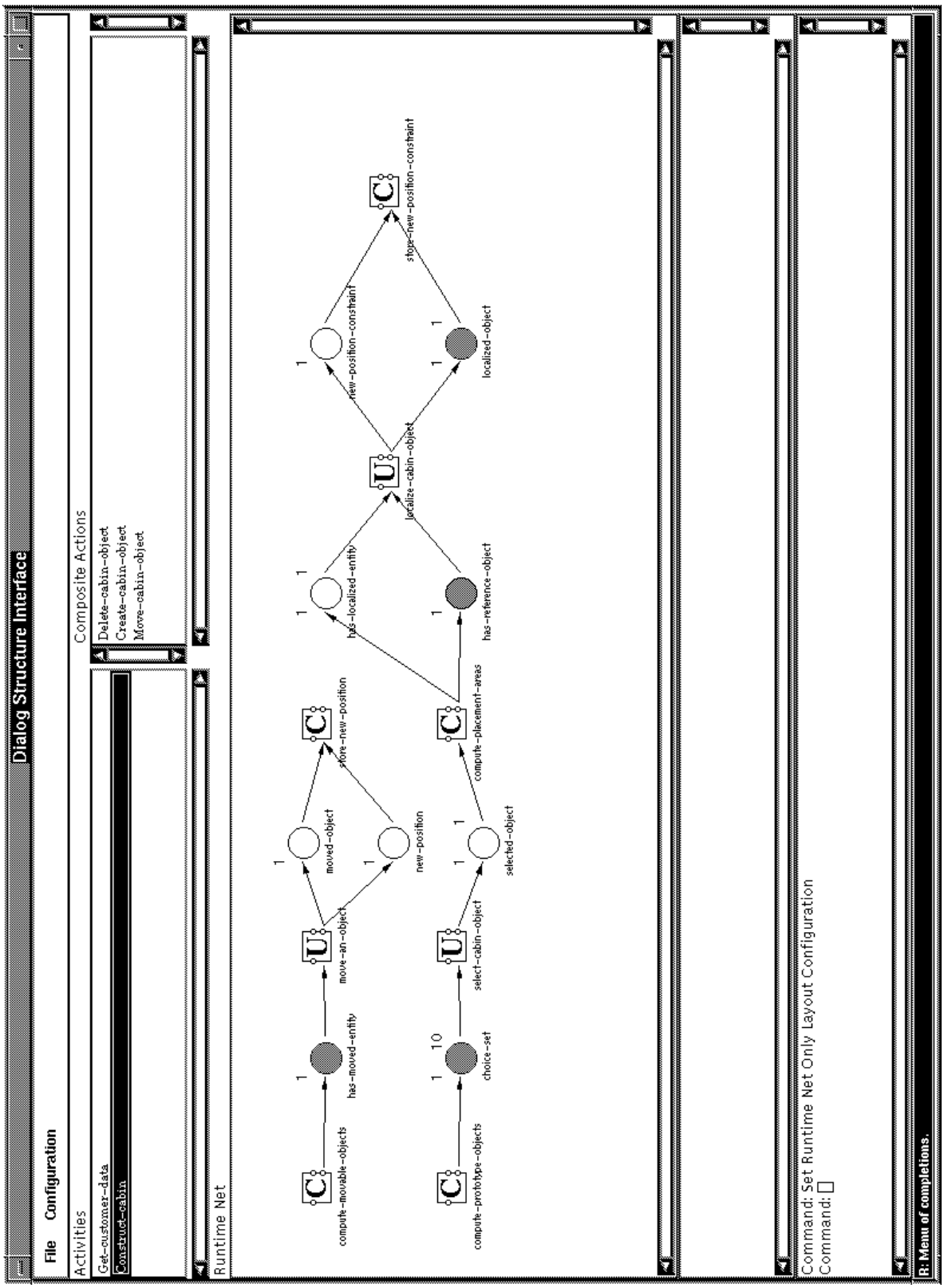


Abbildung 112. Laufzeit-Petri-Netz für die XKL-Applikation. Stellen mit zu präsentierenden Objekten sind grau dargestellt.

Es konnte gezeigt werden, daß mit der hier vorgestellten Form der wissensbasierten Aktionenmodellierung zumindest Teilaspekte der im Bereich Mensch-Maschine-Interaktion vorgeschlagenen Entwurfszeitmodelle für interaktive Anwendungen formalisiert und damit der automatischen Weiterverarbeitung zugeführt werden können (siehe die Ausführungen zum Designmodell in Kapitel 2.2). Nicht zu vernachlässigen ist auch, daß die expliziten Modelle zur Dokumentation der Anwendungsentwicklung dienen. Auch Aspekte eines Implementationsmodells einer speziellen Anwendung werden durch die Beschreibungen der Parameter und Werte von Berechnungs- und Speicheraktionen im Petri-Netz formal erfaßt.[100]

Die Gestaltung von Visualisierungen innerhalb der Gesamtoberfläche ist jedoch durch die Aktionenmodellierung noch nicht vollständig bestimmt. Insbesondere bei der Betrachtung der Lokalisierungshandlung wurde deutlich, daß zur Durchführung der Aktion die zuvor berechneten Plazierungsbereiche in einem Graphikfenster nicht ohne Kontext dargestellt werden können. Es wird noch ein Referenzsystem zur Orientierung benötigt usw. Als Referenzsystem kann das Objekt, das die Plazierungsbereiche räumlich enthält, verwendet werden. Es ist weiterhin notwendig, zur Entwicklungszeit das Referenzsystem nicht erst dann zu ermitteln (und darzustellen), wenn ein Plazierungsbereich als Objekt vorhanden ist. Das Referenzsystem für den Aktionsraum muß schon vorher dargestellt werden, damit eine „benutzerfreundliche“ Interaktionsform entsteht. Durch HAMVIS muß dieses hergeleitet werden können, bzw. HAMVIS muß ein Domänenmodell fordern, aus dem die für die Oberfläche notwendigen Informationen hervorgehen. Sobald die Darstellungssicht von Einzelobjekten festgelegt wurde, können für zusammengesetzte Handlungen auch entsprechende Interaktionsgesten, wie z.B. Ziehen-und-Fallenlassen-Techniken, bestimmt werden (sofern hier durch die Aktionenmodellierung noch Wahlmöglichkeiten gegeben sind). Weiterhin kann die Abfolge von Aktivitäten innerhalb einer Applikation bestimmt werden. Diese Gesichtspunkte werden im nächsten Abschnitt unter dem Aspekt der Dialogstrukturierung detaillierter besprochen.

3.3 Dialogstrukturierung

Anwendungen in der von HAMVIS betrachteten Aufgabenklasse können kaum durch einen rein Werkbank-orientierten Gestaltungsansatz realisiert werden, bei dem der Anwender die bearbeiteten Objekte ausschließlich selbst bestimmt. Die für eine Anwendung durch das Systementwicklungsteam erstellte Aufgabenzerlegung legt fest, „was mindestens“ zur Laufzeit „wann“ darzustellen ist. Die mit der Aufgabenzerlegung gegebenen bzw. abgeleiteten Basisinformationen müssen noch aufbereitet, ergänzt und strukturiert werden, so daß ein vollständiges Laufzeitsystem generiert werden kann. Es ist durch die Aktionenzerlegung noch nicht direkt definiert, „wie“ und „wo“ die Darstellung tatsächlich erfolgen soll (vgl. die Ausführungen zur Gestaltung von IMMP-Systemen in Abbildung 2.4). Für die konkrete Ausgestaltung von Darstellungen und für den Ablauf der Interaktion zur Laufzeit sind durch die Aktionenkonzepte des HAMVIS-Grundmodells lediglich Einschränkungen festgelegt.

Graphische Objekte dürfen nicht beliebig auf dem Bildschirm erscheinen. Darstellungsfolgen innerhalb des Interaktionszyklus unterliegen gewissen „Regularitäten“, die bedingen, daß aus Einzeldarstellungen und Benutzeraktionen eine kohärente Interaktionsform entsteht. Für eine methodische

100. Falls zur Entwicklungszeit nicht bewiesen werden kann, daß die Berechnungsfunktionen die erwarteten Werte liefern, so kann doch zumindest im zu generierenden Laufzeitsystem eine entsprechende Überprüfung erfolgen.

Unterstützung der Entwicklung von Anwendungen muß Wissen über die Gestaltung von graphischen Darstellungen und die Umsetzung von graphischen Kommunikationstechniken zum Einsatz kommen. HAMVIS stellt für die Dialogstrukturierung deklarative Repräsentationsformen für Aspekte dieses Darstellungswissens bereit. Dabei wird nicht von vollständigen, „eingefrorenen“ Domänenmodellen ausgegangen. Schon bei der Aktionenmodellierung wurden in dieser Arbeit die Einflüsse der direktmanipulativen Interaktion auf die Repräsentation von Domänenmodellen und Berechnungsfunktionen aufgezeigt. Um gewünschte Interaktionsformen zu unterstützen, müssen z.B. bestimmte Anwendungsobjekte (zur Laufzeit) berechnet werden können, *bevor* die Aktion mittels einer Geste tatsächlich durchgeführt wird. Für eine Verschiebung muß beispielsweise der mögliche Verschiebungsbereich a priori, also vor der eigentlichen Verschiebehandlung, bestimmt werden. Im vorigen Abschnitt wurde ausführlich geschildert, wie beschreibungslogische Inferenzen ausgenutzt werden können, um dieses zur Entwicklungszeit sicherzustellen. Wir werden in diesem Kapitel sehen, daß sich auch aus der Strukturierung der Darstellung und der Verfeinerung von Darstellungen weitere Anforderungen an die anwendungsspezifischen Domänenmodelle ergeben.

Für die Entwicklung von graphischen Benutzungsschnittstellen und Präsentationen wurden verschiedenartige Sichtweisen entwickelt. Das Kapitel 2 spannte ein Spektrum der relevanten Arbeiten auf. Ergebnisse der KI-Forschung im Bereich der graphischen Kommunikation haben bislang noch kaum Eingang in Arbeiten gefunden, die die Entwicklung von Oberflächen mit graphischen Darstellungen aus der Software-Engineering-Perspektive betrachten. Die Forschungsergebnisse von adaptiven IMMPS-Architekturen belegen allerdings die Bedeutung der Kommunikationssicht und zeigen auf, welches Potential in einer kommunikationsorientierten Sicht auf die Gestaltung von graphischen Präsentationen steckt. Im Kontext dieser Arbeiten werden z.B. die oben angesprochenen „Regularitäten“ über den Begriff der „Kohärenz“ detaillierter formuliert und genauer analysiert (siehe Kapitel 2.4). Obwohl es vielleicht dem Endanwender der zu erstellenden Applikation nicht bewußt zu sein braucht, erscheint es mir vorteilhaft, zur Entwicklungszeit die Vervollständigung und Strukturierung der durch die Aktionenzerlegung definierten Inhalte aus einer Kommunikationssicht heraus durchzuführen. Ich spreche daher bewußt von einer Phase der Dialogstrukturierung.

Mit HAMVIS möchte ich aufzeigen, daß eine wissensbasierte Kommunikationssicht auch in einem Anwendungsszenario wie bei HAMVIS, wo also im Gegensatz zu IMMP-Systemen konkrete Objekte noch nicht vorliegen, wesentlich zur Strukturierung der vielfältigen Entwurfsentscheidungen beitragen kann. Im Gegensatz zu vielen IMMPS-Ansätzen mit agentenorientierter Architektur sind im Kontext von HAMVIS allerdings keine „globalen“ Präsentationsziele wie z.B. `Explain-Route` vorgegeben (vgl. Abbildung 44), die dann in einem Top-Down-Zerlegungsverfahren in kleinere Basiseinheiten aufgebrochen werden können. Stattdessen ergibt sich die Eingabe an die Dialogstrukturierungskomponente aus einem Aktionenmodell bestehend aus einer Aktionenzerlegung und einem Laufzeit-Petri-Netz sowie den jeweils assoziierten Informationen. Basiseinheiten und Darstellungseinschränkungen, die sich aus den Aktionenkonzepten ergeben, müssen in einem Bottom-Up-Prozeß zu einem Ganzen verbunden werden, so daß die Aktionen zur Laufzeit adäquat durchgeführt werden können. Da HAMVIS zur Entwicklungszeit verwendet werden soll, sind die zur Laufzeit auftretenden konkrete Folgen von Einzelhandlungen innerhalb einer Aktivität nicht bekannt, so daß die festzulegende Dialogstruktur für alle möglichen Aktionenfolgen geeignet sein muß.

Wie schon bei der Präsentation der Aktionsmodellierung für eine Anwendung wird bei der Präsentation in den nächsten Abschnitten eine interne Sicht und eine externe Sicht präsentiert. Die *interne*

Sicht beschreibt die formale Modellierung der Wissensbasen zur Dialogstrukturierung. Mit der *externen Sicht* wird anhand der Benutzungsschnittstelle gezeigt, welche Ableitungen mit den internen Wissensbasen möglich sind. Gleichzeitig wird gezeigt, wie die von HAMVIS bereitgestellten Dienste einem Schnittstellenentwickler zur Verfügung gestellt werden. Auf die konkrete Funktionalität der HAMVIS-Oberflächen wird aber nur teilweise eingegangen, da ich den Schwerpunkt dieser Arbeit auf die interne Sicht der Wissensrepräsentation legen möchte.

3.3.1 Aufgaben der Dialogstrukturierungskomponente

Das Ziel von IMMP-Systemen ist eine adaptive, situationsspezifische Interaktionsgestaltung. Es gibt viele Einsatzbereiche, wo dieses sinnvoll und notwendig sein kann: z.B. interaktive Reparaturanleitungen (siehe die Ausführungen in [6]), u.U. auch in Verbindung mit einem WWW-System. In der von XKL vertretenen Anwendungsklasse ist es m.E. jedoch notwendig, für jede Aktivität einer Anwendung zur Entwicklungszeit eine feste Darstellungs- und Interaktionsform zu finden, so daß z.B. bei der Routinetätigkeit der Auswahl von Einrichtungsgegenständen beim Kunden nicht „plötzlich“ Darstellungen generiert werden, die für den Verkäufer des Flugzeugherstellers ungewohnt in der Handhabung sind. Vergleichbares gilt für die Gestaltung von Visualisierungen für Leitwarten. In diesem Kontext muß ggf. sichergestellt sein, daß die Operateure im Umgang mit den Oberflächenelementen auch entsprechend geschult werden können.

Es lassen sich also die folgenden Tätigkeiten umreißen, die im HAMVIS-Kontext von oder mit einer Dialogstrukturierungskomponente durchgeführt werden müssen:

- Ergänzung des minimalen Informationsinhalts der Basiseinheiten (jeweils gegeben durch eine Stelle im Laufzeitnetz), so daß eine adäquate Informationsverarbeitung des Benutzers ermöglicht werden kann (z.B. Orientierung, Situationseinschätzung),
- Festlegung der Zuordnung von Objekten, die zur Laufzeit in Stellen des Laufzeitsystems auftreten, zu ihren Darstellungsfenstern,
- Festlegung des Darstellungszeitpunkts von ergänzten Informationsinhalten,
- Definition der „Sicht“ auf die Objekte der Anwendung (Modellierungssicht und geometrische Sicht),
- Bestimmung einer festen Menge von Teilfenstern (panes) für eine Aktivität (entweder Graphikfenster oder Fenster für Standard-Interaktionsbausteine wie Werkzeugkästen, Auswahlpaletten usw.),
- Komposition von Teildarstellungen unter Berücksichtigung von Einschränkungen bezüglich der Sichten, so daß Graphiken mehrere Aktionen unterstützen können und möglichst wenige Teilfenster benötigt werden,
- Abbildung von zusammengesetzten Handlungen auf Interaktionsgesten zur Realisierung von direktmanipulativen Interaktionsformen (z.B. Ziehen-und-Fallenlassen),
- Festlegung der Reihenfolge von Aktivitäten bzw. Aktivierungsbedingungen für Aktivitäten sowie die Definition des (oder der) Layouts der Teilfenster einer Aktivität.

Bei der Ergänzung der Basisinformationsinhalte können obligatorische und optionale Ergänzungen auftreten. Die Aufgabe der Dialogstrukturierungskomponente von HAMVIS besteht darin, Lösungs-

möglichkeiten zu erkennen und dem HAMVIS-Benutzer als „Varianten“ der Dialoggestaltung zu präsentieren. Innerhalb der HAMVIS-Oberfläche muß der Entwickler entsprechende Entscheidungen fällen können, wobei mehrere „Entwürfe“ verwaltet werden müssen. Die hier aufgeführten Aufgaben sind allerdings nicht unbedingt unabhängig voneinander zu lösen.

3.3.2 Aufgaben einer Dialogstruktur im Kontext von HAMVIS

Das aus der Dialogstrukturierungsphase abzuliefernde Produkt, also die Dialogstruktur, enthält explizite Beschreibungen der Rolle eines Objektes (bzw. einer Menge von Objekten) im Rahmen der Gesamtdarstellung. Ähnlich wie bei IMMP-Systemen läßt sich Objekten eine kommunikative Funktion zuordnen, die davon abhängt, in welcher Stelle im Laufzeitnetz die Objekte auftreten. Es ergibt sich eine rhetorische Struktur, in der die Objektmengen in bestimmten rhetorischen Beziehungen zueinander stehen und innerhalb der Gesamtdarstellung wiederum bestimmte Aufgaben erfüllen. Beziehungen sind nicht nur zwischen graphischen Objekten innerhalb eines Fensters bedeutsam. In dieser Arbeit wurden vielfältige Beispiele aufgezeigt, an denen deutlich wurde, daß auch auf der Ebene der Teilfenster Beziehungen und Einflüsse expliziert werden müssen. Rhetorische Beziehungen leiten sich z.B. auch aus der Realisierung von zusammengesetzten Handlungen durch Interaktionsgesten her und wirken sich z.B. auf die Bestimmung des Layouts der Teilfenster aus. Die Realisierung der zusammengesetzten Handlung `create-cabin-object` in Abbildung 1 durch eine Ziehen-und-Fallenlassen-Geste legt beispielsweise die Nebeneinanderplazierung der jeweiligen Teilfenster nahe.

Im Rahmen der Dialogstruktur wird die kommunikative Funktion von „Konstituenten“ hergeleitet und repräsentiert. Damit sind wiederum Aspekte des Designmodells (siehe Kapitel 2.2.1) einer Anwendung explizit gemacht. Im Rahmen der weiteren Gestaltung von Visualisierungen werden diese Informationen z.B. von der nachfolgenden Präsentationsstrukturierungskomponente verwendet, um dem Schnittstellenentwickler geeignete Zeichenattribute für die tatsächliche Objektdarstellung vorzuschlagen (siehe die Übersicht in Abbildung 52). Durch die Dialogstruktur werden Einschränkungen für die Bestimmung von Zeichenattributen festgelegt. Die Dialogstruktur hat auch die Funktion, spezielle Entwurfsentscheidungen von nachfolgenden Schritten zu legitimieren. So können z.B. in der Präsentationsgenerierungsphase optionale, vom rhetorischen Standpunkt weniger bedeutsame Objekte (oder Objektmengen) sehr abgeschwächt präsentiert oder auch ganz weggelassen werden, wenn sich aus Gründen der Übersichtlichkeit in einem Teilfenster nicht alles darstellen läßt.

Es wird also deutlich, daß die in der Dialogstruktur festgelegten Informationen von nachfolgenden HAMVIS-Komponenten für die Generierung des endgültigen Laufzeitsystems einer Anwendung verwendet werden (siehe das Phasenmodell in Abbildung 52). Nach dieser kurzen Motivation der expliziten Dialogmodellierung möchte ich wieder auf die Aufgaben der Dialogstrukturierung zu sprechen kommen.

Erweiterung des Laufzeitnetzes

Das Modell der Aktionenzerlegung definiert den Basisinhalt von Visualisierungen. Zur Unterstützung des Endbenutzers und zur Bestimmung einer kohärenten Gesamtdarstellung müssen noch weitere Objekte hinzukommen. Ich möchte dieses an einem Beispiel erläutern.

Nehmen wir an, für die Verschiebungsaktion `move-an-object` aus Abbildung 112 wird durch die Dialogstrukturierungskomponente ermittelt, daß zu den verschiebbaren Objekten in der Stelle `has-`

`moved-entity` noch die Objekte der Umgebung zu bestimmen und darzustellen sind. Dieses führt zu einer Erweiterung des Laufzeitnetzes. In Abbildung 113 ist eine Skizze eines solchen erweiterten Laufzeitnetzes dargestellt.

Die in der Stelle zwischen `compute-movable-objects` in `move-an-object` zur Laufzeit vorliegenden Objekte werden als Eingaben für die Funktion `compute-environment-objects` verwendet. Durch diese Funktion, die durch HAMVIS bereitgestellt wird und auf das HAMVIS Grundmodell und die anwendungsspezifischen Domänenmodelle Bezug nimmt, werden die Objekte der Umgebung der Eingangsobjekte bestimmt. Die berechneten Objekte müssen in der gleichen Darstellung wie die verschiebbaren Objekte gezeigt werden.

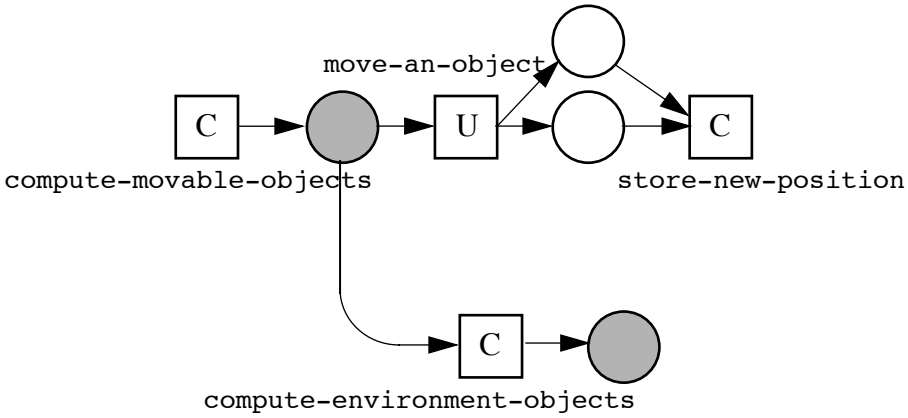


Abbildung 113. Beispiel für die Erweiterung des Laufzeitnetzes für die zusammengesetzte Aktion `move-an-object`.

Ein anderes Beispiel betrifft die Gestaltung der Gesamtdarstellung. Für Aktionen mit räumlichen Objekten wird z.B. ein Referenzsystem benötigt, das anzeigt, in welchem Raum die Aktionen durchgeführt werden. Dieses ist z.B. für die Lokalisierungsaktion `localize-cabin-object` der Fall. In ähnlicher Weise wie in Abbildung 113 wird für bestimmte Eingangsstellen von räumlichen Aktionen (vgl. die Abbildung 112) von HAMVIS eine Erweiterung des Laufzeitnetzes vorgenommen. Genaueres hierzu wird in den nachfolgenden Abschnitten geschildert.

Ausgabe von Objekten im Interaktionszyklus

Neben den zur Entwicklungszeit möglichen Entscheidungen bezüglich des Inhalts von Visualisierungen sind zur Festlegung der konkreten Gestaltung noch einige Aspekte des im Interaktionszyklus zur Laufzeit tatsächlich auftretenden Diskurses zu berücksichtigen. Ich möchte zwischen einem allgemeinen Dialogstrukturmodell, das zur Entwicklungszeit verwendet wird, und einem konkreten Diskursmodell unterscheiden, in dem die konkrete Historie von Ausgaben und Eingaben vermerkt werden kann und das zur Laufzeit der Anwendung erzeugt wird. Die Verwaltung von Ausgaben erfolgt durch einen *Ausgabenverwalter* (display manager), der eine Teilkomponente des zu generierenden Laufzeitsystems ist. Das Laufzeitnetz einer Anwendung wird innerhalb des Interaktionszyklus der Aktivität nach jeder zusammengesetzten Aktion neu evaluiert. Ausgaben werden im Interaktionszyklus durch Füllung bestimmter Stellen im Laufzeitnetz initiiert. Jedes zu zeichnende Objekt wird an den Ausgabenverwalter übermittelt. Der Ausgabenverwalter vermerkt jedes bei einem Durchgang durch den Interaktionszyklus gezeichnete Objekt. Ein Beispiel sind die verschiebbaren Objekte aus

Abbildung 113, die durch eine Berechnungsfunktion ermittelt werden. Es kann jetzt der Fall sein, daß z.B. aufgrund einer Verschiebung eines speziellen Objektes zu einem nachfolgenden Zeitpunkt weniger Objekte verschiebbar sind, d.h. beim nächsten Durchgang durch den Interaktionszyklus werden weniger Objekte in der Stelle `has-moved-entity` zu finden sein. Damit ändert sich eventuell auch die Menge der Umgebungsobjekte. Wenn nun beim erneuten Eintritt in den Interaktionszyklus weniger Objekte in der Ausgangsstelle von `compute-environment-objects` auftreten, so werden bei einem einfachen Vorgehen Objekte bei der Darstellungsverwaltung gelöscht, die vorher noch auf dem Bildschirm gezeigt worden sind. Nach einer weiteren Verschiebung eines Kabinenobjektes können auch andere Objekte vielleicht wieder verschiebbar sein, so daß dann wieder Umgebungsobjekte dargestellt werden, die vorher gerade gelöscht wurden usw. Ähnliches gilt für die Menge der verschiebbaren Objekte selbst. Wenn ein Objekt in einer bestimmten Interaktionssituation einmal nicht verschiebbar ist, sollte es nicht gleich gelöscht, sondern zeitweilig nur abgeschwächt präsentiert werden.

Eine ständige, relativ unmotivierte Änderung der Darstellung muß vermieden werden. Neben dem Laufzeitnetz muß also der Ausgabenverwalter des Laufzeitsystems noch eine Diskursverwaltungskomponente für die Darstellung führen. Je nach kommunikativer Funktion und rhetorischer Stellung von Objekten im Diskurs können dann zur Darstellung entsprechende Zeichenattribute verwendet werden. Innerhalb der Anwenderschnittstelle können auch entsprechende Meta-Interaktionen (also Festlegungen über die Art der Kommunikation selbst) zur Laufzeit erfolgen, so daß z.B. auf Wunsch des Anwenders einige Objekte, die nur noch aufgrund von vorherigen Ausgaben zusätzlich gezeigt werden, ausgeblendet werden können usw. Falls ein solcher Baustein für eine bestimmte Dialogstruktur benötigt wird, müssen entsprechende Interaktionswerkzeuge im Laufzeitsystem vorgesehen werden. Hierzu sind auf UIMS-Ebene entsprechende Abstraktionen vorzusehen, so daß die vorgeschlagenen Möglichkeiten in die entsprechenden Wirtssysteme eingebettet werden können. Ein anderes Beispiel für benutzergesteuerte Darstellungsaspekte zur Laufzeit ist die Einteilung von präsentierten Objekten in bestimmte Gruppen oder auch Schichten, die von Benutzer der Anwendung zur Laufzeit ebenfalls aus- oder eingeblendet werden können.

Die Aufgabe der Dialogstrukturierungskomponente besteht darin, zur Entwicklungszeit für die Stellen im Laufzeitnetz festzulegen, welche Art von Diskursverwaltung für bestimmte Objekttypen (aus dem HAMVIS-Grundmodell) notwendig ist, bzw. welche Objektmengen zur Laufzeit wie und zu welchem Zeitpunkt an die entsprechenden Diskursverwaltungskomponenten übermittelt werden. Zur Bestimmung von Zeichenattributen muß insbesondere der *Diskurszweck* der Objekte einer Stelle im Petri-Netz repräsentiert werden. Weiterhin werden für den Ausgabenverwalter für jede Stelle Informationen über die Darstellungsart (genannt *Diskursposition*) benötigt. Einige Objekte werden „statisch“ dargestellt (Referenzsystem), andere Objekte nur innerhalb eines Durchgangs durch den Interaktionszyklus („dynamische“ Diskursposition), wiederum andere Objekte werden immer wieder gezeichnet, wenn sie einmal präsentiert worden sind und nicht aus dem Domänenmodell gelöscht wurden („dynamisch-persistent“).

Für die Herleitung dieser Darstellungsinformationen (zur Entwicklungszeit) muß Wissen zur Modellierung von Dialogstrukturen bereitgestellt werden. Bevor die konkrete Formalisierung dieses Wissens in HAMVIS erläutert wird, möchte ich zunächst einmal auf der Wissensebene den Einsatz des Wissens betrachten und die gewählte Art und Weise der Wissensmodellierung motivieren.

3.3.3 Wissen zur Modellierung von Dialogstrukturen in HAMVIS

Am Beispiel von APEX wurde in Kapitel 2.3.3 diskutiert, aus welchen Gesichtspunkten heraus bestimmte Objekte in eine Darstellung aufgenommen werden können: manipulierte Objekte (Frame-Objekte), Kontextobjekte, Landmarkenobjekte, ähnliche Objekte zur Vermeidung von Mehrdeutigkeiten, stützende Objekte zur Vermeidung von artifiziellen, physikalisch ungewöhnlichen Szenarien usw.

Domänenobjekte, die in Graphikfenstern dargestellt werden sollen, erben von bestimmten Konzepten des HAMVIS-Grundmodells. Sichergestellt wird dieses im HAMVIS-Kontext durch die Aktionenkonzepte und die entsprechenden Kasusrollen im Aktionendekompositionsmodell. In Abhängigkeit von der Rolle, die die Objekte bei der (Inter-)Aktion spielen (z.B. „manipuliertes Objekt“), wird für die im HAMVIS-Grundmodell deklarierten Konzepte repräsentiert, welche Zusatzinformationen für den Benutzer notwendig sind (siehe Abbildung 113). Damit verbunden ist jeweils eine sog. *Ergänzungsfunktion*, die festlegt, wie die Informationen zur Laufzeit bestimmt werden können. HAMVIS bestimmt die Ergänzungsfunktion automatisch aus den Domänenmodellen, die der Entwickler für eine Anwendung deklariert.

Die „ergänzten“ Zusatzinformationen ließen sich auch durch weitere Kasusrollen der Aktion kennzeichnen. Jedoch wären diese Kasusrollen keine Ein- oder Ausgaberollen. Da die Funktionen zur Bestimmung der Rollenfüller nicht nur von den Aktionenkonzepten abhängen, sondern auch von den anwendungsspezifischen Domänenmodellen (insbesondere aus den durch HAMVIS berechneten Objektzerlegungsgraphen), werden Zusatzinformationen gesondert behandelt und nicht als Kasusrollen modelliert.

Zur Realisierung der Ergänzungsfunktionen muß vorausgesetzt werden, daß die Modellierung des Domänenwissens mit bestimmten Relationen erfolgt, die ebenfalls im HAMVIS-Grundmodell deklariert worden sind. Weiterhin muß, wie schon erläutert, die Diskursstellung von ergänzten Objekten festgelegt werden können. Ein Referenzsystem wie z.B. die Flugzeugkabine muß schon beim Starten einer Aktivität präsentiert werden und nicht erst, wenn die Plazierungsbereiche als Instanzen bestimmt worden sind. Einige Objekte müssen sich also aus unvollständigen, konzeptuellen Informationen herleiten lassen. Anders ausgedrückt: Einige Ergänzungsfunktionen werden zwar durch Objekte, die zur Laufzeit gezeigt werden sollen, motiviert, dürfen aber nur von der konzeptuellen Information abhängen, die zur Entwicklungszeit über die Objekte bekannt ist.

Weiterhin ist Wissen zu repräsentieren, das beschreibt, unter welchen Bedingungen eine Kombination von Visualisierungen möglich ist und welche Auswirkungen sich für die bei einer Kombination von Visualisierungen in Beziehung gesetzten Teilkonstituenten ergeben. Aus den Beziehungen zwischen Konstituenten lassen sich ggf. Einschränkungen für mögliche Anordnungen von Teilfenstern herleiten.

Zur Bestimmung von Lösungsvorschlägen für die Aufgaben der Dialogstrukturierung und zur Verifikation von Vorgaben des HAMVIS-Benutzers muß Wissen zur Strukturierung und Gestaltung von Visualisierungen für eine interaktive Schnittstelle explizit repräsentiert werden. Die Diskussion der IMMP-Systeme in Kapitel 2 hat gezeigt, daß hierzu die verschiedensten Ansätze entwickelt worden sind.

Aus welcher Perspektive wird Wissen explizit gemacht?

In IMMP-Systemen erfolgt die Inhaltszusammenstellung aus der agentenorientierten Perspektive in Form von Präsentationszielen eines automatischen Präsentationsplaners. Die hiermit verbundene Grundidee der Präsentationsplanung liegt darin, daß textuelle und graphische Ausgaben in klassischer Weise als Kommunikationshandlungen interpretiert werden, die mentale Zustände eines Rezipienten entsprechend beeinflussen. Diese Sicht- und Vorgehensweise hat sich m.E. für die situationsadaptierte automatische Generierung von Äußerungen zur Laufzeit bewährt, wobei jedoch die Generierungsarchitekturen bisweilen recht komplex werden können. In einigen Einsatzszenarien wurde allerdings der Bedarf einer Beeinflussungsmöglichkeit der Gestaltung von Präsentationen hervorgehoben (siehe die Ausführungen zu SAGE und auch Kapitel 2.4.4). Dieses kann entweder zur Entwurfszeit durch einen menschlichen Designer (SAGE) oder zur Benutzungszeit durch den Endanwender erfolgen (siehe WIP und seine Nachfolgersysteme sowie auch IBIS).

Wünsche und Vorgaben des menschlichen Designers lassen sich zwar in „Designdirektive“ (SAGE) oder auch in entsprechende Präsentationsziele eines automatischen Planungsagenten umsetzen (siehe Kapitel 2.4.4). Bei dem automatischen, agentenorientierten IMMPS-Ansatz geht man aber davon aus, daß die Wissensmodellierung (des Hintergrundsystems, siehe Abbildung 42) a priori und vollständig definiert ist. Im Entwicklungsszenario von HAMVIS ist jedoch zu berücksichtigen, daß anwendungsspezifische Modelle für Domänenobjekte von Darstellungsgesichtspunkten beeinflußt werden und kaum „in einem Rutsch“ vollständig ohne Berücksichtigung von Interaktionsgesichtspunkten deklariert werden können. Ich gehe davon aus, daß das zu repräsentierende Domänenwissen von Oberflächenaspekten entscheidend beeinflußt ist. Die Aktionenmodellierungkomponente von HAMVIS hat gezeigt, wie auf der Ebene der Aktionenmodellierung die Einflüsse koordiniert werden können. Auch auf der Ebene der Dialogstrukturierung können Anforderungen dazu führen, daß Domänenwissen neu erfaßt oder feiner modelliert werden muß. Diese Möglichkeit sollte m.E. Bestandteil der Architekturkonzeption sein und nicht notwendiges Übel.

Modelle für mentale Zustände des automatischen Präsentierers und des angenommenen menschlichen Rezipienten während des Planungsprozesses sind nicht einfach zu modellieren (siehe die Ausführungen zu IMMP-Systemen in Kapitel 2.4). André streicht heraus, daß z.B. ein deduktiver Planansatz zwar logisch korrekt und vollständig auf Grundlage der Eingaben ist oder sein kann, aufgrund der unsicheren Daten der Rezipientenmodellierung ist dieses allerdings kaum ein entscheidender Vorteil ([6], S. 123f.). Die häufig verwendeten propositionalen Modelle für die Rezipientenmodellierung können nur Teilaspekte der menschlichen Perzeptionsvorgänge grob erfassen.

Eine einfachere Repräsentationsform für Kommunikationswissen, bei der mentale Zustände der Produzenten und des Rezipienten nicht expliziert werden, sind kommunikative Schemata. In schemabasierten Ansätzen zur Repräsentation von Kommunikationswissen (z.B. mit kontextfreien Grammatiken wie bei McKeown [206]) äußern sich die Bezüge von Konstituenten in einem gewissen Sinne „extern“ durch den Typ des gemeinsamen Oberknotens. Im folgenden möchte ich einen Ansatz präsentieren, der durch den kognitionspsychologisch geprägten Ansatz von Leyton motiviert ist und gestattet, die Bezüge zwischen Teilkomponenten aus einer „internen“ Sicht heraus zu modellieren [174].

Ich möchte seinen Gedankengang an einem kleinen Beispiel erläutern, das aus [174] entnommen ist. Eine Konstituentenregel für Nominalphrasen sieht z.B. wie folgt aus:

```
NP -> DET, A, N
```

Leyton diskutiert als Beispiel für eine solche NP den Teilsatz „der ärgerliche Student“. Um nun zu ermitteln, wie die Menge von Konstituenten {DET, A, N} generiert wurde, muß eine „externe Inferenz“ getätigt werden, wie Leyton es nennt. Es muß der Oberknoten NP in der Konstituentenstruktur betrachtet werden. Die Beziehungen in einer solchen Nominalphrase sind gerichtet bzw. asymmetrisch. Das Adjektiv „ärgerlich“ beschreibt eine Eigenschaft des Nomens „Student“, d.h. „ärgerlich“ *modifiziert* „Student“ und nicht umgekehrt. Ähnliches gilt für den Artikel „der“. Leyton schlägt vor, diese Modifikationsbeziehungen auch in den Grammatikregeln in Form von sogenannten *Modifikationsregeln* explizit zu machen.

```
N -> A(N) -> DET(A(N)) = NP
```

Die grammatischen Kategorien wirken hier als Modifikatoren. Obwohl die Modifikation an sich gerichtet ist, ergeben sich doch Wechselwirkungen in beide Richtungen, denn der Modifikand beeinflußt die Menge der möglichen Modifikatoren. Die Modifikationsreihenfolge hat dabei keine Bedeutung für die Reihenfolge der Konstituenten im Satz, d.h. lineare Ordnungsregeln sind zusätzlich notwendig. Die Modifikationsreihenfolge definiert aber eine interne Historie von Zuständen, die sich von der externen Expansionshistorie der Phrasenstrukturansätze unterscheidet.

Welche Bedeutung hat nun die Idee der Modifikation für HAMVIS? Eine zu kommunizierende Basiseinheit, also eine Stelle aus der Aktionenmodellierung, muß durch einen Repräsentanten in der Dialogstruktur vertreten werden. Ich möchte diesen Repräsentanten als *Dialogstruktursegment* bezeichnen. Zu einem Segment gehören Informationen über die Aktion und über die Konzepte der beschriebenen Laufzeitobjekte. Wenn nun eine Basiseinheit in die Dialogstruktur integriert wird, wenn also ein Dialogstruktursegment erzeugt wird, so entsteht ggf. ein bestimmter *Modifikationsbedarf* bzw. eine *Modifikationsanforderung*. So wie also z.B. bestimmte Verben ein direktes Objekt zur Vervollständigung eines Satzes benötigen, benötigen in einer graphischen Darstellung Objekte unter bestimmten Bedingungen weitere Objekte, damit die Darstellung „vervollständigt“ wird. Es muß ein Modifikator von einem bestimmten Konzept bestimmt werden, d.h. aus dem Modifikanden wird eine Menge von Domänenobjekten hergeleitet, die jeweils als Modifikatoren dienen. Eine Modifikation kann explizit als Objekt repräsentiert werden und modelliert damit explizit die rhetorischen Bezüge zwischen Konstituenten (siehe die Rhethorical Structure Theorie [192] als den Urvater einer Vielzahl von Ansätzen, die die rhetorischen Bezüge zwischen Konstituenten explizit machen). Zur Beschreibung von Modifikationen und Modifikationsanforderungen lassen sich wiederum verschiedene Konzepte in einer Wissenbasis bereitstellen. Das allgemeinste, notwendige Konzept bestimmt die obligatorische Modifikationsform. Speziellere, optionale Modifikationsformen können durch Unterkonzepte beschrieben werden. Im nächsten Abschnitt wird detaillierter erläutert, mit welchen Repräsentationstechniken der hier skizzierte Ansatz in HAMVIS formalisiert wurde.

Durch welche Repräsentationstechniken wird Wissen explizit gemacht?

Die obige Beschreibung deutete es schon an: Auch für die Beschreibung der Dialogstruktur werden in HAMVIS Beschreibungslogiken als Basiswerkzeug eingesetzt. Während bei Planoperatoransätzen die „treibenden Kräfte“ der Inferenz noch nicht erfüllte Teilziele innerhalb der Dekomposition sind und bei Grammatikansätzen noch nicht expandierte Nonterminale die Inferenzaktivitäten auslösen, so lassen sich bei Beschreibungslogiken unvollständige Instanzen, die vervollständigt werden müssen,

als ein Mittel zur Modellierung des Aufbaus der Dialogstruktur einsetzen. Die Unvollständigkeit von Individuen (bzgl. einer Rolle) wurde in dieser Arbeit schon bei der Betrachtung von Modellen eingehend unter dem Thema der Existenzaussagen und der Berechnung von konkreten Rollenfüllern (Kapitel 3.1.6 und Kapitel 3.1.10) besprochen. Die nachfolgenden Abschnitte zeigen, wie die eingeführten Grundprinzipien auch für den Aufbau bzw. Ausbau einer Dialogstruktur ausgenutzt werden können.

3.3.4 Dialogstrukturierung mit HAMVIS

Anhand des XKL-Beispiels aus den vorangegangenen Kapiteln werden in diesem Abschnitt die Wissensbasen von HAMVIS und die Inferenzschemata zur Dialogstrukturierung erläutert. Neben den formalen Aspekten der Repräsentation von internen Strukturen ist auch die Präsentation von Informationen für den HAMVIS-Benutzer zu berücksichtigen. Die HAMVIS-Modelle für Dialogstrukturierungswissen müssen auch Informationen für die Kommunikation mit dem Schnittstellenentwickler enthalten, damit die Ableitungen transparent gemacht werden können. Die Ableitungen von HAMVIS für das XKL-Beispiel werden, wie schon im vorigen Unterkapitel, zusammen mit der graphischen Benutzungsschnittstelle von HAMVIS diskutiert. Aus der Diskussion der Benutzungsschnittstelle ergeben sich auch die konkreten Arbeitsaktivitäten des Schnittstellendesigners.

Bestimmung des Typs des Darstellungsfenster und der anwendbaren Modelle

Ausgangspunkt für die Dialogstrukturierung ist die Aktionenmodellierung und das daraus abgeleitete Laufzeitnetz. Für jede Stelle des Laufzeitnetzes, die innerhalb des Interaktionszyklus eine Menge von zu kommunizierenden Objekten enthält, sind durch die gewählten Aktionenkonzepte aus dem HAMVIS-Grundmodell entsprechende Einschränkungen für die Gestaltung der Darstellung definiert. Dazu gehört: die Menge der Konzepte der Objekte, die zur Darstellung zu verwendende Modellklasse, die Klasse des Darstellungsfensters usw.

Die mit der Aktionenzerlegung für eine Stelle im Laufzeitnetz festgelegten Einschränkungen werden dem HAMVIS-Benutzer innerhalb der HAMVIS-Dialogstrukturierungsoberfläche präsentiert. Zur Illustration der nachfolgend beschriebenen Ableitungsschritte zeigt Abbildung 114 hier die gesammelten Informationen zu der Stelle `has-moved-entity`.

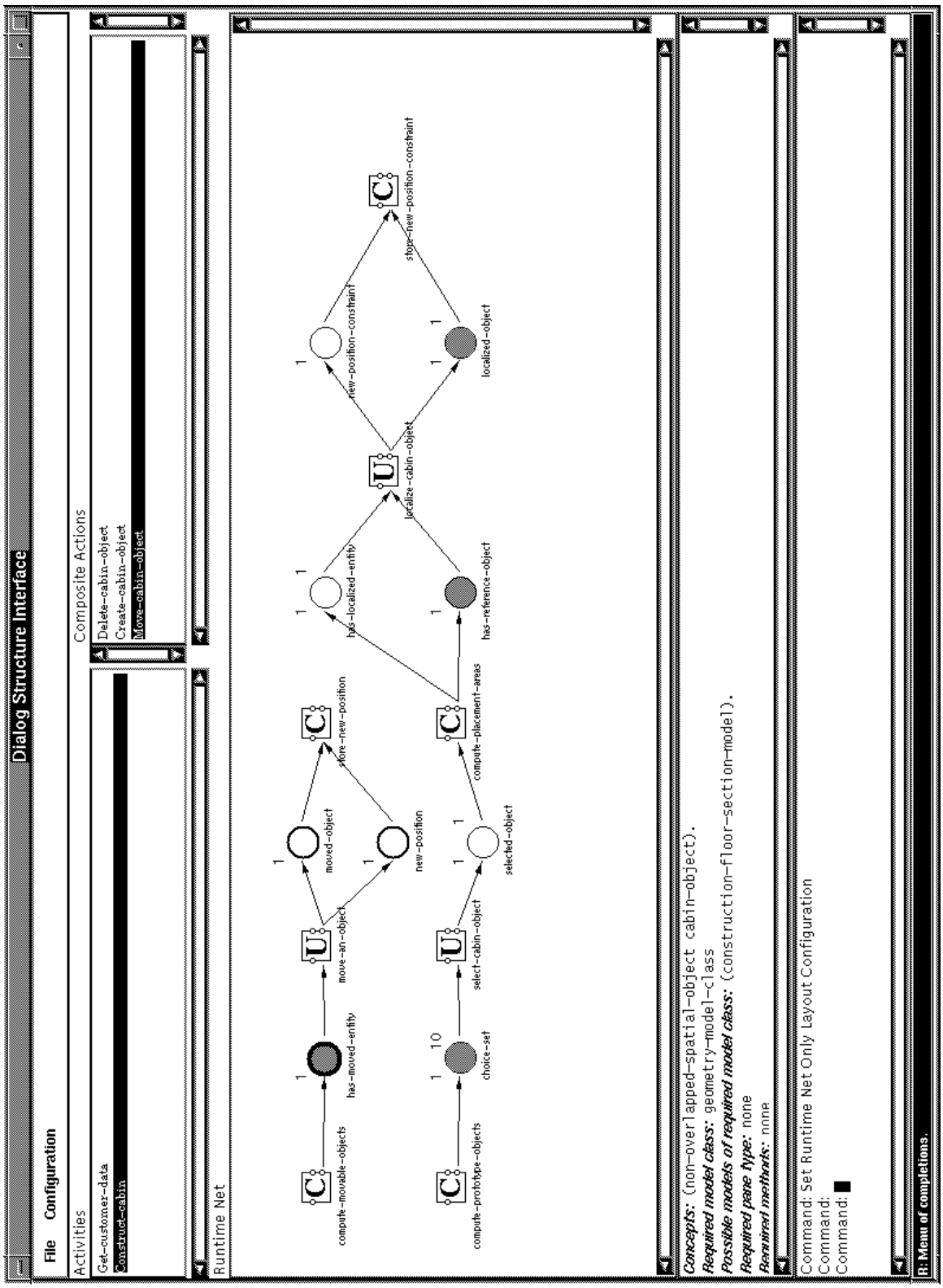


Abbildung 114. Darstellung des Laufzeitnetzes für die Aktivität `create-cabin-object` innerhalb der HAMVIS-Dialogstrukturierungsoberfläche mit abgeleiteten Einschränkungen für Stellen. Information über die markierte Stelle `has-moved-entity` (dicker Rand) werden im Fenster unterhalb des Netzes präsentiert. Die Stellen der aktuellen zusammengesetzten Aktion (siehe die rechte Tabelle `Composite Actions`) sind mit einem etwas dickeren Rand gezeichnet.

In der Abbildung wird die markierte Eingangsstelle `has-moved-entity` der Aktion `move-an-object` näher gekennzeichnet. Es wurde aus der Aktionenzerlegung ermittelt, daß es sich bei den Füllern der Stelle um Instanzen von `cabin-object` handelt, wobei durch das spezielle Konzept der Aktion `move-an-object` gefordert wird, daß zusätzlich `non-overlapped-spatial-object` gelten muß (siehe Abbildung 110).

Wenn nun in der Berechnungsfunktion `compute-movable-objects` zur Laufzeit Instanzen von `xkl-model:cabin-object` erzeugt werden, so muß `non-overlapped-spatial-object` nicht in jedem Modell gelten. Es muß also zur Entwicklungszeit noch ein Modell bestimmt werden, in dem sichergestellt ist, daß zur Laufzeit `non-overlapped-spatial-object` für alle Instanzen von `xkl-model:cabin-object` auch gilt.

Da die Aktion `move-an-object` vom Konzept `spatial-movement` ist (siehe Abbildung 108), wird zur Darstellung der Objekte der Eingangsstelle ein Modell von der Klasse `geometry-model-class` benötigt (siehe das Modell der Aktion in Abbildung 96). Durch die Aktionenkonzepte ist die *Klasse* des notwendigen Modells eingeschränkt. Für die Präsentation von Objekten, die zur Laufzeit in die Stelle `has-moved-entity` gesetzt werden, muß also noch ein *konkretes* Modell der Klasse `geometry-model-class` gefunden werden, in dem `non-overlapped-spatial-object` gilt. Kandidaten hierfür sind die für die Anwendung vom Systementwicklungsteam definierten Visualisierungsmodelle.

Definition: Mögliche Modelle für die Präsentation von Objekten einer Stelle

> Es sei durch die Aktionenzerlegung festgelegt, daß alle Instanzen, die zur Laufzeit in einer Stelle auftreten, von einem Konzept C_1 aus dem Basismodell der Anwendung subsumiert werden. Weiterhin sei durch das Konzept der mit der Stelle assoziierten Aktion als Zusatzeinschränkung das Konzept C_2 und die Modellklasse M festgelegt. Falls mehr als ein Zusatzkonzept gefordert wird, so stehe C_2 für das Konjunkt der Konzepte. Ein Modell m der Modellklasse M ist ein mögliches Modell für die Präsentation der Objekte einer Stelle genau dann, wenn gilt: Jede Instanz von C_1 wird im Modell m von C_2 subsumiert.

Um Modelle zu finden, lassen sich wiederum die Dienste der Beschreibungslogik aus dem Bereich der Subsumtionstests in Anspruch nehmen. Es wird „ohne Beschränkung der Allgemeinheit“ eine *Prototypinstanz* vom Konzept C_1 erzeugt, da nur so die Effekte von Regeln wirksam werden (siehe z.B. die impliziten Regeln von `define-concept-refinement` und die Regeln, die durch `:asserted-concepts` definiert werden). Für diese Instanz wird ein Modell m der Modellklasse M gesucht, in dem sie von C_2 subsumiert wird.

HAMVIS legt noch weitere Einschränkungen bezüglich der Wahl von Modellen fest. Falls eine Modellklasse gefordert wird, die eine Unterklasse von `geometry-model-class` ist, so müssen die Objekte zusätzlich von einem Basiskonzept oder von einem kategorialen Oberkonzept subsumiert werden (siehe die Definition von direkter Visualisierbarkeit auf Seite 201).

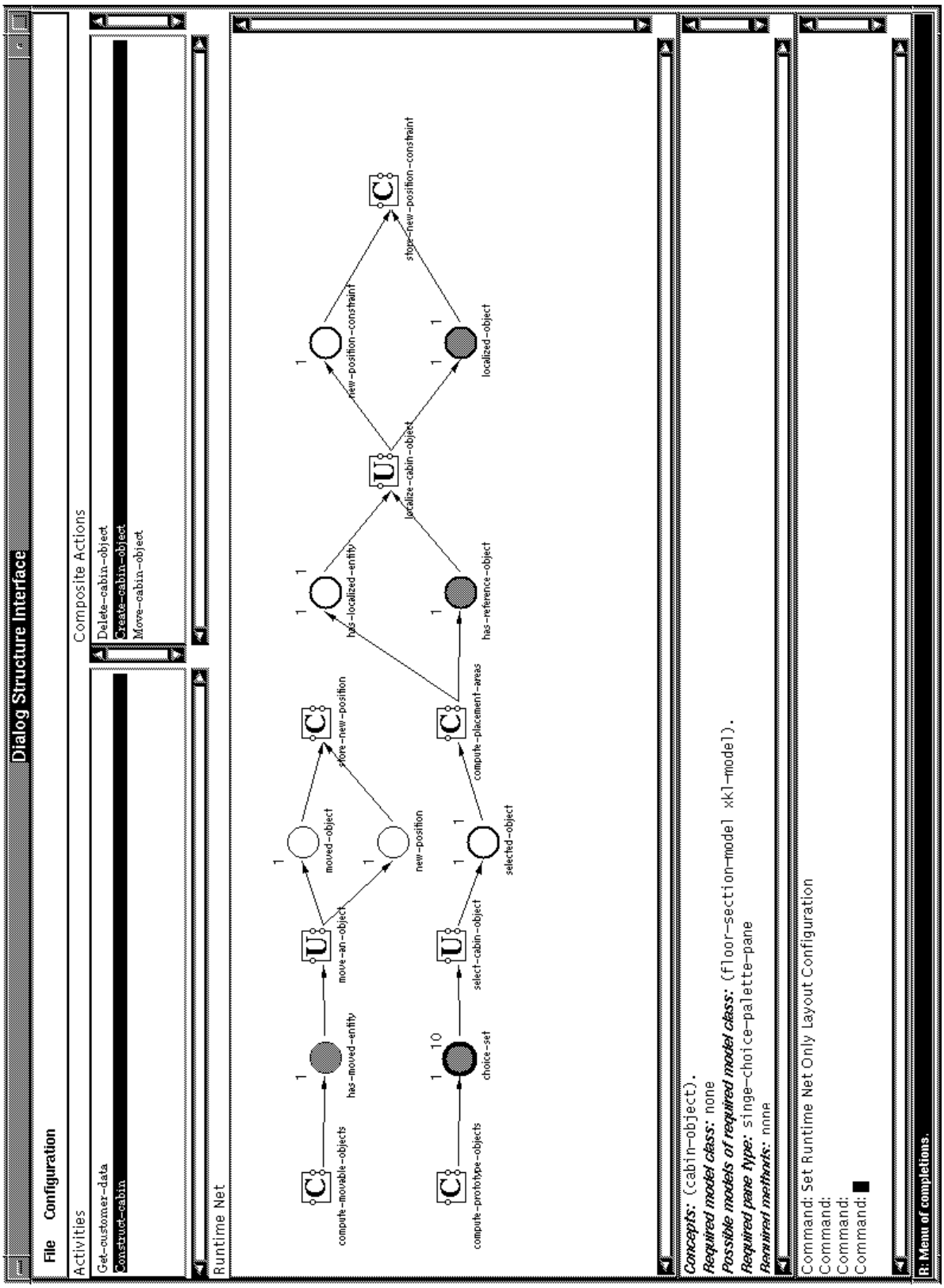


Abbildung 115. Darstellung der Darstellungseinschränkungen für die Stelle `choice-set`.

Die Bedingungen werden in dem konkreten Beispiel aus Abbildung 114 nur von dem Modell `construction-floor-section-model` erfüllt. Wie aus der Wissensbasisdefinition des Modells `construction-floor-section-model` ersichtlich ist (Abbildung 77), wird in diesem Modell für jede Instanz von `cabin-object` auch `non-overlapped-spatial-object` assertiert. Instanzen von

`xkl-model:cabin-object` werden im Modell `construction-floor-section-model` von einem kategorialen Oberkonzept subsumiert (`construction-floor-section-model:cabin-object`).

HAMVIS zeigt die auf dieses Weise bestimmten „möglichen Modelle“ im Fenster unter dem Laufzeitnetz unter der Überschrift „*Possible models of required model class:*“ (siehe Abbildung 114). Es kann auch der Fall eintreten, daß mehr als ein Modell möglich ist. In Abbildung 115 wird ein Beispiel präsentiert, in dem dieses der Fall ist.Die Aktivität `create-cabin-object` enthält als Handlungsalternative weiterhin noch die zusammengesetzte Aktion `create-cabin-object`. Die Einschränkungen für die Eingangsstelle `choice-set` der darin enthaltenen elementaren Benutzerhandlung `select-cabin-object` werden in Abbildung 115 gezeigt. Für die Darstellung wird nicht ein Graphikfenster, sondern ein spezielles Fenster zur Darstellung einer Palette verwendet (siehe die Definition des zugehörigen Aktionenkonzepts `selection-from-small-palette` aus Abbildung 95). Wenn keine Einschränkungen für die Klasse des Darstellungsfensters („*Required pane type:*“) gegeben ist, wird ein allgemeines HAMVIS-Graphikfenster vorgesehen (siehe Abbildung 114).

Zu beachten ist, daß HAMVIS bei der Liste der möglichen Modelle im Informationsfenster nur die gröbsten Modelle auflistet. Es wird also zunächst nur `floor-section-model` als Modell für die Darstellung der Objekte der Stelle `choice-set` betrachtet. Das feinere Modell `construction-floor-section-model`, das die Bedingungen auch erfüllt, wird hier nicht (direkt) aufgeführt. Die Idee dahinter ist, die Einschränkungen der Modelle so allgemein wie möglich zu halten.

Iterationen bei der Modellbildung

Auch für die anderen Stellen im Laufzeitnetz können in dem hier betrachteten XKL-Beispiel die möglichen Modelle hergeleitet werden. Falls allerdings für einige Stellen kein Modell gefunden werden kann, das die durch die Aktionenkonzepte gestellten Bedingungen erfüllt, so gibt es mehrere Möglichkeiten:[101]

- Es wird für die Aktion, die die zu starken Einschränkungen definiert hat, ein weniger spezielles Aktionenkonzept gewählt (soweit möglich).
- Es wird das in den bestehenden Domänenmodellen erfaßte Wissen überprüft und ggf. verfeinert.
- Es wird ein neues Domänenmodell definiert, das für die Visualisierungssituation die notwendige Modellbildung vorsieht.

Nehmen wir für die weitere Diskussion an, daß für jede Stelle mindestens ein Modell gefunden werden kann. Die zu kommunizierenden Einheiten des Laufzeitnetzes (grauen Stellen) können dann

101. In der Oberfläche kann ein fehlendes Modell durch ein entsprechenden Anzeichen (z.B. ein Blitz an der Stelle) verdeutlicht werden, so daß der HAMVIS-Benutzer nicht nach Stellen mit fehlendem Modell suchen muß. Ich verzichte hier auf eine ausführliche Betrachtung der entsprechenden Visualisierungstechniken.

nacheinander in die Dialogstruktur integriert werden.[102] In Abbildung 116 wurde die Stelle `has-moved-entity` als erstes Segment in die Dialogstruktur integriert.

In der rechten Hälfte des Interaktionsfensters werden Informationen über die Dialogstruktur präsentiert. Im obersten Fenster werden die verschiedenen „Konstruktionsschritte“ als Varianten in Form eines Baumes dargestellt (`Dialog Structures`). Am Anfang steht der Knoten für die Integration von `has-moved-entity`.[103] Für jedes mögliche Modell wird ein solcher Knoten automatisch von HAMVIS erzeugt. Wie aus den im linken unteren Teilfenster ausgedruckten Stelleninformationen hervorgeht (siehe auch die obige Diskussion), kommt für `has-moved-entity` nur ein Modell in Frage (`construction-floor-section-model`, siehe die Knotenbeschriftung in Klammern). Falls mehrere Modelle anwendbar sind, so werden mehrere „Varianten“ verwaltet. Im Vorgriff möchte ich hierzu kurz auf Abbildung 126 verweisen.

Die markierte Dialogstruktur des Variantenfensters `Dialog Structures` wird im darunterliegenden Fenster in Form eines Graphen genauer dargestellt (`Selected Dialog Structure`, siehe Abbildung 116). Der hervorgehobene Knoten stellt das Dialogstruktursegment für die gerade integrierte Stelle `has-moved-entity` dar (siehe die Knotenbeschriftung). Aus der Darstellung für die Dialogstrukturvariante geht folgendes hervor: Für die Präsentation der durch das Segment beschriebenen Objekte wird ein eigenes Darstellungsfenster benötigt (siehe den Oberknoten in der Dialogstruktur mit der Beschriftung „Pane”). Im darunterliegen Fenster werden für die markierten Dialogstrukturknoten die abgeleiteten Informationen präsentiert. Für die verschiebbaren Objekte wird ein Referenzsystem benötigt. In der Dialogstruktur wurde schon ein entsprechender Segmentknoten zur Beschreibung der Dialogstrukturinformationen für dieses Referenzsystem eingefügt. Weiterhin muß die Umgebung der beweglichen Objekte präsentiert werden (Knoten `Environment`).

---

102. Die Integration einer Stelle erfolgt durch Mausklick. Ich gehe hier nicht näher auf die realisierten direktmanipulativen Interaktionstechniken ein. Die integrierten Stellen erscheinen in den in dieser Arbeit verwendeten Schwarz-Weiß-Darstellungen in hellerem grau. Auf einem Farbbildschirm werden statt dunkelgrau und hellgrau die Farben magenta und seegrün verwendet.

103. Der Knoten ist ein Nachfolger der „leeren“ Dialogstrukturvariante.

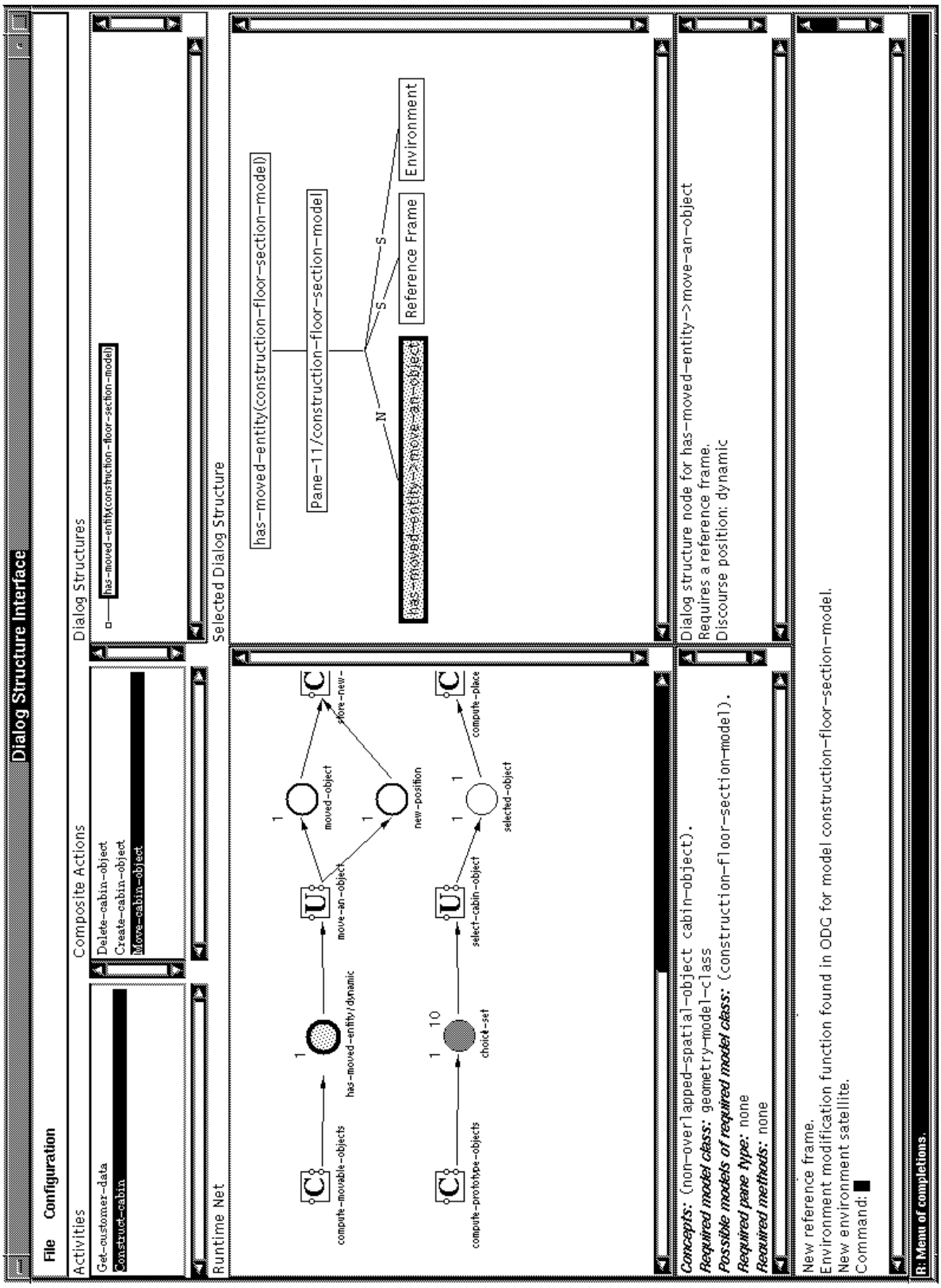


Abbildung 116. Andere Fensterkonfiguration der Dialogstrukturierungsoberfläche mit Darstellungsfenstern für eine Dialogstruktur.

Im folgenden möchte ich die Dialogstrukturwissensbasen beschreiben und die internen Ableitungsschritte erläutern, die zur Herleitung dieser Struktur geführt haben. Informationseinheiten auf der Ebene der Dialogstrukturierung sind jeweils mit Stellen im Laufzeitnetz verknüpft. Die Informationseinheiten werden als *Dialogstruktursegmente* bezeichnet. Sie werden schrittweise in eine Dialogstruktur integriert. Bei der Integration von Dialogstruktursegmenten in die Dialogstruktur müssen notwendige Modifikatoren hergeleitet werden. Dabei sind die Konzepte der Anwendungsobjekte und die Konzepte der Aktionen, die diese Objekte manipulieren, zu berücksichtigen. Das zu integrierende Dialogstruktursegment und die notwendigen Modifikator-Dialogstruktursegmente müssen zu einem *Modifikationsaggregat*, das die Modifikationsform repräsentiert, verbunden werden. Die Ebene der Aggregate modelliert die Modifikationsbeziehungen zwischen Konstituenten durch ein explizites Objekt. Ein Dialogstrukturaggregat wiederum wird einer höheren *Dialogstruktureinheit* zugeordnet, die jeweils an der Benutzungsoberfläche in einem Teilfenster (pane) präsentiert wird. Verschiedene Dialogstruktureinheiten bilden dann eine vollständige *Dialogstruktur* (DS). In Abbildung 117 ist eine grobe Skizze der vier Ebenen der Dialogstruktur für die Aktivität `construct-cabin` von XKL wiedergegeben. Informationen über die Darstellung von Objekten im Laufzeitnetz wird in der Dialogstruktur in DS-Segmentobjekten repräsentiert.

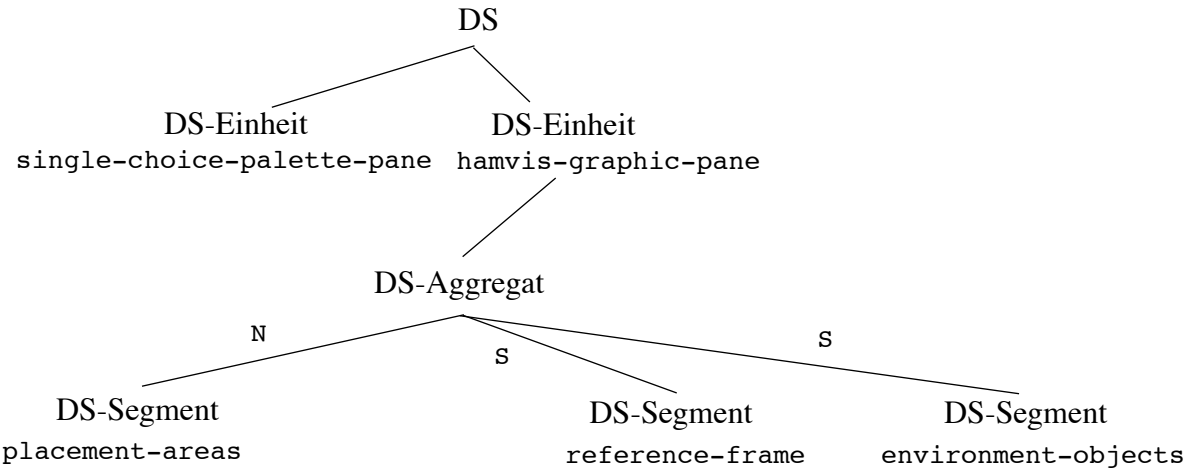


Abbildung 117. Skizze der Schichtung und Zerlegung von Dialogstrukturebenen: „Segmente“, „Aggregate“ und „Einheiten“.

Die Kantenbeschriftungen „N“ und „S“ stehen für Nukleus und Satellit. Die Nuklei (oder Kopf-Segmente) sind jeweils Ausgangspunkt für die Bildung eines Modifikationsaggregats. Innerhalb eines Aggregats wird der Nukleus von seinen Satelliten modifiziert. Die zu einem Nukleussegment bekannten Informationen bestimmten die Art der notwendigen Modifikation.

Für die Herleitung einer Dialogstruktur im Rahmen eines Bottom-Up-Verfahrens sind folgende Inferenzschemata notwendig: Für ein Dialogstruktursegment, das in eine bestehende Dialogstruktur eingebettet werden soll, muß ein entsprechender Modifikator instantiiert werden (Bildung eines neuen DS-Segments einschließlich Aggregierungsschritt). Für jedes Aggregat muß wiederum eine Dialogstruktureinheit instantiiert werden, die wiederum in die jeweils betrachtete Dialogstrukturvariante (Knoten DS) eingehängt werden muß. Jeder Knoten der Dialogstruktur verwaltet dabei die zur Darstellung notwendigen Informationen und leitet Informationen entsprechend weiter.

Im nachfolgenden Abschnitt wird erläutert, wie sich die Inferenzschemata mit Beschreibungslogiken (bzw. Erweiterungen hiervon) realisieren lassen.

3.3.5 Formale Modellierung von Dialogstrukturierungswissen

Dialogstrukturknoten lassen sich als ABox-Individuen repräsentieren. Eine Dialogstruktur wird als Graph repräsentiert, d.h. Dialogstrukturknoten werden über Relationen, die die Graphkanten darstellen, miteinander in Beziehung gesetzt. Je nach Stellung in der Strukturhierarchie werden zur Beschreibung der ABox-Individuen unterschiedliche Konzepte aus dem im folgenden vorgestellten Teil des HAMVIS-Grundmodells verwendet. Die nachfolgenden Abschnitte zeigen, wie mit Hilfe der Objektklassifikation der Beschreibungslogik unter Verwendung von definierten Konzepten sowie Methoden zur Füllerberechnung die Zusammensetzung einer Dialogstruktur modelliert werden kann.

```
(define-primitive-concept ds-node classic-thing)

(define-primitive-role has-ds-component has-decomposition
                       :inverse ds-component-of)

(define-primitive-role represents-place nil)
(define-primitive-role describes-displayed-object nil)
(define-primitive-role generated-by-action nil)

(define-primitive-concept dss
    (and ds-node
         (at-least 1 ds-component-of)
         (at-least 1 has-discourse-purpose)
         (at-most 1 has-discourse-purpose)
         (all has-discourse-purpose discourse-purpose)
         (at-least 1 generated-by-action)
         (at-most 1 generated-by-action)
         (all generated-by-action user-action))
  (:asserted-concepts (:at-least 1 (k has-discourse-purpose))))

(define-primitive-role has-nucleus has-ds-component
                       :inverse nucleus-of)

(define-primitive-role has-satellite has-ds-component
                       :inverse satellite-of)

(define-primitive-concept dsa
    (and ds-node
         (at-least 1 ds-component-of)
         (at-least 1 has-nucleus)
         (at-most 1 has-nucleus)
         (all has-nucleus dss)
         (all has-satellite dss)))
```

Abbildung 118. Basiskonzepte und -rollen für Dialogstrukturknoten (Teil 1).

Konzepte und Rollen zur Modellierung von Dialogstrukturkomponenten

In Abbildung 118 werden die notwendigen Basiskonzepte und -rollen für Segmente und Aggregate deklariert. Die Basisrelation zwischen Dialogstrukturknoten (Instanzen von `ds-node`) zum Aufbau der Dialogstruktur heißt `has-ds-component`, eine Unterrelation zu `has-decomposition` (vgl. Abbildung 56, “Ausschnitt aus dem HAMVIS-Grundmodell mit den Basisrelationen für die Dekompositionsbeschreibung und Beschreibung von geometrischen Daten. Die Formrepräsentation erfolgt

mit dem CSG-Modellierer VANTAGE (siehe Text).," auf Seite 159). Zur Unterscheidung von Nuklei und Satelliten werden zwei entsprechende Unterrollen zu `has-ds-component` mit entsprechenden Inversen deklariert.

Eine Bottom-Up-Konstruktion einer Dialogstruktur wird durch den HAMVIS-Benutzer Schritt für Schritt durch Integration einzelner Stellen initiiert (s.o.). In Abbildung 116 wurde die Dialogstruktur nach der Integration der Stelle `has-moved-entity` skizziert. Die folgenden Abschnitte zeigen, wie die Struktur mit Hilfe der beschreibungslogischen Wissensbasis hergeleitet wurde. Zunächst einmal wird der Auftrag, eine Stelle in die Dialogstruktur zu integrieren, aus logischer Sicht dargestellt.

Integration einer Stelle in die Dialogstruktur aus logischer Sicht

Für jede Stelle des Laufzeitnetzes wird ein Dialogstruktursegment erzeugt. Ein Dialogstruktursegment repräsentiert die für die Stellen jeweils bekannten Informationen und dient als Ausgangspunkt für die beschreibungslogischen Ableitungsschritte zum Aufbau der Gesamtstruktur.

Dialogstruktursegmente (Konzept `dss`) sind durch die folgenden Rollen gekennzeichnet: `represents-place` stellt die Beziehung zu der Stelle im Laufzeitnetz her, `describes-displayed-object` beschreibt die Objekte, die durch ein Segment in der Dialogstruktur repräsentiert werden und `generated-by-action` beschreibt die Aktion, die zur Generierung des Segmentes geführt hat. Bei der Integration einer Stelle in die Dialogstruktur wird die nachfolgende Aktion (repräsentiert durch die nachfolgende Transition im Petri-Netz) mit dem Dialogstruktursegment über die Rolle `generated-by-action` in Beziehung gesetzt. Die Konzepteinschränkungen der Stelle werden als Rollenrestriktionen für `describes-displayed-object` etabliert.

Ein konkretes Segment wird als ABox-Instanz repräsentiert. Die Assertionen für ein spezielles Segment werden automatisch aus dem Laufzeitnetz ermittelt. Nehmen wir an, die erste ausgewählte Stelle sei `has-moved-entity` (vgl. Abbildung 116). Das erste Segment bekommt den Namen `dss-1`. Es ergeben sich folgende Assertionen zur Beschreibung dieses Segments (in KRSS-Syntax).

```
(state (instance dss-1 ds-segment))
(state (instance dss-1 (all describes-displayed-object
                            (and cabin-object non-overlapped-spatial-object))))
(state (related dss-1 move-an-object generated-by-action))
```

Jedes Segment wird zu mindestens einem übergeordneten Knoten in der Diskursstruktur in Beziehung gesetzt (Rolle `ds-component-of`, siehe Abbildung 118). Die übergeordneten Knoten werden als Aggregate bezeichnet und dienen zur Realisierung von Modifikationsanforderungen. Knoten auf dieser Ebene werden automatisch durch HAMVIS erzeugt (s.u.). In Abbildung 118 wird das Grundkonzept `dsa` zur Repräsentation eines Modifikationsaggregates definiert. Ein DS-Aggregat kann genau einen Nukleus haben. Füller der Rolle `has-nucleus` oder `has-satellite` sind wiederum DS-Segmente. Zur Beschreibung von Modifikationskonzepten sind verschiedene *definierte* Konzepte vorgesehen. Über definierte Konzepte und beschreibungslogische Inferenzen lassen sich Modifikationsanforderungen automatisch herleiten. Die Modifikatorkonzepte werden etwas später detaillierter erläutert. Ich möchte zunächst noch die Repräsentation von Diskursinformationen für den Ausgabenverwalter schildern.

Diskursinformationen und kommunikative Funktionen von DS-Segmenten

Zur Bestimmung der Zeichenattribute muß die kommunikative Funktion von Konstituenten innerhalb der Dialogstruktur explizit repräsentiert werden (siehe hierzu Abschnitt 2.4). Ich übernehme hier den Ansatz von Hovy und modelliere eine taxonomische Hierarchie von Diskurszwecken (siehe Abschnitt 2.4.3 sowie Abbildung 51). Die Grundidee besteht darin, mit den Diskurszweckkonzepten auch spezielle Darstellungseinschränkungen zu verbinden, d.h. je spezieller die Diskurszweckkonzepte sind, desto spezieller sind auch die mit den Konzepten verbundenen Einschränkungen für die Wahl von Darstellungsparametern.

HAMVIS repräsentiert die in der Anwendungsklasse wichtigen Diskurszwecke: „Darstellungshintergrund“ „manipuliertes Objekt“, „Referenzsystem“, „Umgebungsobjekt“, „Auswahlalternative“ usw. Im Kontext von HAMVIS werden hier nur die semantischen DS-Segmentrelationen betrachtet (vgl. Abbildung 51). Die in Abbildung 51 aufgeführten präsentationsbezogenen Segmentrelationen beziehen sich schon auf die Darstellung. Ich werde in Abschnitt 3.4 im Zusammenhang mit der Bestimmung von Darstellungsattributen darauf zu sprechen kommen.

Ähnlich wie Hovy verwendet HAMVIS nicht einen kleinen Satz von kanonischen Konzepten, sondern geht davon aus, daß die Wissensbasis (also das Grundmodell) ggf. für neue Anwendungsfelder entsprechend erweitert werden muß. HAMVIS bietet den Rahmen, in den die Wissensquellen entsprechend integriert werden können. Die formale beschreibungslogische Definition dieser Diskurszwecke ist in Abbildung 119 skizziert.

DS-Segmente sind über die Rolle `has-discourse-purpose` jeweils mit genau einem Objekt verbunden, das von `discourse-purpose` subsumiert wird. Durch dieses Objekt wird der Diskurszweck des Segments modelliert. Die hierzu eingeführten Beschreibungskonzepte sind in Abbildung 119 widergegeben. Jedes Diskurszweck-Individuum steht über die Rollen `has-discourse-position` zu einem Objekt zur Beschreibung der Diskursposition in Beziehung. Spezielle Objekte, die vom Ausgabenverwalter in besonderer Weise ausgenutzt werden, sind ebenfalls in Abbildung 119 deklariert.

Der Diskurszweck von Segmenten wird ebenfalls durch beschreibungslogische Inferenzen über notwendige Modifikatorkonzepte auf der Aggregatebene automatisch bestimmbar. Ich gehe etwas später genauer auf die Bedeutung der Individuen und Konzepte für die Darstellungsgestaltung ein.

In Abbildung 120 werden die höheren Dialogstrukturebenen beschrieben. Für Dialogstruktureinheiten sind die Rollen `uses-model` und `required-pane-type` relevant. Jedes Teilfenster (pane) hat eine bestimmte Klasse und stellt Objekte unter Verwendung eines bestimmten Modells dar. Eine der Aufgaben der Dialogstruktur ist es, genau diese Informationen herzuleiten und zu präsentieren (s.o.).

Die Konzeptdefinition `dsu` definiert auf dieser Ebene entsprechende Einschränkungen auch für die DS-Zerlegungsrelation `has-ds-component`. An der Spitze der Dialogstruktur steht ein Knoten vom Konzept `ds`. HAMVIS erzeugt zur Herleitung einer Dialogstruktur automatisch ein Individuum vom Konzept `ds`, das die Wurzel der Dialogstruktur repräsentiert. Nennen wir es hier einmal `ds-2`. Durch Inferenzen muß nun eine „Verbindung“ zwischen dem Ausgangssegment `dss-1` und dem Wurzel der Dialogstruktur `ds-2` hergestellt werden. Mit anderen Worten: Das Segment `dss-1` muß in `ds-2` „integriert“ werden. Dieses wird durch Subsumtionsschlüsse und Berechnung von Rollenfüllern durch K-Operatoren gesteuert.

```
(in-model hamvis-upper-model)

(define-primitive-concept discourse-position classic-thing)
(define-distinct-individual static)
(define-distinct-individual dynamic)
(define-distinct-individual dynamic-persistent)
(define-distinct-individual dynamically-activated)

(state (instance static discourse-position))
(state (instance dynamic discourse-position))
(state (instance dynamic-persistent discourse-position))
(state (instance dynamically-activated discourse-position))

(define-primitive-role has-discourse-purpose nil)
(define-primitive-role has-discourse-position nil)

(define-primitive-concept discourse-purpose
    (all has-discourse-position discourse-position))

(define-disjoint-primitive-concept background
    (discourse-purpose)
    (and discourse-purpose (fillers has-discourse-position static)))

(define-disjoint-primitive-concept reference-frame-modifier
    (discourse-purpose)
    (and discourse-purpose (fillers has-discourse-position static)))

(define-disjoint-primitive-concept descriptor-for-manipulated-object
    (discourse-purpose)
    (and discourse-purpose (fillers has-discourse-position dynamic)))

(define-disjoint-primitive-concept descriptor-for-created-object
    (discourse-purpose)
    (and discourse-purpose (fillers has-discourse-position dynamic-persistent)))

(define-disjoint-primitive-concept selection-alternative
    (discourse-purpose)
    (and discourse-purpose (fillers has-discourse-position dynamically-activated)))

(define-disjoint-primitive-concept environment-modifier
    (discourse-purpose)
    (and discourse-purpose (fillers has-discourse-position dynamic-persistent)))

(defparameter *discourse-purpose-ranking*
    '(background
      descriptor-for-manipulated-object
      reference-frame-modifier
      selection-alternative
      environment-modifier
      descriptor-for-created-object))
```

Abbildung 119. Basiskonzepte und -rollen für Diskursinformationen für den Ausgabenverwalter.

.

```
(define-primitive-role uses-model nil)
(define-primitive-role required-pane-type nil)

(define-primitive-concept dsu
    (and ds-node
         (all uses-model (test-h hamvis-model-class-p))
         (all has-ds-component dsa)
         (at-least 1 ds-component-of))
  (:asserted-concepts (at-least 1 (k ds-component-of))))

(defun hamvis-model-class-p (object)
  (typep object 'hamvis-model-class))

(define-primitive-concept ds
    (and ds-node
         (all has-ds-component dsu)
         (at-most 0 ds-component-of)))
```

Abbildung 120. Basiskonzepte und -rollen für Dialogstrukturknoten (Teil 2, siehe auch Abbildung 118).

Aufbau einer Dialogstruktur durch Integration von Segmenten

In der Konzeptdefinition von `dsu` wird in einer Regel für assertierte Konzepte der K-Operator verwendet, d.h. Dialogstruktureinheiten werden automatisch in die Dialogstruktur eingehängt. Wenn also hergeleitet wird, daß eine DS-Einheit (also ein Fenster) benötigt wird, so wird dieses einfach in der Dialogstruktur vermerkt.

```
(define-method compute-role-fillers ((ind dsu)
                                     (role-name (eql 'ds-component-of))
                                     (model hamvis-model-class))
  (with-dialog-structure (ds)
    (let ((model (context-model (current-context ds))))
      (values (list ds)
              (considering (ind role-name model)
                (holds
                   ind
                   `(fills uses-model ',model))
                '("Using a model failed."))))))
```

Abbildung 121. Methode zur Berechnung des Oberknotens einer Dialogstruktureinheit.

Eine entsprechende Methode für den K-Operator in `dsu` ist hier in Abbildung 121 illustriert. Die „aktuelle Dialogstruktur“ wird durch `with-dialog-structure` referenziert und steht über eine lokale Variable (hier: `ds`) zur Verfügung. Durch die Methode werden weitere Assertionen zu der DS-Einheit berechnet.

Im Gegensatz zu der einfachen Form der Rollenfüllerberechnung aus Kapitel 3.1.10 wird in Abbildung 121 eine erweiterte Form verwendet. Neben den Werten für die Rollenfüller können noch weitere Bedingungen angegeben werden, die gelten müssen, damit die Rollenfüller tatsächlich eingesetzt werden. Bedingungen werden mit der Form

```
(considering (ind role-name model)
   (holds ind1 concept-term ...)
   failure-explanations)
```

angegeben. Die Terme *ind*, *role-name* und *model* kennzeichnen die entsprechenden Parameter der Methode zu `compute-role-fillers`. Die berechneten Rollenfüller für *role-name* (hier der Wert von `(list ds)`) werden eingesetzt, wenn die Assertierung der Konzeptterme für das Individuum *ind1* konsistent ist (*ind* und *ind1* können identisch sein). Dieses wird wiederum durch die ABox von CLASSIC überprüft. Eine Inkonsistenz führt hier dazu, daß kein Rollenfüller berechnet werden kann. Für die Erläuterung einer Inkonsistenz an der Benutzungsoberfläche kann als dritter Term bei `considering` eine Fehlererklärung gegeben werden. Zusammen mit den unter `values` (s.o.) angegebenen positiven Erklärungstexten lassen sich für die HAMVIS-Oberfläche die notwendigen Ausgabetexte zusammenstellen.

Falls eine Inkonsistenz auftritt, werden die durch die Konzeptterme beschriebenen Aussagen wieder zurückgenommen und die Instanz bleibt unvollständig, d.h. der Aufbau der Dialogstruktur schlägt fehl und es wird eine Iteration in der Modellbildung notwendig (s.o.).

Für die im HAMVIS-Grundmodell deklarierten Konzepte zur Dialogstrukturierung sind entsprechende Methoden zur Füllerberechnung definiert.

Die in den Abbildungen 118 und 120 eingeführten Konzepte dienen dazu, das Basisgerüst der Dialogstruktur aufbauen zu können. Wir haben schon gesehen, daß zur Bestimmung der notwendigen Darstellungsfenster und deren Typen sowie der notwendigen Zusatzinformationen für die Darstellung der manipulierten Objekten auf die Aktionen und die Domänenobjekte Bezug genommen wird. Erst wenn bekannt ist, auf welche Handlung sich ein DS-Segment bezieht und welche Objekte dargestellt werden (z.B. räumliche Objekte) kann ein DS-Segment entsprechend in der Dialogstruktur verankert werden. Für einige Segmente bedeutet dieses, sie können erst in die Diskursstruktur integriert werden, wenn weitere notwendige Segment bestimmt werden können, wenn also notwendige Modifikatoren hergeleitet wurden.

Realisierung von Modifikationsanforderungen

In vorigen Abschnitten wurde die Bedeutung einer Modifikationsanforderung für den Aufbau der Dialogstruktur und für die Repräsentation von Kommunikationswissen erläutert. Ich möchte nun schildern, wie Modifikationsanforderungen auf der technischen Ebene realisiert werden.

Zur Realisierung von Modifikationsanforderungen, d.h. zur Bestimmung eines Modifikators, werden wiederum die im HAMVIS-Grundmodell definierten Methoden für die generische Funktion `compute-role-fillers` verwendet. Das Konzept eines Modifikators wird durch die Art der gewünschten Modifikationswirkung bestimmt. Die notwendige Modifikationswirkung für ein Dialogstruktursegment richtet sich nach den konzeptuellen Informationen über die Aktion und die mit der Aktion in Zusammenhang stehenden Domänenobjekte.

Zur Modellierung dieser (Modifikations-)Einflüsse reichen jedoch die bisher eingeführten Konzepte aus den Abbildungen 118 und 120 noch nicht aus. Inferierbares Wissen über speziellere Segmente wird im HAMVIS-Grundmodell mit definierten Konzepten deklariert. Abbildung 122 schildert zur Illustration des Prinzips einen Ausschnitt aus der HAMVIS-Wissenbasis zur Dialogstrukturierung.

```
(define-primitive-concept reference-frame-requiring-dss dss)

(define-concept spatial-action-dss
    (and dss
         (at-least 1 describes-displayed-object)
         (all describes-displayed-object spatial-object)
         (at-least 1 generated-by-action)
         (all generated-by-action
              (all has-manipulated-object spatial-object)))
  (:asserted-concepts reference-frame-requiring-dss
                      (at-least 1 (k ds-component-of))
                      (fillers has-discourse-purpose
                               descriptor-for-manipulated-object)))

(define-concept selection-action-dss
    (and dss
         (at-least 1 generated-by-action)
         (all generated-by-action selection-from-small-palette))
  (:asserted-concepts (at-least 1 (k ds-component-of))
                      (all ds-component-of
                           (at-most 1 has-ds-component))
                      (fillers has-discourse-purpose selection-alternative)))
```

Abbildung 122. Beschreibungen von Dialogstruktursegmenten für spezielle Aktionenkonzepte.

Das hier beispielhaft vorgestellte definierte DSS-Konzept `spatial-action-dss` behandelt Informationseinheiten für Aktionen, bei denen das manipulierte Objekt ein räumliches Objekt ist (siehe die Einschränkung `(all has-manipulated-object spatial-object)` für die Rolle `generated-by-action`). Weiterhin müssen die in der korrespondierenden Stelle auftretenden Objekte räumliche Objekte sein (Einschränkung für `describes-displayed-object`).

Betrachten wir wieder das Beispiel des oben erzeugten DS-Segments `dss-1`. Die Einschränkungen für `spatial-action-dss` sind in diesem Fall erfüllt. In Abbildung 114 haben wir gesehen, daß die Objekte in der Stelle Instanzen von `cabin-object` sind, ein Unterkonzept von `spatial-object`. Die Kabinenobjekte werden durch die Aktion `move-an-object` verschoben (siehe die Einschränkung für die Kasusrolle `has-moved-entity` in Abbildung 108, “Automatische Spezialisierung von `move-an-object`.,” auf Seite 237). Die Rolle `has-moved-entity` ist eine Unterrolle von `has-manipulated-object` (siehe die Definition in Abbildung 96, “Aktionenkonzepte für die Anwendungsklasse der Layoutsysteme (Teil 2).,” auf Seite 221). Auch die Kardinalitätenrestriktion zu `describes-spatial-object` ist erfüllt (siehe die Darstellung des Laufzeitnetzes in Abbildung 108, “Automatische Spezialisierung von `move-an-object`.,” auf Seite 237). Daher wird das für die hier betrachtete Stelle `has-moved-entity` erzeugte DS-Segment `dss-1` automatisch zu `spatial-action-dss` spezialisiert, wodurch wiederum die in Abbildung 122 bei der Konzeptdefinition angegebenen Assertionen etabliert werden. Es handelt sich also um ein DS-Segment, das ein Referenzsystem benötigt (`reference-frame-requiring-dss`) und in die Dialogstruktur eingebettet werden muß, da gilt: `(at-least 1 (k ds-component-of))`. Auch die Stellung im Diskurs wird aus diesen Informationen automatisch hergeleitet (Füller für `has-discourse-purpose`).

In der dritten Konzeptdefinition aus Abbildung 122 werden ähnliche Inferenzschritte für Auswahlhandlungen formalisiert.

```
(defun create-ds-node (concept-name &optional parent-node concept-expr)
  (let ((new-node (create-ind (gentemp (symbol-name concept-name))
                              `(and ds-node . ,(if concept-expr
                                                   (list concept-expr)
                                                 nil)))))
    (unless (null parent-node)
      (cl-ind-add new-node `(fills ds-component-of ,(cl-name parent-node))))
    (cl-ind-add new-node concept-name)))

(define-method compute-role-fillers ((ind spatial-action-dss)
                                     (role-name (eql 'ds-component-of))
                                     (model hamvis-model-class))
  (list (create-ds-node 'dsa nil `(fills has-nucleus
                                         ,(cl-name ind)))))

(define-method compute-role-fillers ((ind selection-action-dss)
                                     (role-name (eql 'ds-component-of))
                                     (model hamvis-model-class))
  (let* ((place (represents-place ind))
         (pane-type (place-required-pane-type place))
         (dsa (create-ds-node 'dsa
                              nil
                              `(and (fills has-nucleus
                                           ,(cl-name ind))
                                    (fills required-pane-type
                                           ,pane-type)))))
    (list dsa)))
```

Abbildung 123. Methoden zur Berechnung von Oberknoten (DS-Aggregate) von DS-Segmenten.

Für die Berechnung von Füllern der Rolle `ds-component-of` der hier skizzierten DS-Segmente sind für die hier angeführten Beispiele wiederum entsprechende Methoden im HAMVIS-Grundmodell definiert (Abbildung 123).

Mit den Methoden wird also auch ein Objekt auf der Aggregatebene erzeugt (Konzept `dsa`), wobei der Segmentknoten, der die Instantiierung bewirkt hat, als Füller der Unterrolle `has-nucleus` eingesetzt wird. Die Methode für `selection-action-dss` sorgt auch dafür, daß die Informationen über den Typ des zur Darstellung zu verwendenden Fensters in der Dialogstruktur nach oben auf die Aggregatebene propagiert wird (siehe den entsprechenden `fills`-Term für die Rolle `required-pane-type`). Eine Methode generiert also neben den entsprechenden Individuen auch weitere Assertionen. Man beachte, daß diese Assertionen nicht vordefiniert in der statischen Wissensbasis stehen können, sondern dynamisch berechnet werden müssen, da nicht a priori bekannt ist, wieviele DS-Segmente zur Beschreibung einer Anwendung benötigt werden. Die mit den Methoden aus Abbildung 123 berechneten Assertionen sorgen für eine „Propagierung“ von Informationen aus der Segmentebene in die Ebene der Modifikationsaggregate.

Wissen über Modifikationsaggregate ist ebenfalls in Form von definierten Konzepten im HAMVIS-Grundmodell deklariert. Abbildung 124 zeigt einen Ausschnitt der Wissensbasis zur Modifikation von Nuklei mit Referenzsystemen.

```
(define-primitive-concept reference-frame-dss dss)

(define-primitive-role has-reference-frame-satellite has-satellite)

(define-primitive-concept dsa-with-reference-frame-satellite
    (and dsa
         (at-least 1 has-reference-frame-satellite)
         (all has-reference-frame-satellite
              (and reference-frame-dss
                   (fillers has-discourse-purpose reference-frame-modifier)
                   (all describes-displayed-object spatial-object))))
  (:asserted-concepts (at-least 1 (k has-reference-frame-satellite))
                      (at-least 1 (k ds-component-of
                                     / (k has-nucleus)
                                       (k has-reference-frame-satellite)))))

(define-concept dsa-with-reference-frame-requiring-nucleus
    (and dsa
         (all has-nucleus reference-frame-requiring-dss))
  (:asserted-concepts dsa-with-reference-frame-satellite))


(define-concept dsa-with-selection-dss
    (and dsa
         (at-least 1 (k has-ds-component))
         (all has-ds-component selection-action-dss))
  (:asserted-concepts (at-least 1 (k ds-component-of))))
```

Abbildung 124. Modifikationsaggregatkonzepte (siehe Text).

Bei der Besprechung des Wissensbasisausschnitts in Abbildung 122 haben wir gesehen, daß für das DS-Segment der betrachteten Stelle `has-moved-entity` durch die Herleitung von `spatial-action-dss` auch die Konzepteinschränkung `reference-frame-requiring-dss` hergeleitet wird. Für das oben erzeugte Aggregat sind damit die hinreichenden Bedingungen des Konzeptes `dsa-with-reference-frame-requiring-nucleus` aus Abbildung 124 erfüllt. Durch eine Regel wird das Konzept `dsa-with-reference-frame-satellite` assertiert. Die entsprechende Konzeptdefinition von `dsa-with-reference-frame-satellite` in Abbildung 124 zeigt, daß ein Rollenfüller für die Rolle `has-reference-frame-satellite`, einer Unterrolle von `has-satellite`, berechnet werden muß (Marke 1). Weiterhin muß der Oberknoten bestimmt werden, sofern der Nukleus und der Füller von `has-reference-frame-satellite` berechnet sind (Marke 2), d.h. das Aggregat wird erst nach erfolgreicher Bestimmung des Modifikators (Referenzsystem) in die Dialogstruktur eingehängt und propagiert ggf. Informationen in die nächsthöhere Ebene.

Das letzte Konzept aus Abbildung 124 illustriert, daß auch für Aggregate, die für Auswahlhandlungen gebildet werden, die Integration in die Dialogstruktur erfolgen kann. Für die einzelnen Modifikationskonzepte sind wiederum entsprechende Methoden vordefiniert (siehe Abbildung 125).

```
(define-method compute-role-fillers ((ind dsa-with-reference-frame-satellite)
                                     (role-name
                                     (eql 'has-reference-frame-satellite))
                                         (model hamvis-model-class))
  (values (list (create-ds-node 'dss ind))
          nil
          '("New reference frame.")
          nil))

(define-method compute-role-fillers ((ind ds-node)
                                     (role-name (eql 'ds-component-of))
                                     (model hamvis-model-class))
  (declare (ignore ind role-name model))
  (values nil
          nil
          nil
        (list "Not enough information available for dialog structure node while comput\
ing a supernode.")))

(define-method compute-role-fillers
     ((ind dsa)
      (role-name (eql 'ds-component-of))
      (model hamvis-model-class))
  (let* ((pane-type (or (required-pane-type ind)
                        'hamvis-display-pane))
         (dsu (create-ds-node 'dsu
                              nil
                              `(fills required-pane-type
                                      ',(cl-name pane-type)))))
    (values (list dsu) nil '("New pane allocated.") nil)))
```

Abbildung 125. Weitere Methoden für die Funktion `compute-role-fillers` zur Vervollständigung von Instanzen.

Durch die erste Methode wird die Dialogstruktur in diesem Falle nach unten expandiert. Es wird ein Knoten für einen Referenzrahmen als Füller der Rolle `has-reference-frame-satellite` erzeugt. Es werden für das erzeugte Individuum keine Zusatzbedingungen angegeben (zweiter Wert ist gleich nil). Die beiden weiteren Werte von `compute-role-fillers` sind für Erklärungstexte gedacht, die an der Oberfläche angegeben werden. Der dritte Wert liefert eine positive Erklärung und der vierte eine negative, die verwendet wird, falls kein Rollenfüller bestimmbar ist. Dieses ist z.B. in der zweiten Methode der Fall. Sie dient als Fänger, falls speziellere Methoden nicht gefunden werden können. Die dritte Methode für `compute-role-fillers` erzeugt eine DS-Einheit (für ein Fenster) und fügt zu dieser die auf der Aggregatebene bekannten Informationen hinzu.

Die hier vorgestellten Dialogstrukturierungskonzepte und -methoden ermöglichen den Aufbau der in Abbildung 116 gezeigten Dialogstruktur für die Integration der Stelle `has-moved-entity`. In der Abbildung 126 wurde jetzt die Stelle `choice-set` ebenfalls in die Dialogstruktur integriert. Es wurde eine Dialogstruktureinheit zur Beschreibung der Informationen für ein weiteres Teilfenster erzeugt. Modifikationsbedarf für `choice-set` besteht in diesem Falle nicht (vgl. die Definition des entsprechenden Aggregats `dsa-with-selection-dss` in Abbildung 124).

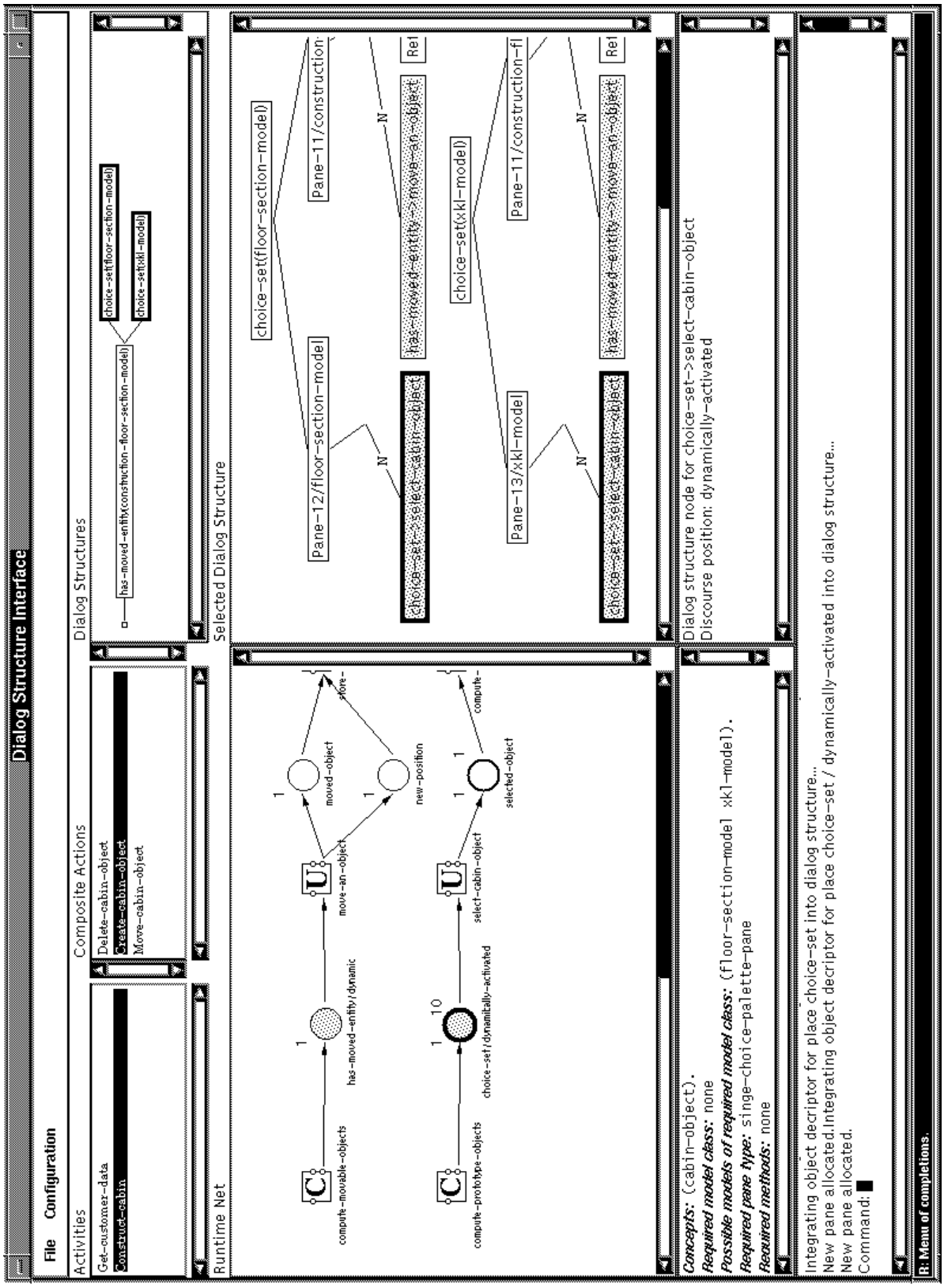


Abbildung 126. Integration der Stelle `choice-set` in die Dialogstruktur. Im Fenster `Dialog Structures` werden die sich aus den möglichen Modellen ergebenden Varianten präsentiert.

Für die Stelle `choice-set` sind aufgrund der Bedingungen der Aktion `select-cabin-object` die beiden Modelle `xkl-model` und `floor-section-model` für die Darstellung geeignet. Für jede der möglichen Modelle wird eine Dialogstrukturvariante verwaltet. Damit ist die Bearbeitung der Stelle `choice-set` beendet und der Entwickler kann Schritt für Schritt auch die anderen Stellen, in denen zur Laufzeit zu präsentierende Objekte auftreten, in die Dialogstruktur integrieren. Wenn alle Stellen berücksichtigt sind (dieses wird in der Oberfläche angezeigt), ist die Phase der Dialogstrukturierung beendet.

3.3.6 Varianten in der Dialogstrukturierung

Für jedes mögliche Modell, das für die Stelle ermittelt wurde (siehe das entsprechende Informationsfenster in Abbildung 126), wird als „Nachfolger“ der *aktuellen* Dialogstruktur eine Variante erzeugt. Die Dialogstrukturierungsoberfläche zeigt die jeweils aktuellen Varianten an. Eine Variante wird durch den Term *`Stellenname`*`(`*`Modellname`*`)` gekennzeichnet. Die markierten Varianten (dicker Rand) werden detaillierter im darunterliegenden Fenster angezeigt.

Interaktive Deaktivierung von Varianten

Der HAMVIS-Benutzer kann einzelne Varianten (durch Mausklick) deaktivieren. Deaktivierte Varianten werden bei der weiteren Expansion der Variantenstruktur nicht mehr betrachtet. In Abbildung 127 wurde beispielsweise die Variante `choice-set(xkl-model)` deaktiviert.

Für den weiteren Aufbau der hier betrachteten Beispieldialogstruktur wird also die graue Variante nicht mehr betrachtet.[104] Durch interaktive Deaktivierung von Varianten kann Wissen des Schnittstellenentwicklers die Gestaltung der Dialogstruktur einfließen. Für HAMVIS ist in diesem Fall allerdings nicht transparent, warum die mögliche Variante deaktiviert wurde. In einigen Fällen werden aber Varianten auch durch HAMVIS deaktiviert, wenn bestimmte, zur Präsentation notwendige Objekte nicht aus den deklarierten Domänenmodellen abgeleitet werden können.

104. Sie wird auch im Fenster zur detaillierten Darstellung von Dialogstrukturen nicht mehr gezeigt. Auf einen Farbbildschirm werden deaktivierte Varianten im Fenster `Dialog Structures` als gelber Knoten gezeigt.

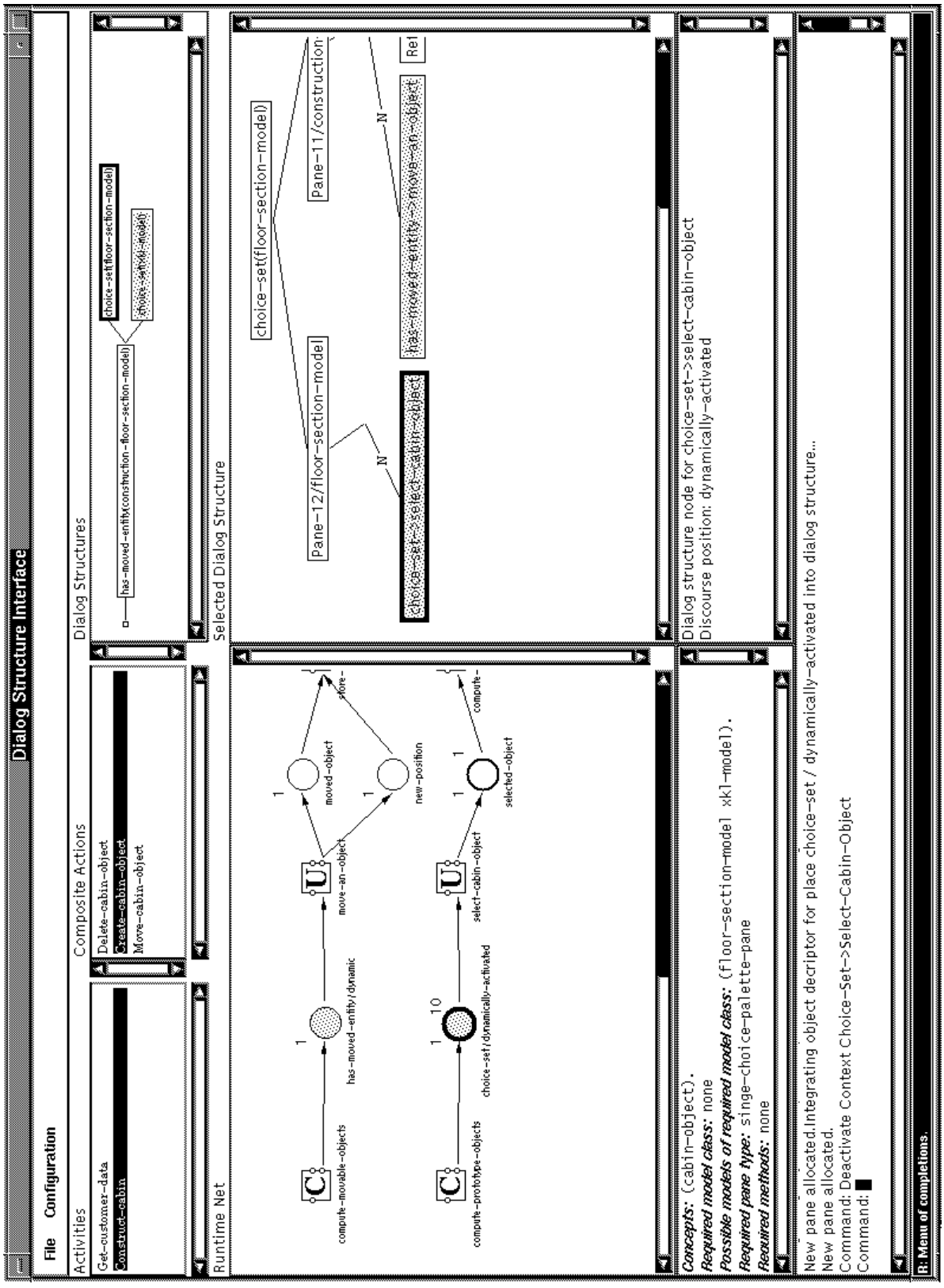


Abbildung 127. Deaktivierung der Variante, in der Objekte der Stelle `choice-set` aus der Sicht `xkl-model` dargestellt werden.

Deaktivierung von Varianten durch Unvollständigkeit der Modifikation

Varianten werden automatisch deaktiviert, wenn Modifikatoren aufgrund von fehlenden Informationen im betrachteten Domänenmodell nicht hergeleitet werden können. Falls dieser Fall eintreten sollte, so muß das jeweils betrachtete Domänenmodell durch den Schnittstellenentwickler angepaßt werden. Der nächste Abschnitt erläutert, wie Inferenzen über Konzepte, die in Domänenmodellen deklariert sind, zum weiteren Aufbau der Dialogstruktur ausgenutzt werden können.

3.3.7 Vollständigkeit der Modifikation

Zur Illustration möchte ich das Beispiel des Referenzsystems weiterführen. Für Referenzsysteme sind im HAMVIS-Grundmodell noch speziellere definierte Konzepte zur detaillierten Beschreibung vorgesehen. Ein Beispiel ist hier in Abbildung 128 widergegeben.

```
(define-primitive-role has-static-reference-object nil)

(define-concept reference-frame-modifier-dss
    (and reference-frame-dss
         (at-least 1 has-discourse-purpose)
         (fillers has-discourse-purpose reference-frame-modifier))
  (:asserted-concepts (at-least 1 (k has-static-reference-object))
                      (at-most 1 has-reference-object)))
```

Abbildung 128. Weitere Konzepte zur Vervollständigung der Referenzrahmenmodifikation.

Das Konzept erzwingt die Bestimmung eines Rollenfüllers für die Rolle `has-static-reference-object`, sofern der Diskurszweck `reference-frame-modifier` für ein DS-Segment vom Konzept `reference-frame-dss` bekannt wird. Die korrespondierende Methode für den K-Operator ist in Abbildung 129 definiert.

```
(define-method compute-role-fillers ((ind reference-frame-dss)
                                     (role-name
                                      (eql 'has-static-reference-object))
                                     (model geometry-model-class))
  (let ((reference-object (find-reference-frame-in-odg
                           (modified-nucleus ind)
                           model)))
    (if reference-object
        (values (list reference-object)
                nil
                (list (format nil "Reference object ~S found in ODG."
                                  reference-object))
                nil)
       (values nil
               nil
               nil
               (list (format nil
                             "Reference frame which must be available at
 development time cannot be found while traversing ODG for model ~S."
                             model))))
```

Abbildung 129. Methode zur Suche von geeigneten Referenzrahmen in dem Objektzerlegungsgraphen des entsprechenden Modells.

Ein Referenzsystem muß zur Entwicklungszeit als Individuum aus den Konzeptdeklarationen ermittelbar sein, da es gezeichnet werden muß, bevor z.B. konkrete Kabinenobjekte als Individuen berechnet werden können. Die Bestimmung des Referenzsystems hängt von den konzeptuellen Informationen über den Nukleus ab. Die notwendigen Ableitungsschritte sind in Abbildung 129 als Funktion `find-reference-frame-in-odg` repräsentiert. Die internen Ableitungen zur Realisierung dieser Funktion werden im nächsten Abschnitt skizziert.

Interne Ableitungen: Verwendung des Objektzerlegungsgraphen

Zur Herleitung eines Referenzsystems betrachtet HAMVIS die partonomische Zerlegung von Objekten. In diesem Kontext sind die in Kapitel 3.1 eingeführten Begrifflichkeiten relevant (siehe Kapitel 3.1.8 über „Partonomische und taxonomische Konzepthierarchien“). HAMVIS bestimmt zu jedem Modell einen sogenannten Objektzerlegungsgraphen. Ein Objektzerlegungsgraph macht die partonomische Zerlegung von visualisierbaren Konzepten in visualisierbare Konzepte explizit. Siehe hierzu die Definitionen der Begriffe Basiskonzept und Primärkonzept zur modellspezifischen Charakterisierung von Konzepten sowie auch den Begriff der direkten Visualisierbarkeit in Kapitel 3.1.

Im Falle der verschiebbaren Objekte aus der Stelle `has-moved-entity` handelt es sich um Objekte vom Konzept `cabin-object`. Das in der Dialogstrukturvariante betrachtete Modell ist das Modell `construction-floor-section-model`. In diesem Modell ist `cabin-object` ein kategoriales Oberkonzept. Zu diesem Konzept werden nun die subsumierten Primärkonzepte bestimmt (ein einfacher beschreibungslogischer Schluß): `lavatory`, `galley`, `seat`, usw. Anschließend wird im Objektzerlegungsgraph der „kleinste“ gemeinsame Oberknoten ermittelt. Durch Traversierung des Objektzerlegungsgraphen ergibt sich i.a. eine Rollenkette mit Rollen der partonomischen Zerlegung von Objekten aus dem HAMVIS-Grundmodell (`spatially-enclosed-by`, `form-representation-constituent-of` usw.). In dem hier betrachteten Fall muß eine einzige Rolle „verfolgt“ werden: `spatially-enclosed-by`. Der gemeinsame Oberknoten im Objektzerlegungsgraphen selbst liefert die notwendigen konzeptuellen Informationen für das in der Zerlegungshierarchie übergeordnete Objekt. In dem hier betrachteten Beispiel handelt es sich um eine Instanz von `cabin-body`.

Über die Rollenkette kann nun das Referenzsystem ausgehend von einem `construction-floor-section-model:cabin-object` bestimmt werden. Zu beachten ist, daß schon aus der Konzeptbeschreibung von `construction-floor-section-model:cabin-object` abgeleitet werden muß, daß *alle* jemals erzeugten Instanzen in dem *gleichen* Flugzeugrumpf liegen! Genauer gesagt: für alle Objekte, die zur Laufzeit in der Stelle `has-moved-entity` auftreten, muß das gleiche Referenzsystem verwendet werden. Intuitiv mag dieses zwar klar sein: Man konstruiert die Inneneinrichtung genau eines Flugzeugs. Für die automatische Oberflächengestaltung muß dieses allerdings in jedem Fall in einem Domänenmodell explizit gemacht werden.

Ich möchte hier noch einmal die Konzeptdefinition für `cabin-object` aus dem Modell `construction-floor-section-model` (siehe Abbildung 77) betrachten:

```
(define-concept-refinement (cabin-object xkl-model)
    (and (all has-projective-form xy-projective-form)
         (at-most 1 has-projective-form)
         (at-least 1 has-projective-form)
         (at-most 1 spatially-enclosed-by)
         (fillers spatially-enclosed-by uac-cabin-body))
  (:categorial-superconcept-p t)
  (:asserted-concepts non-overlapped-spatial-object))
```

Es ist also schon auf der konzeptuellen Ebene modelliert, daß alle Kabinenobjekte über die Relation `spatially-enclosed-by` zu genau einem „Oberobjekt“ `uac-cabin-body` in Beziehung stehen.

Man beachte, daß diese Information nicht unbedingt direkt in der Konzeptdefinition zu stehen braucht, sondern über beschreibungslogische Inferenzen auch abgeleitet werden könnte. HAMVIS verwendet die entsprechenden Erfragefunktionen für Rollenfüllereinschränkungen, die durch CLASSIC bereitgestellt werden.

Sollte allerdings keine entsprechende Einschränkung für die Füller hergeleitet werden können, so kann (zur Entwicklungszeit) kein Referenzsystem bestimmt werden (die Funktion `find-reference-frame-in-odg` liefert eine leere Menge), und die Modifikation bleibt unvollständig, d.h. eine Modifikationsanforderung kann bei den gegebenen Definitionen für das Domänenmodell nicht erfüllt werden. Die entsprechende Dialogstrukturvariante wird deaktiviert und es wird automatisch ein entsprechender Erläuterungstext mit der Variante assoziiert (siehe Abbildung 129), der in der interaktiven Oberfläche angezeigt werden kann.

Externe Präsentation: Anzeige an der Oberfläche

In Abbildung 130 wird die Diskursstruktur nach Integration der Stelle `has-moved-entity` gezeigt. Bei der obigen Definition für `cabin-object` im Modell `construction-floor-section-model` läßt sich ein konkretes Referenzsystem zur Entwicklungszeit nicht ermitteln, wenn die Zeile `(fillers spatially-enclosed-by uac-cabin-body)` fehlt oder nicht durch Subsumtionsschlüsse hergeleitet werden kann.

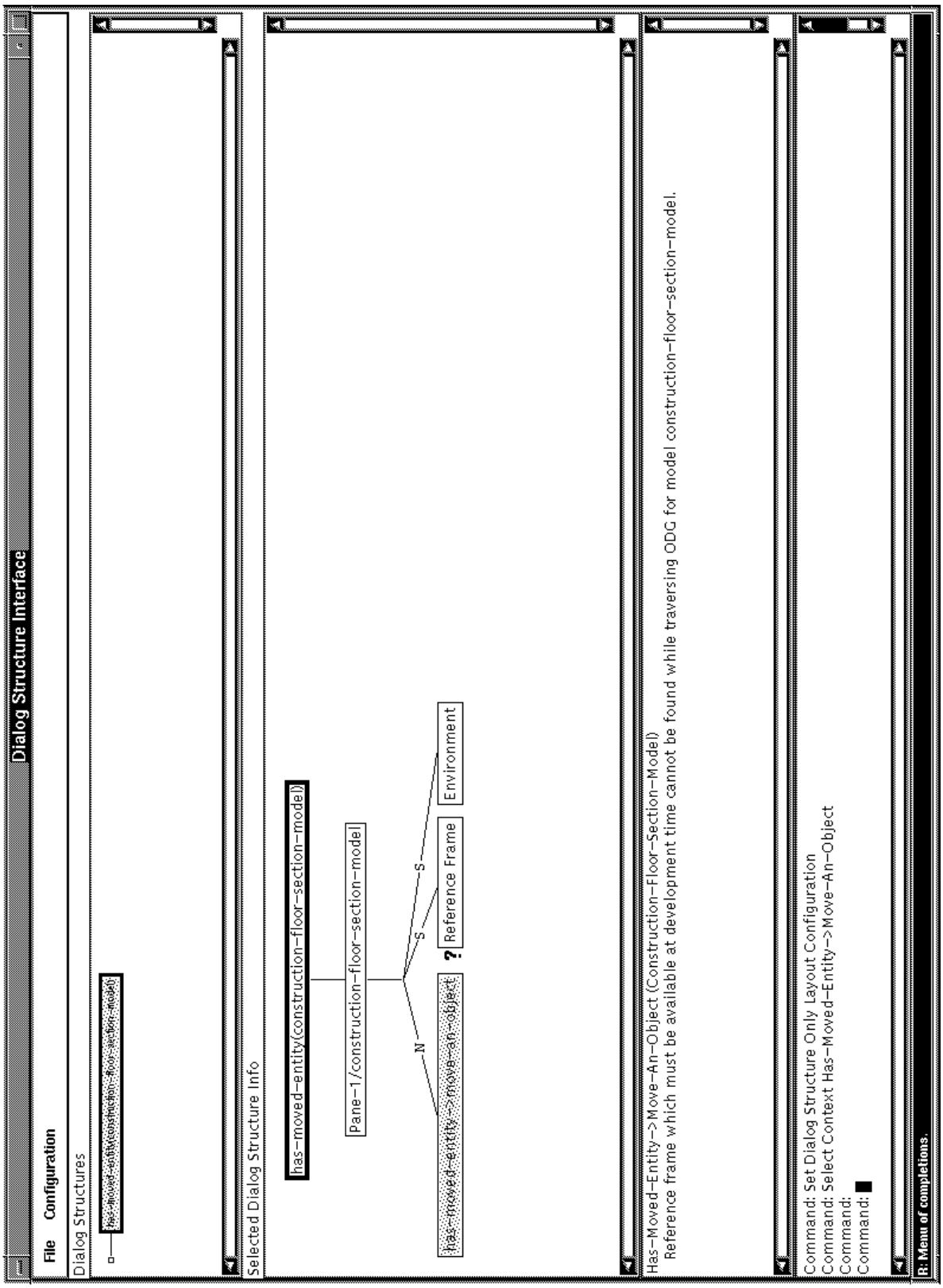


Abbildung 130. Fehlgeschlagene Suche nach einem notwendigen Referenzsystem führt zur Deaktivierung der betreffenden Dialogstrukturvarianten.

In diesem Fall gibt es keine weitere Dialogstrukturvariante mehr, in die die restlichen Stellen integriert werden könnten. Es wird ein Iterationsschritt bei der Entwicklung der Domänenmodelle notwendig.

Nehmen wir für die weitere Diskussion an, das Domänenmodell `construction-floor-section-model` sei so definiert, daß der entsprechende Füller für `has-static-reference-frame` für das DS-Segment bestimmt werden kann (siehe die Methodendefinition in Abbildung 129). Das für die Modifikation der bewegbaren Objekte in Abbildung 116 eingeführte DS-Segment `Reference Frame` ist damit vollständig spezifiziert. Die Definitionen aus dem HAMVIS-Grundmodell, die hier als Abbildungen 128 und 129 wiedergegeben sind, ermöglichen die Bestimmung eines konkreten Referenzsystems durch Betrachtung des Objektzerlegungsgraphen des betreffenden Modells der modifizierten Dialogstruktureinheit (in diesem Falle `construction-floor-section-model`). Wir nehmen also an, daß es eine aktuelle DS-Variante gibt, die durch Integration weiterer Stellen „expandiert“ werden kann.

3.3.8 Interaktive und automatische Generierung von DS-Varianten

Optionale Modifikationsobjekte

Die durch die oben skizzierten Schlüsse bestimmte Instanz `uac-cabin-body` ist sicherlich nur eine Möglichkeit, ein Referenzsystem zu festzulegen. Bei Betrachtung des Objektzerlegungsgraphen aus Abbildung 77 wird deutlich, daß auch dieses Objekt noch Bestandteil eines weiteren Objekts ist. Statt des Kabinenrumpfes hätte auch das ganze Flugzeug als Referenzsystem vorgesehen werden können. Das bedeutet, HAMVIS könnte zur Repräsentation dieser Möglichkeit eine weitere Dialogstrukturvariante „aufspannen“. Aus technischer Perspektive ist dieses mit den vorgestellten Abstraktionen leicht möglich.

Ich möchte diese Varianten aus Gründen der Übersichtlichkeit in dieser Arbeit jedoch nicht weiter betrachten und mit der Integration einer weiteren Stelle in die Dialogstruktur fortfahren.

Nachfolgevarianten in der Dialogstruktur

Nehmen wir an, die nächste Stelle, die der Entwickler in die Dialogstruktur integrieren möchte, sei `has-reference-object`, eine Eingangsstelle der Aktionstransition `localize-cabin-object` (vgl. Abbildung 115). Bei der Integration dieser Stelle werden ähnliche Ableitungen durchgeführt wie für die Stelle `has-moved-entity`. Auch für diese Aktion wird ein Referenzsystem benötigt. Weiterhin müssen die Objekte in der Umgebung präsentiert werden usw. In Abbildung 131 wird die von HAMVIS abgeleitete Variante im oberen Teil des Fensters für den Dialogstrukturgraphen dargestellt. Es werden in dieser Variante drei Darstellungsfenster vorgesehen. Ein Fenster stellt die zur Auswahl stehenden Objekte für die Lokalisierungshandlung dar (Pane-12 vom Typ `single-choice-palette-pane`, siehe Abbildung 115). Die für die Lokalisierung notwendigen Referenzobjekte werden in einem Graphikfenster präsentiert (Pane-14). Die Verschiebehandlung wird in einem weiteren Graphikfenster durchgeführt (Pane-11). Es wird für diese Handlung in dieser Dialogstrukturvariante also ein weiteres Referenzsystem benötigt. Auch für diese Variante wird der Kabinenrumpf (`uac-cabin-body`) benutzt. Neben der mit den oben beschriebenen Techniken abgeleiteten „Standardvariante“ wird noch eine weitere, automatisch von HAMVIS ermittelte Variante präsentiert.

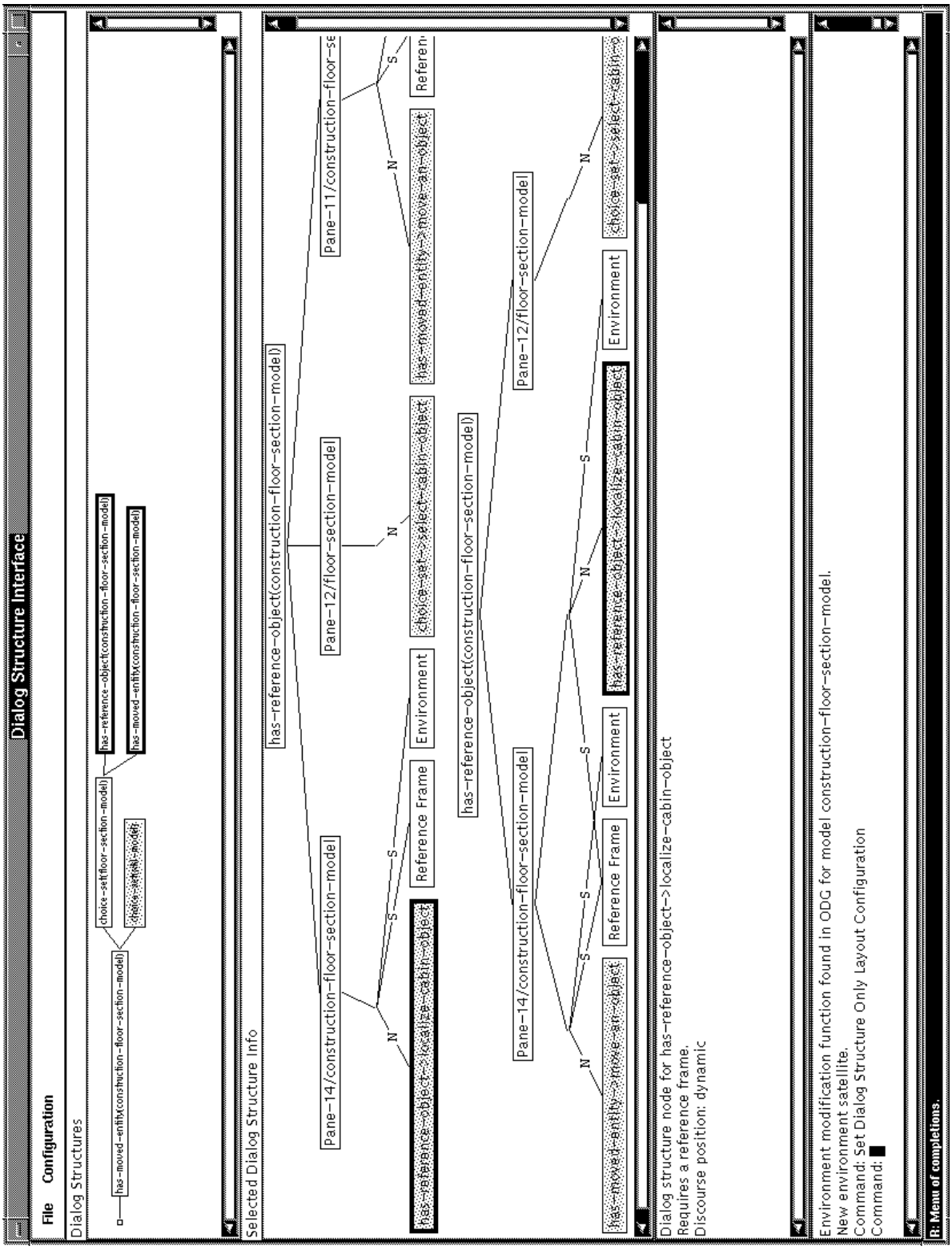


Abbildung 131. Ausschnitt aus der Oberfläche zur Dialogstrukturierung mit Anzeige des gefundenen statischen Referenzobjekts.

Verschmelzung von DS-Knoten

Zur Einsparung eines Graphikfensters können Dialogstruktursegmente z.B. „verschmolzen“ werden. In der zweiten Variante, die in Abbildung 131 gezeigt ist, wird nur noch ein einziges Graphikfenster benutzt, das jetzt beide Handlungen gleichzeitig unterstützt: die Lokalisierung und die Verschiebung. Die neue Variante wird von HAMVIS automatisch als Ableitung aus der Standardvariante aufgrund von „Designregeln“ ermittelt. Die HAMVIS-Wissensbasis enthält hierzu eine Menge von Funktionen zur Modellierung von Bedingungen zur Umgestaltung einer Dialogstruktur. Ein Beispiel ist die hier angeführte Zusammenfassung von Referenzsystemen durch Verschmelzung der entsprechenden Knoten in der Dialogstruktur. Ein anderes Beispiel wäre die „Mitbenutzung“ eines Referenzsystems aus einem anderen Fenster unter Hinzufügung von Layouteinschränkungen für die (relative) Anordnung der betroffenen Teilfenster (vgl. die Diskussion der Abbildung 3 in der Einleitung dieser Arbeit).

Die Begriffe „Designregel“, „Verschmelzung“ und „Umgestaltung“ sind durch Sichtweise der Dialogstruktur als Graph bedingt. Der nächste Abschnitt gibt eine formale Deutung dieser Begriffe auf logischer Ebene. Man beachte, daß sich hinter einer „Kante“ des Dialogstrukturgraphen, die in einer Visualisierung mit den entsprechenden Beschriftungen innerhalb der HAMVIS-Oberfläche erscheint, jeweils eine Menge von Aussagen verbergen. Siehe hierzu z.B. die Verwendung der Unterrollen `has-nucleus` und `has-satellite`.

3.3.9 Deutung der Variantenbestimmung auf logischer Ebene

Auf der logischen Ebene betrachtet, ist eine Dialogstruktur eine Menge von ABox-Aussagen, die aus einer Menge von Basisaussagen hergeleitet wurden. Neue Individuen werden durch K-Operatoren und die mit den jeweiligen Konzepten verbundenen Berechnungsmethoden eingeführt. Die Basisassertionen können auch als Annahmen gedeutet werden. Sofern die Annahmen inkonsistent sind oder unvollständig bleiben müssen, weil notwendiges Wissen sich nicht aus den bereitgestellten Domänenmodellen ableiten läßt, wird eine Variante deaktiviert. Aus einer konsistenten und vollständigen Menge von Aussagen einer Variante kann durch eine „Designregel“ eine neue Variante berechnet werden. Durch eine Variantendesignregel wird also eine Abbildung einer Menge von ABox-Aussagen in eine neue Menge von ABox-Aussagen definiert. Dabei sind folgende Aspekte zu berücksichtigen:

- Externe Herleitung von Gleichheitsaussagen

  Die implizite Annahme bei den K-Operator-Methoden, daß neue Individuen zu erzeugen sind, muß nicht unbedingt gerechtfertigt sein. Wie im Falle des Referenzsystems, kann als Rollenfüller innerhalb einer K-Operator-Methode auch ein schon vorher erzeugtes Individuum geliefert werden.

- Rücknahme von Assertionen

  Wenn eine Menge von Objekten in einem anderen Fenster dargestellt werden soll, so müssen die Aussagen, die ein Segment über die Ebene der Aggregate mit dem betreffenden DSU-Knoten in Beziehung setzen, wieder zurückgezogen werden. Die Verbindung zu dem neuen DSU-Knoten muß allerdings über eine Zusatzaussage hergestellt werden.

- Herleitung von Zusatzaussagen

  Mit der Verwendung eines schon erzeugten Individuums als Rollenfüller können Zusatzaussagen zur Einschränkung des Layouts der Darstellungsfenster (DSU-Ebene) einhergehen.

- Weglassen nicht mehr relevanter Assertionen

  Ein Knoten auf der DSU-Ebene, der nicht zu Modifikationsaggregaten und damit nicht zu DS-Segmenten in Beziehung steht, braucht nicht mehr betrachtet zu werden.

Designregeln liefern „Änderungen“ einer Variante bezüglich der oben aufgeführten Kategorien. Eine Funktion zur Realisierung einer Designregel traversiert die Dialogstruktur und prüft, ob bestimmte Bedingungen erfüllt sind. Sollte dieses der Fall sein, werden mehrere Werte zurückgegeben, die die oben skizzierten Umbildungsaspekte beschreiben. Die Designregel zur Referenzsystemverschmelzung beispielsweise sucht nach Referenzrahmen-Segmenten, die das gleiche Referenzobjekt (Rolle `has-static-reference-object`) beschreiben und Nachfolger von verschiedenen DSU-Knoten sind, für die das gleiche Modell vorgesehen ist. In der ersten Dialogstrukturvariante aus Abbildung 131 tritt eine solche Situation auf. Es wird zweimal das Referenzsystem `uac-cabin-body` in verschiedenen Teilfenstern verwendet.

Das für die Realisierung der Prototypimplementierung von HAMVIS verwendete beschreibungslogische Repräsentationssystem CLASSIC gestattet nicht die Repräsentation von verschiedenen „Welten“ oder „Kontexten“. Daher wurde zur Realisierung der Variantenverwaltung eine „Kopierfunktion“ implementiert, die eine Menge von ABox-Aussagen zur Beschreibung einer Dialogstruktur in eine neue Menge von ABox-Aussagen unter Berücksichtigung der Werte von Designregeln transformiert.

Nehmen wir an, der Schnittstellenentwickler entscheidet nach Analyse der in Abbildung 131 präsentierten Varianten, daß in diesem speziellen Falle innerhalb der XKL-Anwendung für die beiden Handlungen `move-an-object` und `localize-cabin-object` ein gemeinsames Fenster verwendet werden soll. Die erstere Variante, die zwei Fenster vorsieht, wird deaktiviert.

In Abbildung 132 ist ein weiteres Beispiel aufgeführt, das die hier vorgestellten Techniken illustriert. In dem Beispiel wurde jetzt die letzte noch in die Dialogstruktur zu integrierende Stelle `localized-object` berücksichtigt (vgl. auch Abbildung 115). HAMVIS ermittelt zwei Varianten, wovon wiederum die zweite mit Hilfe der oben skizzierten Umbildungsoperation ermittelt wurde.

Varianten können innerhalb des weiteren Verlaufs der Dialogstrukturentwicklung zu unterschiedlichen Gestaltungsformen der Oberfläche führen. Dieses wird besonders deutlich bei der Betrachtung von Verschmelzungsmöglichkeiten von Dialogstrukturknoten, die jeweils zu einer speziellen Umgestaltung einer Dialogstrukturvariante führen. Nicht in jeder Nachfolgevariante von zwei Ausgangsvarianten müssen die gleichen Verschmelzungsmöglichkeiten gegeben sein.

Mit dem Beispiel aus Abbildung 132 liegen zwei vollständige Varianten vor, in denen alle Stellen des Laufzeitsystems, die zu kommunizierende Objekte enthalten, berücksichtigt sind. Der HAMVIS-Benutzer kann sich für eine der Varianten entscheiden und mit der weiteren Gestaltung der graphischen Darstellung fortfahren. Nehmen wir an, er entscheidet sich für die untere Variante, die ein Graphikfenster und ein Fenster zur Darstellung einer Auswahlpalette vorsieht.

3.3.10 Umsetzung der abgeleiteten DS-Informationen in ein Laufzeitsystem

Die in der gewählten Dialogstruktur hergeleiteten Informationen zur Modifizierung von Segmenten können wiederum aus der Perspektive des Laufzeit-Petri-Netzes gedeutet werden. Nach Auswahl einer vollständigen Variante führt HAMVIS eine sogenannte „Erweiterung“ des Laufzeitnetzes durch.

Erweiterung des initialen Laufzeitnetzes um Modifikatortransitionen

Die Erweiterung des Laufzeitnetzes definiert die für die Generierung des Laufzeitsystems zu berücksichtigenden Modifikatorfunktionen einer ausgewählten vollständigen Dialogstrukturvariante. Jedes Modifikationssegment wird in eine neue Transition umgesetzt. Die Eingangsstellen sind jeweils durch die Nukleus-DS-Segmente definiert, die mit den grauen Ausgangsstellen im Laufzeitnetz korrespondieren. In Abbildung 133 wird ein Beispiel anhand der HAMVIS-Benutzungsoberfläche präsentiert.[105] Modifikatorfunktionen sind durch Rollenketten definiert (siehe die obigen Ausführungen zur Verwendung der Objektzerlegungsgraphen) und werden wie die anderen Transitionen zur Laufzeit evaluiert, sobald ihre Eingangsstellen mit konkreten Objekten belegt sind (Petri-Netz-Semantik). In den Ausgangsstellen werden die über die Rollenketten zur Laufzeit der Anwendung ermittelten Objekte plaziert. Diese Objekte werden dann ebenfalls an den Ausgabenverwalter zur Darstellung übermittelt. Das vorgesehene Darstellungsfenster geht aus der Dialogstruktur hervor.

105. Da die Modifikatortransitionen mit Berechnungs- und Speicherfunktionen vergleichbar sind, werden auch sie innerhalb des Fenster zur Anzeige des Petri-Netzes auch mit einem „C“ gekennzeichnet.

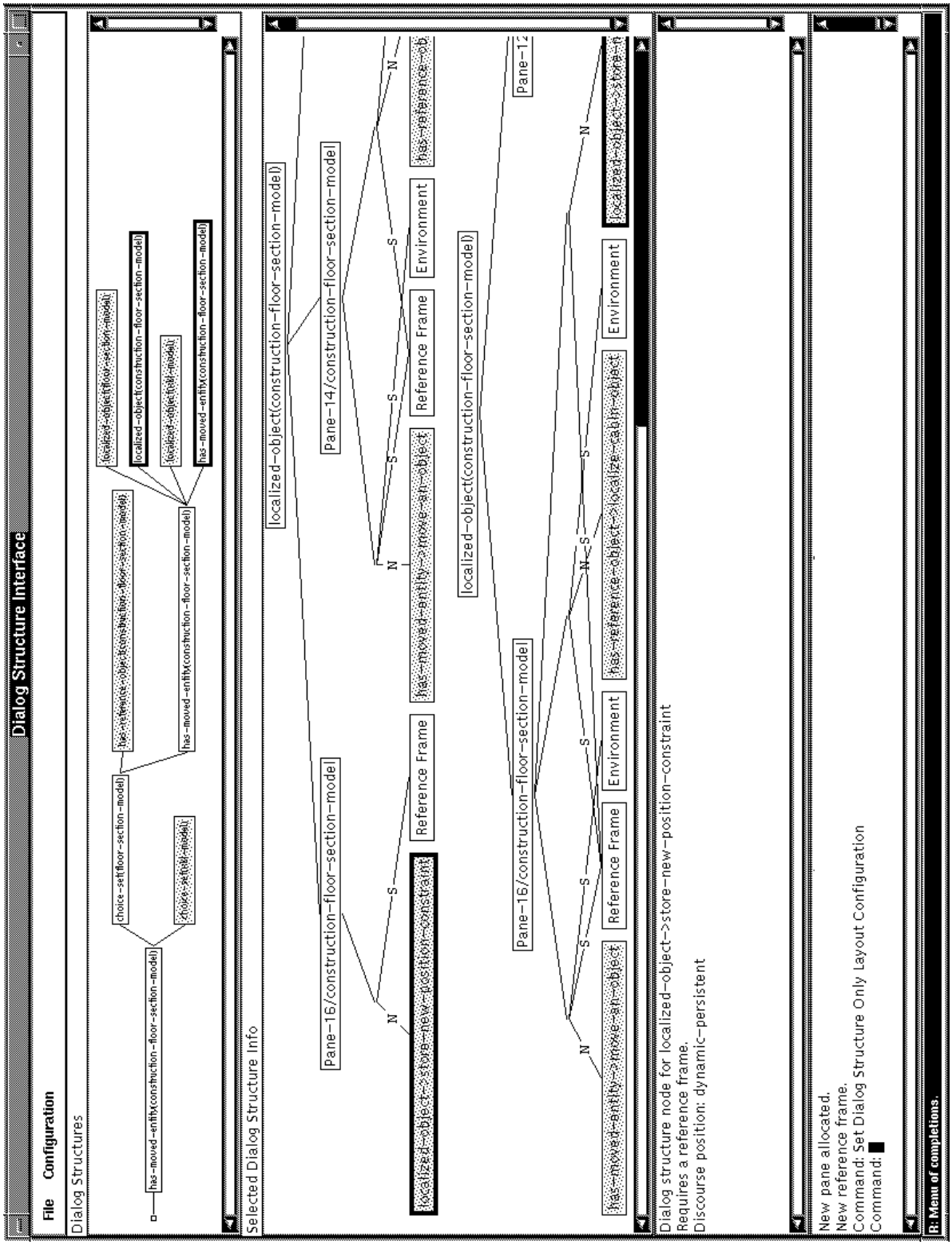


Abbildung 132. Weiteres Beispiel für die Variantengenerierung durch „Verschmelzung“ von Knoten.

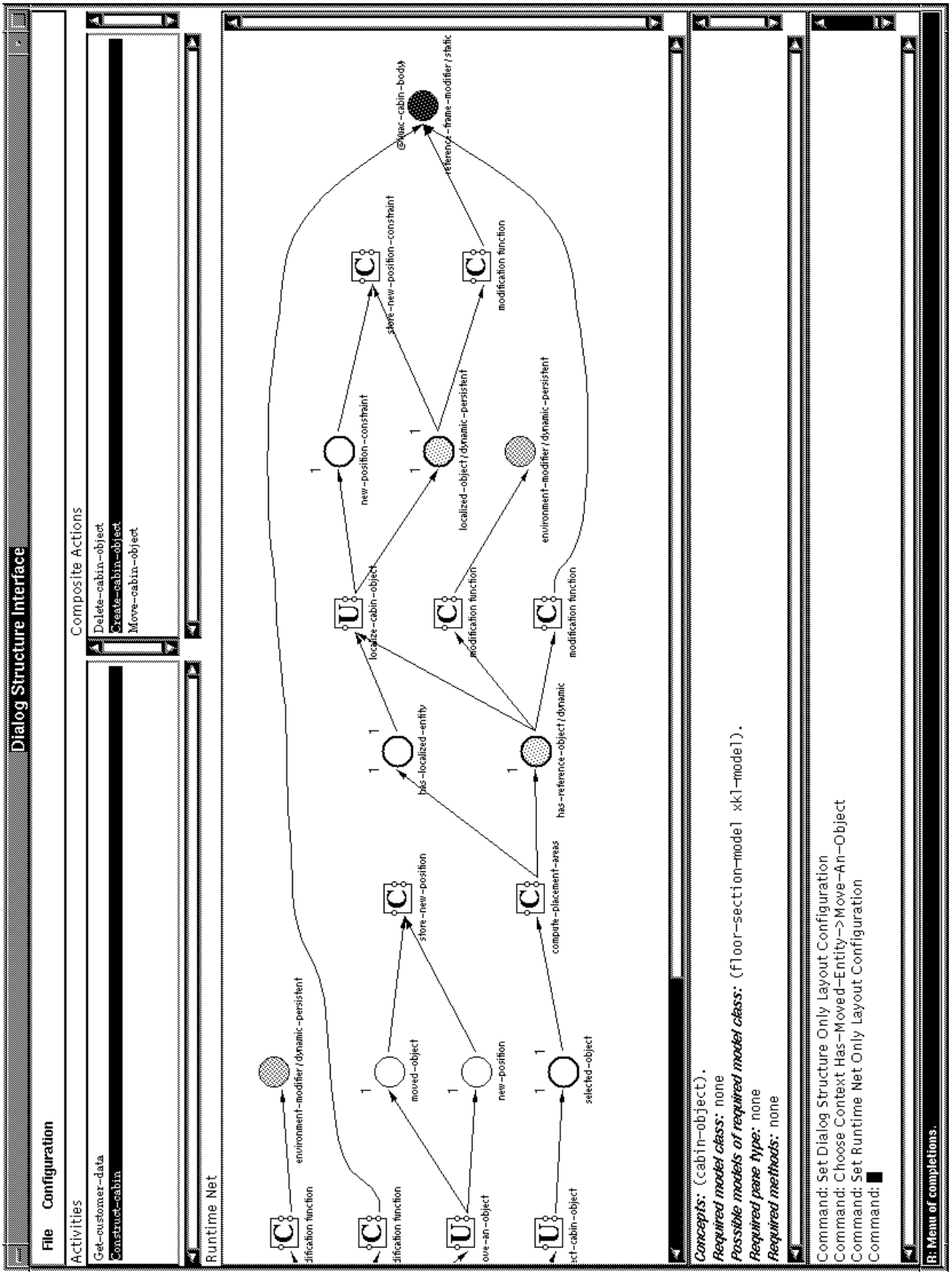


Abbildung 133. Erweitertes Laufzeitnetz, das die notwendigen Modifikationen aus der gewählten unteren Dialogstrukturvariante aus Abbildung 132 berücksichtigt.

Eine Besonderheit stellt die Stelle auf der rechten Seite dar. Der innerhalb des Interaktionszyklus bestimmte Rollenfüller `uac-cabin-object` ist schon a priori zur Entwicklungszeit bekannt.[106] Während die Teilnetze für die zusammengesetzten Aktionen `Move-cabin-object` und `Create-cabin-object` (siehe die Tabelle `Composite Actions`) in Abbildung 114 noch getrennt waren, sind sie jetzt in Abbildung 133 „zusammengewachsen“. Dieses ist eine direkte Folge der Ableitungen zur Ermittlung der Dialogstruktur. Die Aktionen können innerhalb eines gemeinsamen Fensters durchgeführt werden.

Das erweiterte Laufzeitnetz zeigt auch die abgeleiteten Informationen für den Ausgabenverwalter (siehe die Konzeptdefinitionen aus Abbildung 119). Bevor jedoch das Laufzeitsystem für die Anwendung generiert werden kann, müssen noch die Interaktionsgesten (wie z.B. Ziehen-und-Fallenlassen) zur Realisierung von zusammengesetzten Aktionen festgelegt werden.

3.3.11 Abbildung von zusammengesetzten Aktionen auf Interaktionsgesten

Durch die vom Systementwicklungsteam festgelegte Aktionenmodellierung ist die Teil-Ganzes-Struktur einer Anwendung aus Sicht der Aktionenzerlegung festgelegt. Für elementare Benutzeraktionen werden konzeptuelle Beschreibungen vorgegeben und zum Teil auch automatisch aus den Beziehungen zu den Parametern und Werten von Berechnungsfunktionen hergeleitet. Details hierzu wurden in Kapitel 3.2 ausführlich geschildert. Zur Zusammenfassung wird in Abbildung 134 noch einmal die Zerlegung von `move-cabin-object` und `construct-cabin-object` gezeigt.

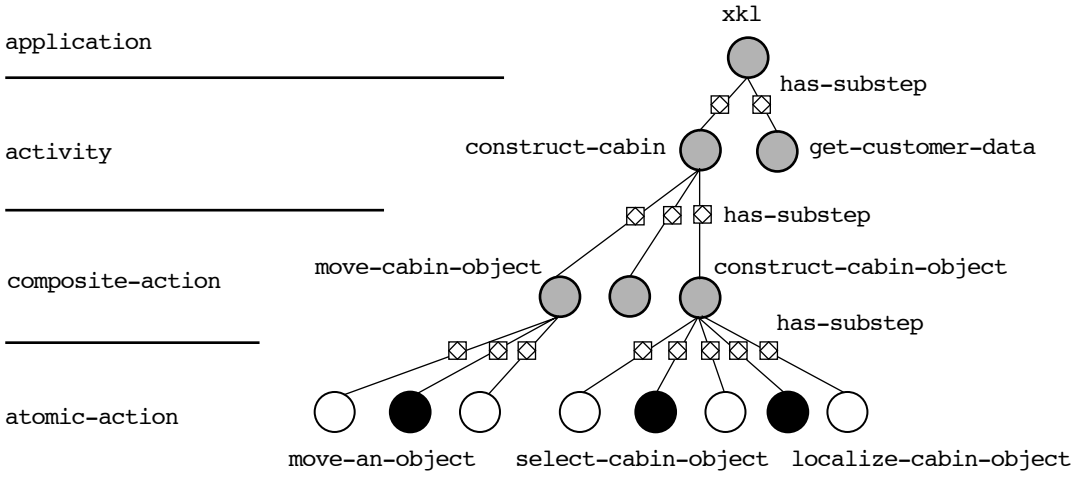


Abbildung 134. Teil-Ganzes-Struktur der XKL-Anwendung.

Zwei der Komponenten von `construct-cabin-object` sind elementare Benutzeraktionen (schwarze Kreise): `select-cabin-object` und `localize-cabin-object`. Beide Benutzeraktionen sind durch entsprechende Konzepte beschrieben. Für die Aktion `select-cabin-object` wurde das Konzept `selection-from-small-palette` und für `localize-cabin-object` wurde das Konzept `spatial-localization-in-xy-bounding-rectangle` festgelegt (siehe Kapitel 3.2).

106. In diesem Fall wird als obere Beschriftung nicht die minimale und maximale Kardinalität angezeigt, sondern die Füller werden direkt oberhalb der Stelle präsentiert.

Die Aktion `move-cabin-object` enthält nur eine elementare Benutzeraktion `move-an-object` vom Konzept `spatial-movement`.

Auf der Ebene der *zusammengesetzten* Aktion muß nun festgelegt werden, durch welche Interaktionstechniken die jeweils zugrundeliegenden elementaren Benutzeraktionen umgesetzt werden soll. Dabei spielen die jeweiligen Konzeptbeschreibungen der elementaren Teilaktionen ein entscheidende Rolle.

In HAMVIS wurden zwei Möglichkeiten zur Umsetzung von Benutzeraktionen durch Interaktionstechniken vorgesehen: eine direkte Umsetzung mit graphischen Interaktionswerkzeugen (interactors, gadgets, gauges) oder eine Umsetzung durch generelle Interaktionstechniken wie z.B. Direktes-Anklicken oder, etwas komplexer, Ziehen-und-Fallenlassen-Gesten. Zunächst einmal möchte ich auf die graphischen Interaktionswerkzeuge zu sprechen kommen.

Abbildung auf spezielle graphische Interaktionswerkzeuge

HAMVIS definiert für eine spezielle Anwendungsklasse mit Hilfe eines zugrundeliegenden UIMS eine Menge von Basis-Interaktionswerkzeugen. In Abbildung 135 wird ein ganz spezielles Verschiebewerkzeug gezeigt.

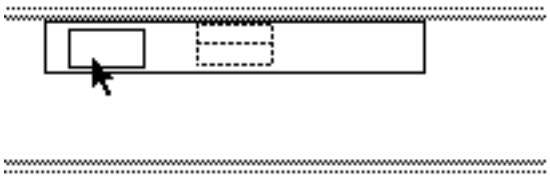


Abbildung 135. Direktmanipulative Festlegung einer Position innerhalb eines vorgegebenen Rechtecks, das die wählbare Position entsprechend einschränkt.

Ein Interaktionswerkzeug aus der Bibliothek wird im HAMVIS-Grundmodell mit einem Aktionenkonzept in Beziehung gesetzt. Auf die hierzu vorgesehenen syntaktischen Konstrukte möchte ich in diesem Zusammenhang nicht näher eingehen. Zu beachten ist, daß sehr viel mehr Aktionenkonzepte deklariert sein können als Interaktionswerkzeuge bereitgestellt werden. UIMS-Interaktionswerkzeuge für eine Anwendungsklasse sind also in diesem Sinne generisch verwendbar. Das in Abbildung 135 gezeigte Verschiebewerkzeug ließe sich z.B. auch für andere Aktionenkonzepte anwenden, die sich nicht auf räumliche Objekt beziehen, sondern z.B. zeitliche Aspekte betreffen. Durch eine solche Aktion würde dann also eventuell eine Terminverlegung mit dem gleichen Interaktionswerkzeug auf der UIMS-Ebene realisierbar. Ein anderes Beispiel wären Interaktionswerkzeuge, die an Armaturen (gauges) erinnern. Auch hierdurch können die verschiedensten höheren Konzepte für Benutzeraktionen realisiert werden.

Abbildung auf generelle graphische Interaktionstechniken

Nicht alle Aktionskonzepte werden durch generische Interaktionsbausteine aus der UIMS-Bibliothek umgesetzt. Eine Aktion `delete-cabin-object` (vgl. Abbildung 100) ließe sich durch einen einfachen Mausklick realisieren (auch hier sind noch Parameter wie z.B. Umschalttasten etc. festzulegen). Bekannt ist auch die Umsetzung von generellen Löschoperationen durch Ziehen-und-Fallenlassen-Mausgesten. Auch die beiden elementaren Teilaktionen `select-cabin-object` und `localize-`

cabin-object, die Bestandteil der zusammengesetzten Aktion create-cabin-object sind, lassen sich durch Ziehen-und-Fallenlassen realisieren. Durch die Wahl einer Interaktionsgeste lassen sich die Teilkomponenten einer zusammengesetzten Aktion zusammenfassen, so daß der Endbenutzer der Anwendung vielleicht die zugrundeliegenden Teilaktionen gar nicht wahrnimmt. Für die Inhaltsfestlegung und für die Bestimmung der Dialogstruktur ist jedoch die „tiefere“ Modellierung der zusammengesetzten Aktion create-cabin-object notwendig.

Die Abbildung von zusammengesetzten Aktionen auf Interaktionsgesten ist nicht trivial und HAMVIS kann zur Zeit nur einen kleinen Teil der möglichen Aktionskombinationen, die in einer anwendungsspezifischen Aktionenzerlegung auftreten können, auch durch Interaktionsgesten unterstützen, da hierzu auf der UIMS-Ebene, d.h. auf der Programmierebene, die entsprechenden Codeteile noch nicht in jedem Fall erzeugt werden können.

Im folgenden möchte ich die zugrundeliegende logische Formalisierung der Bestimmung von graphischen Interaktionstechniken durch Teil-Ganzes-Schlüsse auf der Ebene der zusammengesetzten Benutzeraktionen erläutern.

```
(in-model hamvis-upper-model)

(define-primitive-role has-subaction-1 has-substep)
(define-primitive-role has-subaction-2 has-substep)

(define-concept composite-action-that-can-be-realized-by-drag-and-drop
    (and composite-action
         (at-least 1 has-subaction-1)
         (at-most 1 has-subaction-1)
         (at-least 1 has-subaction-2)
         (at-most 1 has-subaction-2)
         (all has-subaction-1 action-with-focussing-output)
         (all has-subaction-2 action-with-focused-input)))
```

Abbildung 136. Illustration eines Konzeptes zur Beschreibung von zusammengesetzten Aktionen, die aus zwei speziellen elementaren Benutzeraktionen bestehen und sich daher durch eine Ziehen-und-Fallenlassen-Geste realisieren lassen.

Das definierte Konzept composite-action-that-can-be-realized-by-drag-and-drop aus Abbildung 136 wird hergeleitet, wenn zwei elementare Aktionen jeweils über die Rollen has-subaction-1 und has-subaction-2 mit einer zusammengesetzten Aktion in Beziehung gesetzt werden. Mit Konzepten, die Aktionenzerlegungen aus Sicht der Realisierung durch Interaktionsgesten beschreiben, sind jeweils Abbildungen auf UIMS-Dienste verbunden. Details hierzu werden in Kapitel 3.5 erläutert.

Da CLASSIC eine recht eingeschränkte Beschreibungslogik darstellt und z.B. definierte Rollen nicht unterstützt werden, müssen einige der notwendigen ABox-Aussagen „von außen“ kommen. HAMVIS traversiert also den Aktionenzerlegungsgraphen aus Abbildung 134 und setzt jede zusammengesetzte Aktion, die aus zwei elementaren Benutzeraktionen besteht, jeweils über die Relationen has-subaction-1 und has-subaction-2 mit den Teilaktionen in Beziehung. In unserem Beispiel werden also folgende ABox-Aussagen hinzugefügt:

```
(state (related create-cabin-object select-cabin-object has-subaction-1))
(state (related create-cabin-object localize-cabin-object has-subaction-2))
```

Durch beschreibungslogische Inferenzen wird in dem Beispiel für `create-cabin-object` hergeleitet, daß das obige Konzept gilt. Die Inferenzen sind jedoch nicht trivial und nehmen starken Bezug auf die Subsumtionsbeziehungen in der TBox. Die generellen Aktionskonzepte `selection` und `localization` aus Abbildung 95 assertieren jeweils die hinreichenden Bedingungen `action-with-focussing-output` und `action-with-focused-input`.

In der momentanen Version von HAMVIS werden nur einige der möglichen Abbildungen auf Interaktionsgesten unterstützt. Die implizite Beschreibung der Reihenfolge von Aktionen durch die Nummern in den Namen der Unterrelationen von `has-substep` ist sicherlich nicht die beste Lösung. Wegen der eingeschränkten Ausdrucksmächtigkeit von CLASSIC wurde an dieser Stelle in HAMVIS eine pragmatische Lösung gewählt. Der gleiche Weg wurde z.B. auch bei Di Eugenio verfolgt [70].

Für eine umfassendere Formalisierung sind insbesondere die Arbeiten von Devanbu und Litman relevant [68] (siehe auch die Umsetzung von Borgida in [25] sowie auch die Nachfolgearbeiten von Weida und Litman [334] und [335]). In ihrem System CLASP erweitern Devanbu und Litman den Subsumtionsbegriff von Beschreibungslogiken auf Zerlegungsbeschreibungen. Dieses wird insbesondere zur Modellierung von komplexen Aktionenzerlegungen benötigt, in denen auch die Reihenfolge der Teilaktionen berücksichtigt wird. Ich möchte hier die Basisidee an einem Beispiel schildern. Mit dem CLASP-System ließe sich das Konzept `composite-action-that-can-be-realized-by-drag-and-drop` wie folgt definieren. Die Rolle `plan-expression` in CLASP entspricht der Rolle `has-substep` aus HAMVIS.

```
(define-concept composite-action-that-can-be-realized-by-drag-and-drop
  (and composite-action
       (all plan-expression
            (sequence
               action-with-focussing-output
               action-with-focused-input))))
```

Anstatt also die Reihenfolge von Teilaktionen über Unterrollen (`has-subaction-1` und `has-subaction-2`) „von außen“ vorgeben zu müssen, gestattet CLASP die Definition der Zerlegung mit regulären Ausdrücken und ordnet die Teilkomponenten automatisch zu. Das Alphabet der Ausdrücke ist durch die Menge der Aktionenkonzepte gegeben. Wenn man reguläre Ausdrücke als formale Sprachen auffaßt, läßt sich eine mengentheoretische Semantik (Formale Sprachen als Wortmengen) definieren. Damit erhält man eine Möglichkeit, zwischen den Zerlegungsbeschreibungen durch reguläre Ausdrücke auch einen erweiterten Subsumtionsbegriff zu definieren. Eine Aktionenbeschreibung A_1 subsumiert eine andere Aktionenbeschreibung A_2, genau dann wenn die CLASSIC-Konzeptbeschreibung von A_1 die Beschreibung von A_2 subsumiert und wenn die Wortmenge, die durch den regulären Zerlegungsbeschreibungsausdruck von A_1 definiert ist, eine Obermenge der Zerlegungswortmenge von A_2 ist. Die hier vorgestellte erweiterte Subsumtionsbeziehung von CLASP zwischen Aktionenkonzepte wird in der TBox und in der ABox verwendet. Mit den skizzierten Mechanismen lassen sich Zerlegungen flexibler beschreiben. Zerlegungen, die für besondere Interaktionsgesten geeignet sind, lassen sich automatisch über Objektklassifikationen in der ABox bestimmen. Es wäre daher sehr interessant, das CLASP-System in den HAMVIS-Prototypen zu integrieren und damit die Grundlage zu schaffen für eine detailliertere Ausarbeitung der Formalisierung von Inferenzen zur Aktionenzerlegung und zur Realisierung von zusammengesetzten Aktionen durch komplexere Interaktionsgesten. Die realisierte Variante zeigt jedoch schon, daß Subsumtionsschlüsse im Prinzip auch hierfür verwendbar sind.

Auf der nächsthöheren Ebene in der Aktionenzerlegung liegen die Aktivitäten. Aktivitäten sind als Strukturierungsmittel der Applikation in Untereinheiten gedacht, die jeweils in einem separaten Interaktionskontext durchgeführt werden sollen. Üblicherweise wird hierfür auf der UIMS-Ebene ein Fenster verwendet. Zum Abschluß der Dialogstrukturierungsphase muß noch die Aktivierung von Aktivitäten festgelegt werden.

3.3.12 Festlegung der Aktivierung von Aktivitäten

Für die Aktivitäten einer Applikation müssen folgende Angaben gemacht werden:

- Festlegen der Startaktivität der Applikation
- Definition einer festen Reihenfolge von Aktivitäten oder Angabe von Aktivierungsbedingungen für die (asynchrone) Aktivierung von Aktivitäten zur Realisierung von komplexen Interaktionsformen.
- Hinzufügung von Endbedingungen bzw. Abbruchbedingungen für Aktivitäten.

Die Angaben können entweder mit einer textuellen Beschreibungssprache oder mit interaktiven Dialogen leicht erfaßt werden. Bedingungen lassen sich als Prädikate (u.U. definiert durch Lisp-Funktionen) beschreiben. Ich möchte an dieser Stelle nicht näher darauf eingehen.

3.3.13 Bestimmung des Layouts der Teilfenster einer Aktivität

Die in der Dialogstruktur repräsentierten Beziehungen zwischen Teildarstellungen und zwischen Komponenten von Teildarstellungen definieren Einschränkungen bezüglich der (relativen) geometrischen Anordnung von Teilfenstern in dem Gesamtfenster für eine Aktivität einer Anwendung. Auch die Klassen der vorgesehenen Teilfenster liefern weitere Hinweise für eine Plazierung von Teilfenstern.

Einschränkungen können relativ zwischen Teilfenstern gelten (siehe z.B. die Diskussion von Abbildung 3 mit der vertikalen Anordnung der Teilfenster in der unteren Teildarstellung), sie können aber auch relativ zum Gesamtfenster formuliert werden (z.B. die Plazierung einer Palette an der linken Seite eines Fensters). Letztere Einschränkungen sind u.U. vom Wirtssystem der Anwendung abhängig.

HAMVIS muß noch um eine Teilkomponente ergänzt werden, mit der das Layout der Teilfenster der Anwendung festgelegt wird. Das Ziel dieser Komponente ist die Herleitung einer Layoutbeschreibung, die für das verwendete Zielsystem geeignet ist. Eine solche Teilkomponente ist im aktuellen HAMVIS-Prototypen noch nicht realisiert, vgl. aber hierzu die im zweiten Kapitel erwähnten Arbeiten wie z.B. das System DON [148], einer Teilkomponente von UIDE.

3.3.14 Zusammenfassung: Dialogstrukturierung mit Beschreibungslogiken

Wir haben gesehen, daß die in Kapitel 3.3.1 aufgezeigten Dialogsstrukturierungsaufgaben mit Hilfe der interaktiven Oberflächen und der vorgestellten Wissensbasen bearbeitet werden können. Das Ergebnis der Dialogstrukturierungsphase ist eine sogenannte Dialogstruktur und ein daraus abgeleitetes „erweitertes“ Laufzeitnetz. Zur Erfassung von notwendigen und optionalen Modifikatoren kommen weitere Stellen und Transitionen hinzu. Die Stellen des erweiterten Laufzeitnetzes sind durch Attribute gekennzeichnet, die zur Beschreibung der Präsentation von Laufzeitobjekten benötigt wer-

den (Diskurszweck, Diskursposition, Darstellungsfenster). Aspekte des Designmodells einer graphischen Anwendung (im Sinne von Holz, siehe Kapitel 2.2.1) werden durch die Dialogstruktur bzw. durch die Dialogsstrukturvarianten explizit gemacht und können im Entwicklungsteam schon während der Entwicklungsphase einer Anwendung diskutiert werden.

Auch wenn die formalen Ableitungen zum Aufbau der Dialogstruktur für den Schnittstellenentwickler transparent sind, so hat sich doch der in diesem Kapitel aufgezeigte wissensbasierte Modellierungsansatz sehr bewährt. Durch die Verwendung von beschreibungslogischen Konzepten werden die Schlußfolgerungsprozesse auf formaler Ebene beschrieben. Die konsistente Verwendung der Beschreibungslogik ermöglicht es, Schlußfolgerungen über die Dialogstrukturierung mit Schlußfolgerungen über Aktionenkonzepte zu koppeln (vgl. die Diskussion zu Abbildung 122). Aktionenmodelle werden also im Gegensatz zu vielen anderen Ansätzen nahtlos in die weitere Oberflächenentwicklung integriert (vgl. Abschnitt Kapitel 2.2.3). Aus der in HAMVIS realisierten Aktionenmodellierung wird automatisch eine Beschreibung des Laufzeitsystem abgeleitet (Laufzeitnetz). Das Laufzeitsystem wird formal durch ein Petri-Netz repräsentiert und in interaktiven Oberflächen dem HAMVIS-Benutzer angezeigt. Es ist nicht nötig, dieses Laufzeitnetz separat zu erfassen (vgl. die in Kapitel 2.2.4 aufgeführten Ansätze). Die im Kontext von HAMVIS verwendete Granularitätsstufe der Modellierung vermeidet eine Explosion von Knoten (vgl. Kapitel 2.2.4). Auch ein Schließen über die Anwendbarkeit eines Modells wurde durch den Subsumtionsbegriff der Beschreibungslogik formalisierbar. Allerdings waren in diesem Zusammenhang Erweiterungen zur Charakterisierung von Konzepten nötig (siehe die eingeführten Begriffe Basiskonzept, kategoriales Oberkonzept usw.). Ich komme im Zusammenhang mit der Bestimmung von Präsentationsattributen hierauf zurück.

Obwohl für den Schnittstellenentwickler die Erweiterung der (internen) HAMVIS-Wissensbasen zur Dialogstrukturierung nicht vorgesehen ist, so ist dennoch eine leichte Erweiterbarkeit für den *HAMVIS-Entwickler* zu fordern. Die Wissensbasen können mit geringem Aufwand um weitere Konzepte zur Beschreibung von Modifikationsanforderungen und Konzepte zur Beschreibung von möglichen Modifikatoren erweitert werden. Ein Beispiel ist die Modifikation durch Umgebungsobjekte. Bedingt durch eine „räumliche Aktion“ wird für die Entscheidungsfindung nicht nur das manipulierte Objekt gezeigt, sondern auch Objekte in der Umgebung. Dieses können fest installierte Ausrüstungsgegenstände sein oder auch Objekte, die der Benutzer in vorigen Aktionen selbst plaziert hat. Ohne näher darauf einzugehen, habe ich in den oben diskutierten Beispielen die „Umgebungsmodifikation“ schon aufgeführt (siehe z.B. das Dialogstruktursegment Environment in Abbildung 116). Es wurden in Abbildung 119 auch einige Konzepte zur Beschreibung des jeweiligen Diskurszweckes `environment-modifier` eingeführt. Die Modifikationsart wurde durch Hinzufügung weniger Konzepte und Methoden zur Berechnung von Rollenfüllern sowie Methoden zur Darstellung von Informationen an der Oberfläche realisiert. Die Verwendung von generischen Funktionen für CLASSIC-Objekte (siehe [215]) erweist sich in diesem Kontext als extrem vorteilhaft, um eine klare, erweiterbare Software-Architektur zu erhalten.

Weitere Konzepte zur Modellierung von Dialogstrukturierungswissen können hinzugefügt werden, ohne die „Inferenzmaschine“ und die interaktive Oberfläche zu ändern. Auch die Komponente zur Erweiterung des Petri-Netzes braucht hierzu nicht angepaßt zu werden, da die Erweiterungsalgorithmen nur auf den allgemeinen Konzepten zur Beschreibung von Dialogstrukturknoten basieren (siehe die Abbildungen 118 und 120). Wir haben weiterhin gesehen, wie in den Methoden zur Berechnung

von Rollenfüllern die modellspezifischen Objektzerlegungsgraphen, die HAMVIS automatisch berechnet, traversiert werden können.

Falls sich aus den deklarierten Modellen bestimmte Objekte, die zur adäquaten Oberflächengestaltung notwendig sind, nicht bestimmen lassen, können bestehende Modelle überarbeitet oder neue Modelle deklariert werden. Obwohl für den Endbenutzer der Anwendung vielleicht unbedeutend, bietet die kommunikationsorientierte Sicht auf die Dialogstrukturierung *zur Entwicklungszeit* die Möglichkeit, notwendige Bestandteile der Graphik automatisch zu bestimmen und Visualisierungen entsprechend zu kombinieren.

Zur Beschreibung von Kombinationsmöglichkeiten von Visualisierungen (vgl. die Diskussion in Kapitel 2.3.3) wurden einige Möglichkeiten zur Umbildung von Dialogstrukturvarianten aufgezeigt. In der aktuellen Version von HAMVIS sind die „Umbildungsregeln“ durch Funktionen beschrieben, die als Werte eine Beschreibung der zu verschmelzenden Knoten bzw. der neuen Oberknoten liefern. Falls eine Umbildung möglich ist, wird zur Realisierung einer Variante die Eingangsdialogstruktur unter Berücksichtigung der lokalen „Änderungen“ kopiert. Im Bereich der IMMP-Systeme wurden vergleichbare Umbildungen von André und Rist untersucht (siehe die Diskussion von IMMP-Systemen in Kapitel 2.4.3). Die in dem HAMVIS-Prototypen realisierten Umbildungen sind vergleichbar mit den in [6] und [3] unter dem Begriff „Structure Sharing“ bezeichneten Umstrukturierungen. Es wird in [6] auch aufgezeigt, daß durch Hinzufügung von Strukturen neue Ausdrucksmöglichkeiten geschaffen werden können. Auch diese Idee scheint im HAMVIS-Kontext zur Visualisierungskomposition umsetzbar, wurde aber noch nicht näher untersucht.

Das Ergebnis der Anwendung einer Umbildungsregel wird von HAMVIS auf Konsistenz und Vollständigkeit geprüft. Wenn in einer möglichen, „neuen“ Variante eine logische Inkonsistenz auftritt, so läßt sich dieses formal ableiten. Auch Unvollständigkeiten der „neuen“ Struktur (im Sinne meiner Definition bezüglich der K-Operatoren) werden entdeckt und ggf. über Berechnungsmethoden für Rollenfüller „repariert“.

Eine automatische Bewertung von Designvarianten wird von HAMVIS nicht vorgenommen. Die Formalisierung des Wissens im Bereich der Antizipationsrückkopplung ist recht komplex (vgl. Kapitel 2.3.4 und Kapitel 2.4.2). Im Entwicklungsszenario von HAMVIS kann der Schnittstellenentwickler die notwendigen Entscheidungen treffen. Ermöglicht wird dieses durch die interaktiven Oberflächen und die dahinterstehenden beschreibungslogischen Ableitungen über konzeptuellen Informationen.

Die aus der Diskursrepräsentationstheorie bekannten Relationen zwischen Segmenten werden durch explizite Objekte modelliert (Aggregatebene der Dialogstruktur). Innerhalb eines Aggregats erfüllt jedes Teilsegment einen bestimmten Diskurszweck. Die explizite Modellierung des Zwecks einer Dialogstruktursegmentes erfolgt ebenfalls mit beschreibungslogischen Ausdrucksmitteln (vgl. die semantischen Diskurssegmentrelationen von Hovy aus Abbildung 51). Neben dem Zweck ist noch die Diskursposition für die Verwaltung von darzustellenden Objekten durch den Ausgabenverwalter wichtig. Auch diese Informationen werden beim Aufbau der Dialogstruktur mit Hilfe der von HAMVIS bereitgestellten beschreibungslogischen Dialogstrukturierungsmodelle abgeleitet. Obwohl die in der aktuellen HAMVIS-Version bereitgestellten Konzepte für Diskurszwecke auf die Erfordernisse der graphischen Kommunikation ausgerichtet sind, läßt sich doch die Basisidee übertragen: Je spezieller

ein Diskurszweck ist, desto spezifischer sind die Einschränkungen für die konkrete Ausgestaltung der Darstellung.

Im HAMVIS-System werden die Diskurszwecke als Marken (markups) zur expliziten Beschreibung einer Objektpräsentation interpretiert. Maßgeblich war auch hier eine Idee von Hovy und Arens zur Abbildung von Diskurssegmentrelationen auf Konstrukte der Satzsprache LaTeX (siehe [133]).

Im nachfolgenden Teilkapitel wird aufgezeigt, wie aus den Beschreibungsformen der Dialogstruktur die jeweils verwendeten Darstellungsattribute hergeleitet werden können. Aufgrund der enormen Komplexität der menschlichen Perzeptionsprozesse, ist eine automatische Bestimmung von Darstellungsattributen extrem schwierig. HAMVIS beschränkt sich auf die automatische Bestimmung von Voreinstellungen für Präsentationsattribute, die aber durch den Schnittstellenentwickler nach seinen Wünschen angepaßt werden können.

3.4 Bestimmung von Präsentationsattributen

Bei Satzsystemen ist die Beschreibung von Textteilen durch Markierungen (markups) eine verbreitete Technik. Anstatt z.B. Zeichensatzattribute direkt anzugeben, wird beschrieben, welche Funktion ein Textstück hat (z.B. Überschrift, Haupttext, Abbildungsunterschrift usw.). Der Vorteil liegt in einer leichteren Änderbarkeit, da alle Textstücke gleicher Markierung automatisch neu gesetzt werden, wenn Markierungsattribute geändert werden. Der Autor braucht sich auch nicht um die exakte Formatierung zu kümmern. Eine Menge von aufeinander abgestimmten Markierungen wird üblicherweise zu einem Dokumentstil zusammengefaßt. Im Bereich der Informationssysteme wurden inzwischen Normungen durchführt, um die verwendeten Beschreibungssprachen zu standardisieren (HTML, SGML usw.).

Die Verwendung von Markierungen bei der Graphikausgabe hat die gleichen Vorteile wie bei der Textverarbeitung. Zeichenattribute werden konsistent vergeben. In Textverarbeitungssystemen kann man einen speziellen Präsentationsstil auszuwählen, der für Markierungen bestimmte Voreinstellungen für die Wahl von Zeichenattributen vorgibt. Ein Editor kann z.B. als Voreinstellung der Graphikattribute für die Markierung „Zitat“ z.B. „kursiv“ angeben.

Nun müssen Voreinstellungen nicht immer in Form von absoluten Werten angegeben werden. Statt also für den Hintergrund strikt „schwarz“ anzugeben, kann auch „möglichst dunkel“ verwendet werden. Die Grundidee des hier vorgestellten Ansatzes besteht darin, Voreinstellungen bzw. Darstellungsanforderungen als eine Menge von Einschränkungen zu formalisieren. Die Vergabe von Zeichenattributen wird dann als Einschränkungsauflösung bzw. -erfüllung realisiert. Darstellungsanforderungen für spezielle graphische Objekte sind durch den Diskurszweck aus der Dialogstruktur definiert. Wichtige Objekte müssen hervorgehoben werden, ständig als Zusatzinformation präsentierte Objekte sollten abgeschwächt werden.

Ich möchte in den folgenden Abschnitten die für HAMVIS realisierten Beschreibungskonstrukte für Markierungen besprechen. Für die Verwendung von Markierungen zur Laufzeit wurde das für HAMVIS verwendete Zielsystem (backend) CLIM erweitert. Die vorgestellten Konstrukte werden in dem von HAMVIS für eine spezielle Anwendung generierten Laufzeitsystem verwendet.

Da die Erweiterungen auch unabhängig von HAMVIS verwendet werden können, möchte ich die bereitgestellten Konstrukte anhand der Beispielanwendung PETS schildern, die wir schon in

Kapitel 2.1.4 betrachtet haben (siehe auch Abbildung 12). In dieser Anwendung werden Testpläne für Relaisschaltkreise berechnet. Ein Testplan besteht aus mehreren Messungen zwischen den extern zugängigen Anschlußpolen der Schaltung (sog. Klemmen). Abbildung 137 zeigt eine Darstellung eines Schaltkreises, in der ein spezieller Meßpfad hervorgehoben wird.

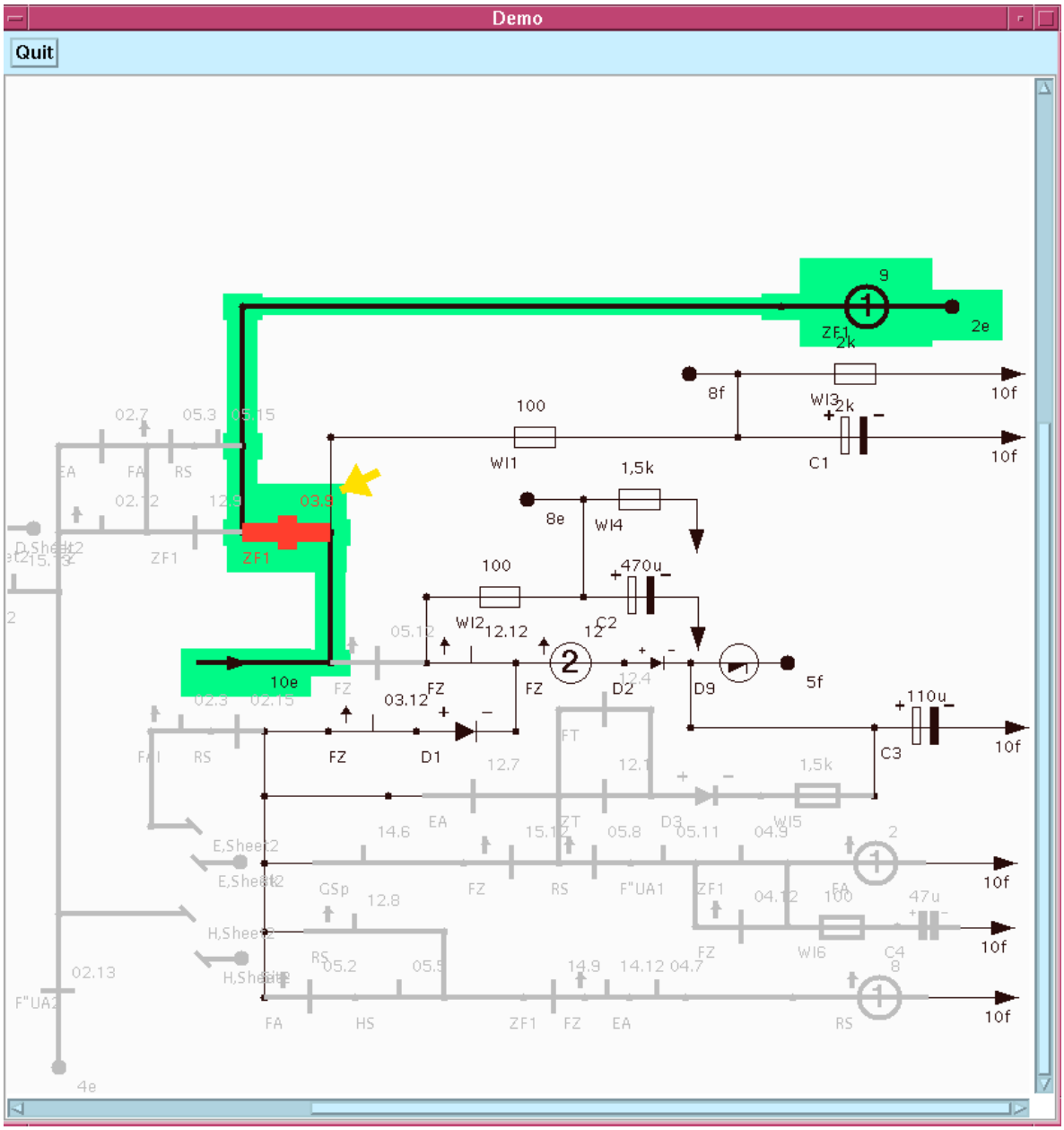


Abbildung 137. Darstellung eines Schaltkreises mit Hervorhebung der Bauelemente auf dem Meßpfad.[107]

Die Abbildung zeigt, daß neben Zeichenattributen wie z.B. Farbe oder Liniendicke auch Unterlegungen und metagraphische Symbole wie z.B. Pfeile unterstützt werden. Der nachfolgende Abschnitt

107. Durch die Widergabe in schwarz-weiß gehen die Farbinformationen des Originals verloren.

schildert kurz, welche Deklarationsformen für die Graphikausgabe mit Markierungen bereitgestellt werden. Näheres zur Implementierung der CLIM-Erweiterung (namens HiGO: Highlighting Graphical Objects) wird von Kaplunova in [144] beschrieben.

3.4.1 Markierungen als Erweiterung von CLIM

Für jedes Teilfenster der Anwendung wird eine Zeichenfunktion definiert. In Abbildung 138 ist eine Skizze der Zeichenfunktion des in Abbildung 137 gezeigten Fensters angegeben.

```
(defmethod draw-circuit ((frame pets)
                         stream)
 (let ((layout (frame-current-layout frame))
       (style (frame-style frame)))
   (with-presentation-markup ('tested-elements-on-path
                              frame style layout stream)
     (dolist (circuit-element (pets-tested-elements frame))
       (draw-pets-element circuit-element frame stream)))
   (with-presentation-markup ('rest-elements-on-path
                              frame style layout stream)
     (dolist (circuit-element (pets-rest-elements-on-path frame))
       (draw-pets-element circuit-element frame stream)))
   (with-presentation-markup ('rest-elements
                              frame style layout stream)
     (dolist (circuit-element (pets-rest-elements frame))
       (draw-pets-element circuit-element frame stream)))))
```

Abbildung 138. Skizze der Zeichenfunktion des PETS-Anwendungsfensters.

Die Ausgabe eines einzelnen Schaltkreiselements erfolgt mit Hilfe der Funktion `draw-pets-element`. Auf die konkrete Graphikausgabe der Schaltkreiselemente möchte ich hier allerdings nicht eingehen. Es werden letztendlich Standardfunktionen wie `draw-line`, `draw-circle`, `draw-rectangle` usw. verwendet.

In der Zeichenfunktion für das Fenster (`draw-circuit`) wird die Menge der zu zeichnenden Schaltkreiselemente in drei verschiedene (disjunkte) Teilmengen geteilt (die jeweiligen Objektmengen werden durch die Funktionen `pets-tested-elements`, `pets-rest-elements-on-path`, `pets-rest-elements` geliefert). Der Zugriff auf die jeweils zu verwendenden Graphikattribute erfolgt durch Angabe des Markierungsnamens mit der Form `with-presentation-markup`.

```
(with-presentation-markup (<markup> <frame> <style> <layout> <stream>)
  ...)
```

Als Argument von `with-presentation-markup` wird u.a. auch der vorgesehene Präsentationsstil des Anwendungsrahmens angegeben (bestimmt durch die Funktion `frame-style`).

In ähnlicher Weise wie bei Textsatzsystemen sind die Markierungen jedoch nicht unabhängig voneinander. Wenn z.B. in einem Satzsystem etwas hervorgehoben werden soll (siehe z.B. die `em`-Umgebung in LaTeX), dann kann z.B. Kursivschrift verwendet werden. Soll aber der umgebende Text schon kursiv gesetzt sein, so wird zur Hervorhebung die Standardschrift verwendet usw. Die Markierungsstruktur der Anwendung und die Abhängigkeiten zwischen Markierungen werden zur Entwicklungszeit festgelegt. Hierzu wird das folgende Konstrukt verwendet:

```
(define-presentation-markup (<markup1> <frame>)
```

```
  :parent <markup2>
  :class <presentation-markup-class>)
```

Für die PETS-Anwendung wurde mit Hilfe einer Reihe von entsprechenden `define-presentation-markup`-Deklarationen der in Abbildung 139 gezeigte Markierungsbaum definiert. Markierungen gelten jeweils für Mengen von Objekten. Der Markierungsbaum definiert eine Zerlegung der Menge von zu präsentierenden Objekten in disjunkte Teilmengen. Die Markierung `background` wird in spezieller Weise behandelt, sie ist Instanz einer speziellen Klasse (siehe die Option `:class` in `define-presentation-markup`) und wird für die Zuweisung der Farbe des Fensterhintergrundes verwendet.[108]

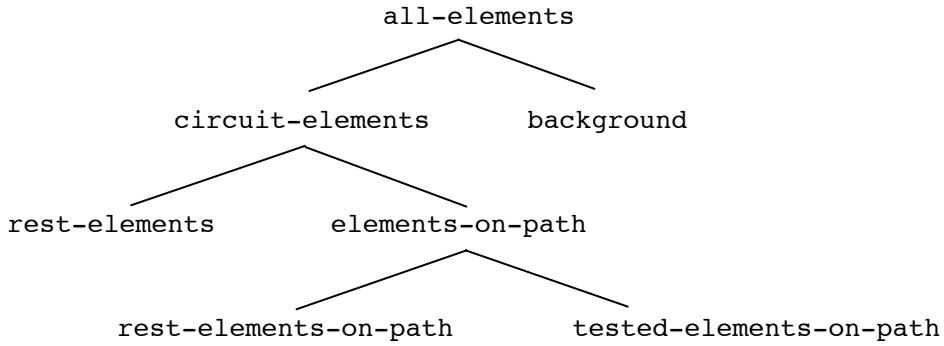


Abbildung 139. Hierarchie der Markierungen in der PETS-Anwendung.

In den Zeichenfunktionen werden als Parameter von `with-presentation-markup` nur Blattmarkierungen verwendet. Die Anwendung der Hintergrundmarkierung (`background`) erfolgt implizit.

Graphische Elemente, die unter einer speziellen Markierung ausgegeben werden, sollen sich (mehr oder weniger stark) von denjenigen Objekten unterscheiden, die unter einer anderen Markierung ausgegeben werden. Standardmäßig wird also dafür gesorgt, daß sich die Zeichenattribute der Blattmarkierungen unterscheiden. Es können aber auch speziellere Angaben zu Beschreibung der Markierungen gemacht werden.

Abhängigkeiten zwischen Zeichenattributen

Um für einen bestimmten Präsentationsstil Abhängigkeiten zwischen den Zeichenattributen verschiedener Markierungen definieren zu können, wird folgende Deklarationsform verwendet.

```
(declare-dependence <style>
  <markup1>
  <markup2>)
```

Die konkrete Definition der Abhängigkeiten erfolgt durch Angabe einer Menge von Prädikaten im Rumpf der Form `define-presentation-markup-constraints`.

```
(define-presentation-markup-constraints <style>
    ((m1 <markup1>) (m2 <markup2>))
  ...)
```

108. Andere Zeichenattribute wie z.B. Liniendicke sind für den Hintergrund nicht anwendbar.

Die konkrete Definition der möglichen Prädikate wird im nächsten Abschnitt erläutert. Ich möchte zunächst einige Beispiele aufführen. Nehmen wir also z.B. an, der Rest der Schaltkreiselemente (Markierung `rest-elements`) soll sich nicht stark vom Hintergrund (Markierung `background`) unterscheiden:

```
(define-presentation-markup-constraints <style>
    ((m1 rest-elements) (m2 background))
  `(similar ,m1 ,m2))
```

Die höheren Ebenen der Markierungshierarchie können dazu verwendet werden, in einfacher Weise Abhängigkeiten zwischen den Zeichenattributen verschiedener Blattmarkierungen zu definieren. Die Abhängigkeiten werden dann jeweils wechselseitig zwischen allen erreichbaren Blattmarkierungen definiert. Zur Realisierung der Unterscheidbarkeit von unter verschiedenen Markierung ausgegebenen Objekten wird also für jeden Markierungsbaum implizit festgelegt:

```
(define-presentation-markup-constraints <style>
    ((m1 <name-of-left-child-of-root-node>) (m2 <name-of-right-child-of-root-node>))
  `(distinguishable ,m1 ,m2))
```

Diese Einschränkung wird im Markierungsbaum „nach unten“ propagiert, so daß `distinguishable` wechselseitig für alle Blattmarkierungen deklariert wird.

Auch absolute Einschränkungen für eine einzelne Markierung können in ähnlicher Weise angegeben werden.

```
(define-presentation-markup-constraints-for-style <style>
    ((m1 <markup>))
  ...)
```

Für die Beispielanwendung sind die getesteten Elemente auf dem Meßpfad besonders wichtig:

```
(define-presentation-markup-constraints-for-style <style>
    ((m1 tested-elements-on-path))
  `(very-important ,m1))
```

In einen Markierungsgraphen werden ein- und zweistellige Prädikate zur Beschreibung von Einschränkungen zur Berechnung von Präsentationsattributen definiert. Die Definition von Einschränkungen erfolgt in Abhängigkeit von einem Präsentationsstil. In Abbildung 140 wird ein Beispiel für die Ausgabe des gleichen Schaltkreises aufgezeigt, wobei allerdings für einen Präsentationsstil `Dark-Background-Style` durch spezielle Einschränkungen der Hintergrund möglichst dunkel gewählt wurde. Die Zeichenattribute der anderen Markierungen werden entsprechend angepaßt. Eine Hervorhebung durch Unterlegungen erfolgt in diesem Präsentationsstil nicht. Vergleiche hierzu die Darstellung des gleichen Schaltkreises mit dem Stil `Light-Background-Style` (Abbildung 137).

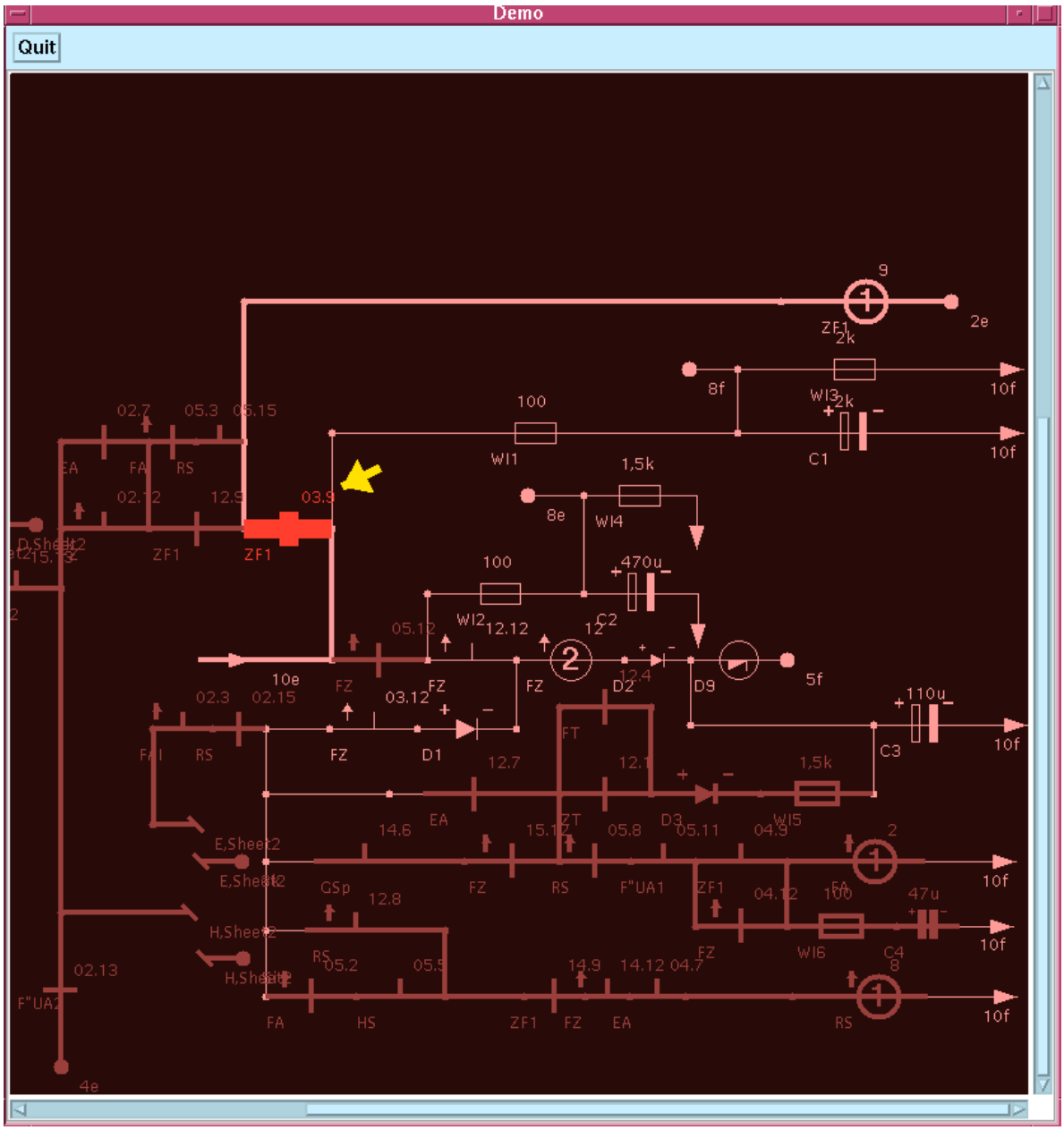


Abbildung 140. Darstellung des Schaltkreises mit dem im CAD-Bereich üblichen Darstellungsstil mit dunklem Hintergrund.

Einschränkungserfüllung durch Techniken der logischen Programmierung

Die formale Definition von Einschränkungen wie z.B. `similar` oder `very-important` erfolgt durch logische Prädikate. Zur Definition von Prädikaten und zur Auflösung von Einschränkungen wurde das System CLP(R) verwendet [123] [137]. Mit CLP(R) wird ein einschränkungsbasiertes System zur logischen Programmierung implementiert. Über der Domäne der reellen Zahlen werden Einschränkungen für die Wertebereiche von logischen Variablen unter Verwendung von linearen Ungleichungssystemen unterstützt. Die Einschränkungen werden innerhalb des Wiederaufsetzsche-

mas des logischen Beweisers verwaltet. Abbildung 141 zeigt einen Ausschnitt aus der Menge der vordefinierten Prädikatsdefinitionen mit CLP(*R*)-Hornklauseln.[109]

```
(defpredicate
  ((distinguishable ?markup1 ?markup2)
   (different-colors ?markup1 ?markup2))
  ((distinguishable ?markup1 ?markup2)
   (different-line-thickness ?markup1 ?markup2))
  ((distinguishable ?markup1 ?markup2)
   (different-frame ?markup1 ?markup2)))

(defpredicate
   ((very-important ?markup)
    (thick-lines ?markup)
    (arrow-frame ?markup))
   ...)

(defpredicate
  ((similar ?markup1 ?markup2)
   (similar-colors ?markup1 ?markup2))
  ((similar ?markup1 ?markup2)
   (similar-line-thickness ?markup1 ?markup2))
  ((simlar ?markup1 ?markup2) i
   (simlar-frame ?markup1 ?markup2)))

(defpredicate
  ((similar-colors ?m1 ?m2)
   (presentation-markup-color-hue ?m1 ?h1)
   (presentation-markup-color-hue ?m2 ?h2)
   (presentation-markup-color-intensity ?m1 ?i1)
   (presentation-markup-color-intensity ?m2 ?i2)
   (presentation-markup-color-saturation ?m1 ?s1)
   (presentation-markup-color-saturation ?m2 ?s2)
   (= ?h1 ?h2)
   (= ?s1 ?s2)
   (> ?i2 (+ ?i1 0.19))
   (< ?i2 (+ ?i1 0.32)))
  ...)
```

Abbildung 141. Ausschnitt aus der CLP(*R*)-Wissensbasis zur Definition von Einschränkungen für Zeichenparameterwerte.

Für die Repräsentation von Farben wird das IHS-Farbmodell mit den Parametern Intensität, Farbwert und Sättigung verwendet.[110] Das untere Prädikat aus Abbildung 141 demonstriert in der ersten Prädikatsklausel die Möglichkeit, eine ähnliche Farbe durch eine kleine Variation der Intensität zu erhalten.

---

109. Die Definition von Prädikaten erfolgt in Lisp-Notation. Die Übertragung in die Standard-Hornklauselschreibweise ist trivial. Hinter `defpredicate` stehen die Klauseln für ein bestimmtes Prädikat. Der erste Term einer Klausel definiert der Kopf, die anderen Terme definieren den Rumpf einer Regel. Prädikate können wie üblich in Lisp-Systemen auf Tastendruck evaluiert werden. Sie werden automatisch an das angekoppelte CLPR(*R*)-System gesandt. CLP(*R*) wird als Unix-Programm gestartet und über Dateikommunikation mit dem Lisp-System verbunden. Zur Kommunikation mit dem CLP(*R*)-Programm werden die Lisp-Horn-Klauseln in die von CLP(*R*) verwendete Edinburgh-Notation umgewandelt. Das Ergebnis einer Anfrage wird von CLP(*R*) in eine Datei geschrieben und von Lisp wieder interpretiert.

Durch Sammeln von Einschränkungen innerhalb der Markierungsstruktur einer Anwendung wird eine Anfrage an CLP(*R*) generiert. Wenn eine Lösung des Einschränkungssystems existiert, können konkrete Werte für die einzelnen Zeichenparameter einer Markierung bestimmt werden. Die möglichen Werte der logischen Variablen werden dem CLP(*R*)-System als Liste vorgegeben. Dadurch lassen sich auch spezielle Ausgabegeräte berücksichtigen (z.B. 4-Bit Farbmonitore etc.). Die Reihenfolge innerhalb der Liste der möglichen Werte beeinflußt die innerhalb des Wiederaufsetz-Schemas von CLP(*R*) zuerst gefundene Lösung. Zur Definition der möglichen Werte können für bestimmte Darstellungsstile die HiGO-Standardmethoden überschrieben werden, so daß z.B. eine spezielle Sortierung erreicht werden kann. Für einen möglichst dunklen Hintergrund wird beispielsweise folgende Definition verwendet.

```
(defmethod presentation-descriptor-intensity-alternatives
        ((style dark-background-style) (markup background))
  (sort (copy-list (call-next-method)) #'<)))
```

Aus Platzgründen kann in dieser Arbeit nur ein kurzer Überblick über die Basisideen zur Verwendung von Markierungen und über die CLIM-Erweiterung HiGO gegeben werden. Ich möchte im nächsten Abschnitt die Anwendung der Techniken im Kontext von HAMVIS besprechen und komme dazu wieder auf das XKL-Beispiel zurück.

3.4.2 Anwendung in HAMVIS: Herleitung der Markierungsstruktur

Das Ergebnis der Dialogstrukturierungsphase ist eine Dialogstruktur und ein erweitertes Laufzeitnetz. Im erweiterten Laufzeitnetz wurde bei jeder Stelle vermerkt, welche Diskurszwecke durch die Präsentation der Objekte erreicht werden sollen, die sich jeweils in den Stellen des Laufzeitnetzes befinden. Für jede Stelle wurde außerdem in der Dialogstruktur angegeben, in welchem Darstellungsfenster und unter welchem Modell die Ausgabe erfolgen soll. Diskurszwecke werden in Markierungen umgesetzt. Das spezielle Konzept des Diskurszwecks (vgl. Abbildung 119) legt dabei die Einschränkungen fest, die für die Darstellungsattribute einer entsprechenden Markierung definiert werden.

Dem HAMVIS-Benutzer steht eine interaktive Oberfläche für die Inspektion der automatisch bestimmten Markierungsstruktur für alle Fenster einer Anwendung zur Verfügung. In Abbildung 142 wird als Beispiel die Markierungsstruktur für das Hauptfenster der XKL-Anwendung gezeigt.

110. Vergleiche die Arbeit von MacIntyre [187]. Es wird hier ein ähnlicher Ansatz zur Berechnung von Zeichenattributen verfolgt. MacIntyre vergleicht mehrere Farbmodelle und bewertet sie auf ihre Eignung zur einschränkungsbasierten Auswahl von Farben in Fenstersystemen.

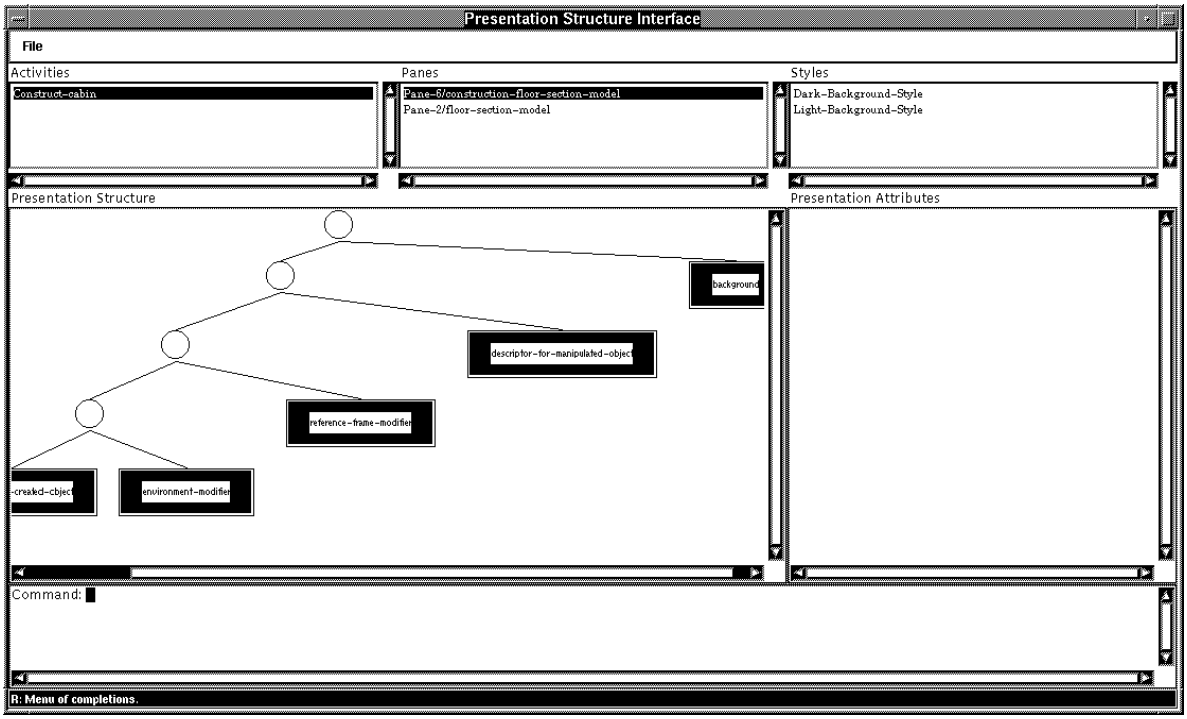


Abbildung 142. Markierungsstruktur des Graphikfensters der XKL-Anwendung.

Durch Auswählen eines Präsentationsstils (siehe die rechte obere Tabelle in Abbildung 142) wird für alle Markierungsbäume einer Anwendung intern eine Anfrage für CLP(*R*) zusammengestellt. Das Ergebnis wird dem Entwickler als Gestaltungsvorschlag in der interaktiven Oberfläche präsentiert. Die automatisch bestimmten Zeichenparameter sind als Vorschlag zu verstehen und können durch den Entwickler noch nach seinen Wünschen angepaßt werden. Zu beachten ist aber, daß die Anpassung der Zeichenattribute für eine Markierung erfolgt, also für alle mit der entsprechenden Markierung ausgegebenen Objekte wirksam wird.

Da es der Fall sein kann, daß ein Objekt in mehreren Stellen des Laufzeitnetzes auftritt (ein Objekt kann verschiebbar sein und auch als Objekt der Umgebung eines anderen Objekts auftreten) muß durch den Ausgabenverwalter des Laufzeitsystems eine eindeutige Markierung erst noch aus der Menge der möglichen Markierungen bestimmt werden. Hierzu sind die Markierungen priorisiert (siehe die Wissensbasis in Abbildung 119). Man kann sich vorstellen, ein zu zeichnendes Objekt durchläuft von oben den Markierungsbaum (siehe die Darstellung in Abbildung 142) und wird wie in einer Sortiermaschine in die erste mögliche Markierung einsortiert. Mit dieser Markierung wird das Objekt dann gezeichnet.

Im Zusammenhang mit Markierungen ist noch zu betonen, daß die für eine Markierung verwendeten Einschränkungen nicht (oder nicht direkt) von den Konzepten der gezeichneten Objekte abhängen, sondern durch die kommunikative Funktion der Objekte in der Gesamtdarstellung bestimmt werden. Ein Beispiel für eine Markierung sind die manipulierten Objekte (Diskurszweck `descriptor-for-manipulated-object` siehe Abbildung 142).

Als Beispiel für diese Markierung möchte ich hier noch einmal die Stelle `has-moved-entity` aus Abbildung 114 betrachten. Nicht jedes „manipulierte Objekt“ sollte gleich aussehen. Es ist beispielsweise in der XKL-Anwendung sinnvoll, Küchen, Waschräume und Sitze durch spezielle Graphikattribute unterscheidbar zu machen. Zur Entwicklungszeit ist allerdings nur bekannt, daß zur Laufzeit Objekte in die Stelle gelegt werden, die von `cabin-object` subsumiert werden. Es werden zur Berechnung von Darstellungsattributen weitere Informationen benötigt, aus denen hervorgeht, welche Unterkonzepte von `cabin-object` bezüglich der graphischen Ausgabe unterschieden werden sollen. In diesem Kontext sind die Konzeptcharakterisierungen „Basiskonzept“ und „Primärkonzept“ relevant (siehe die Definitionen in Kapitel 3.1.6). Im Modell `construction-floor-section-modell` sind unterhalb von `cabin-object` die Primärkonzepte `galley`, `lavatory` und `seat` deklariert (siehe Abbildung 77). Für jedes Primärkonzept werden zur Entwicklungszeit der Anwendung andere Zeichenattribute verwendet. Das wiederum bedeutet, daß für eine Markierung nicht für jeden Zeichenparameter ein konkreter Wert bestimmt wird, sondern zur Entwicklungszeit jeweils ein gewisser „Bereich“ vorgesehen werden muß. Die Größe des Bereichs richtet sich nach der Anzahl der Primärkonzepte, die potentiell unter einer Markierung dargestellt werden. Die notwendigen Fallunterscheidungen bei der Ausgabe werden durch den Ausgabenverwalter vorgenommen.

3.4.3 Zusammenfassung: Bestimmung der Präsentationsstruktur

Mit der Definition von Markierungen und der Bestimmung der jeweiligen Bereiche der Zeichenattribute ist eine weitere Teilkomponente zur Generierung des Laufzeitsystems einer Anwendung festgelegt. Ich möchte die gesammelten Informationen unter dem Begriff Präsentationsstruktur zusammenfassen.

Mit der Repräsentation von Darstellungseinschränkungen konnte ein erster Schritt zu einer Formalisierung in diesem Bereich getan werden. Obwohl viele Empfehlungen zur Gestaltung von Benutzungsoberflächen veröffentlicht wurden (siehe z.B. Murch [220], MacDonald [186], Haber [113]), zeigte sich auch bei HiGO wieder einmal, daß die Formalisierung von Designvorschlägen und -richtlinien außerordentlich schwierig ist (siehe auch die Arbeiten zur Evaluierung von Benutzungsschnittstellen: z.B. [255]).

HiGO unterstützt als einfache „visuelle Variablen“ (vgl. Kapitel 2.3.1) die Attribute Strichdicke und Zeichenfarbe sowie Unterlegungen und spezielle Markierungspfeile. Das verwendete IHS-Farbmodell stellt einen Kompromiß zwischen Implementierungsaufwand und Nutzen dar. Es sind inzwischen „nichtlineare“ Farbmodelle bekannt, die besser auf die Farbperzeption des Menschen abgestimmt sind (siehe z.B. Robertson [263]). Die Komplexität der menschlichen Perzeptionsprozesse ist jedoch enorm, so daß auf absehbare Zeit Modelle nur für Teilaspekte erwartet werden können und ein interaktives Vorgehen notwendig ist (siehe z.B. Langen [170]).

Auch domänenspezifische Darstellungsformen können in Form von visuellen Sprachen in den Mechanismus der Verwaltung von Einschränkungen für Zeichenattribute eingebunden werden. In ähnlicher Weise wie von Helm und Mariott vorgestellt (siehe die Diskussion von visuellen Sprachen in Kapitel 2.3.2) kann durch den Schnittstellenentwickler eine Notation für bestimmte Primärkonzepte definiert werden. Ein Beispiel ist die in der XKL-Anwendung im „Cabin Configuration Guide“ verwendete Darstellung von Küchen durch Querstriche an der Rückwand zur Kennzeichnung der Anzahl der Küchenwagen (trolleys). Zur Einbettung dieser visuellen Notation könnten weitere CLP(R)-Prädikate zu HiGO hinzugefügt werden. Bei Verwendung von anwendungsspezifischen Notationen ergibt

sich innerhalb der Verwaltung der Markierungseinschränkungen ein weiterer Spielraum für die Zuweisung von Zeichenattributen bei anderen Parametern. Eine weitere Möglichkeit zur Erhöhung der Ausdruckskraft bietet sich durch die Verwendung von Beschriftungen (siehe Kapitel 2.3.2). Die Arbeiten zur Integration dieser Techniken wurden allerdings noch nicht durchgeführt. Zu beachten ist auch, daß die Vergabe von Zeichenattributen wie Farbe und Strichdicke i.a. nicht ohne weiteres ohne Berücksichtigung der Darstellungsgröße erfolgen kann, so daß HiGO Zugriff auf „prototypische Objekte“ haben sollte.

Die Menge der von HAMVIS bzw. HiGO verwendeten Darstellungsattribute: Farbe, Stiftdicke sowie Sonderformen wie z.B. Unterlegungen und metagraphische Hervorhebungssymbole (z.B. Pfeile) ließe sich noch erweitern. Livingstone schlägt nach einer Analyse der menschlichen Sinnesphysiologie z.B. auch Bewegung von Graphikobjekten als besondere Hervorhebungstechnik vor [177].

Die realisierte Anbindung der Verwaltung von Einschränkungen für Markierungen erfolgt zur Zeit nicht über beschreibungslogische Schlüsse, sondern über objektorientierte Programmiertechniken mit Vererbung und Methodenkombination. Eine interessante Perspektive bietet die Erweiterung des beschreibungslogischen Repräsentationssystems um sogenannte konkrete Domänen (concrete domains) unter Einbeziehung eines Einschränkungssystems (siehe [15] und [116]). Ein solches System würde eine weitere Formalisierung erlauben, stand allerdings als Erweiterung von CLASSIC nicht zur Verfügung.

Das als Prototyp realisierte System stellt die Voraussetzungen bereit, um ein Laufzeitsystem für eine Anwendung generieren zu können. Im folgenden Abschnitt möchte ich zur Illustration des Laufzeitsystems ausgewählte Teile des für XKL generierten CLIM-Codes aufführen.

3.5 Codegenerierung für ein UIMS: Die endgültige Anwendung

Mit einem UIMS ist die wesentliche Grundlage für die Erzeugung eines Laufzeitsystems gegeben (siehe die Ausführungen in Abschnitt 2.1.4). Ich betrachte in diesem Abschnitt die Verwendung des UIMS CLIM als Zielsystem. HAMVIS definiert einige anwendungsübergreifende Erweiterungen zu CLIM, die zur Erzeugung des Laufzeitsystems notwendig sind (Interpreter für Laufzeit-Petri-Netz usw.). Für die Darstellung von graphischen Objekten wurden entsprechende Klassen für Anwendungsrahmen und Teilfenster als Erweiterung von CLIM implementiert. Diese Bausteine können in verschiedenen Anwendungsklassen eingesetzt werden. Weiterhin wurden für die in dieser Arbeit betrachtete Anwendungsklasse der Layoutprobleme entsprechende Interaktionswerkzeuge implementiert, so daß die Bibliothek von Aktionenkonzepten im HAMVIS-Grundmodell aufgebaut werden konnte. Diese vordefinierten Teile sind im unteren Teil der Abbildung 52 auf der linken Seite eingezeichnet. Für eine spezielle Anwendung müssen noch weitere Deklarationen für das Ziel-UIMS generiert werden (siehe Abbildung 52). Die Generierung nimmt dabei Bezug auf die drei Modelle, die während der Designphase erstellt worden sind: erweitertes Laufzeitnetz, Dialogstruktur und Präsentationsstruktur.

Im folgenden wird skizziert, wie im HAMVIS-Kernsystem Interaktionswerkzeuge bereitgestellt werden, welche Aufgabe die vordefinierten Klassen für Anwendungsrahmen und Teilfenster haben und welche Struktur der für XKL generierte CLIM-Code hat.

3.5.1 Interaktionswerkzeuge für eine Anwendungsklasse

Bei der Vorstellung der Aktionenmodellierung in Kapitel 3.2.3 wurden die Aktionenkonzepte mit Funktionen zur Realisierung durch Interaktionsgesten verbunden (siehe z.B. die Abbildungen 96 und 97). Die Funktionen werden zur Laufzeit aufgerufen, wobei die Objekte aus den Eingangsstellen der Aktion als Parameter übergeben werden. In Abbildung 143 ist die Definition der in den Abbildungen 96 und 97 verwendeten Funktionen widergegeben. Neben den im Aktionenmodell aufgeführten Eingangsparametern (Kasusrollen) wird das manipulierte Objekt übergeben. Es kann sich hierbei um ein angeklicktes Objekt handeln oder um das Ziel einer Ziehen-und-Fallenlassen-Geste usw. Zu beachten ist, daß nicht das Graphikobjekt übergeben wird, sondern das mit einem Graphikobjekte assoziierte Anwendungsobjekt. Die Assoziation wird automatisch durch CLIM verwaltet (siehe Kapitel 2.1.4).

```
(defun spatial-localization-in-xy-bounding-rectangle
     (&key
      manipulated-object
      model
      stream
      has-localized-entity
      has-reference-object)
  (list `(has-localized-entity ,has-localized-entity)
        `(has-reference-object ,has-reference-object)
        `(has-new-position-constraint
          ,(list (position-constraint manipulated-object)))))

(defun move-spatial-object-within-xy-bounding-rectangle-position-constraint
    (&key
     manipulated-object
     model
     stream
     has-moved-entity)
  (list `(has-moved-entity ,(list manipulated-object))
        `(has-new-position-constraint
         ,(list (clim:accept
                 'bounding-rectangle-position-constraint-with-xy-default
                 :stream stream
                 :default (position-constraint manipulated-object)
                 :prompt nil
                 :view model)))))
```

Abbildung 143. Anbindung von Interaktionstechniken durch Funktionen, die im Aktionenmodell mit den Aktionenkonzepten assoziiert werden.

Der Wert der Funktion `spatial-localization-in-xy-bounding-rectangle` ist ein Assoziationsliste von Paaren bestehen aus Ausgangsparameternamen und entsprechendem Wert. Die Definition der Funktion `spatial-localization-in-xy-bounding-rectangle` ist trivial. Es werden das lokalisierte Objekt, die Referenzobjekte und die Positionseinschränkung des manipulierten Objekts zurückgegeben. Das Laufzeitsystem sorgt dafür, daß die Werte in die entsprechenden Ausgangstellen gelegt werden. Durch den Petri-Netz-Interpreter werden dann die nachfolgenden Transitionen gefeuert, d.h. es werden die zugeordneten Berechnungs- und Speicherfunktionen der Anwendung evaluiert.

Die Definition der zweiten Funktion `move-spatial-object-within-xy-bounding-rectangle-position-constraint` ist etwas komplexer. Es wird ein neues Objekt vom Konzept `bounding-rectangle-position-constraint-with-xy-default` zur Beschreibung der neuen Position inklusive der möglichen Positionsveränderungen erzeugt. Die Erzeugung erfolgt in einer `accept`-Methode.

```
(define-presentation-method accept
    ((type bounding-rectangle-position-constraint-with-xy-default)
     stream
     (view hamvis::section-view-model-class)
     &key
     default)
  ...)
```

Abbildung 144. Beispiel für die Definition einer `accept`-Methode.

Auf die Präsentation des Rumpfes der Methode möchte ich hier verzichten. Die Verwaltung von Mausereignissen und die Realisierung der Beschränkung der Verschiebemöglichkeiten sowie die notwendigen Koordinatentransformationen (Translation und Skalierung) umfassen ca. 300 Zeilen CLIM-Code.

Für eine Anwendungsklasse sind sehr viele solcher `accept`-Methoden deklariert. Wie oben geschildert, beschreibt HAMVIS durch die Modellierung von Aktionen auf konzeptueller Ebene jedoch Interaktionsdienste auf einem hohen Abstraktionsniveau, so daß sich der Entwickler nicht mit den „Innereien“ der Akzeptierungsmethoden beschäftigen muß. Durch beschreibungslogische Inferenzen werden die für Eingangsdatentypen und erwarteten Werte „passenden“ Interaktionsformen automatisch ermittelt. Falls dieses nicht möglich ist, bietet HAMVIS dem Schnittstellenentwickler die jeweils möglichen Aktionenkonzepte zur Auswahl an.

3.5.2 Vordefinierte Frame- und Pane-Klassen als Erweiterung von CLIM

Für HAMVIS wurde eine neue Klasse für CLIM-Anwendungsrahmen definiert (`hamvis-frame`). Die Klasse ermöglicht die Handhabung von Markierungen (`higo:presentation-markup-mixin`) und übernimmt die Verwaltung des Laufzeit-Petri-Netzes. Zur Realisierung des Ausgabenverwalters sind besondere Klassen von Teilfenstern (panes) definiert. Für jedes Teilfenster werden die gezeichneten Objekte in einer „Historie“ zusammen mit dem Diskurszweck und der Diskursposition gespeichert. Diskurszweck und Diskursposition ergeben sich aus der Stelle des Laufzeitnetzes, aus der die Objekte an den Ausgabenverwalter übermittelt wurden. Der Diskurszweck bestimmt die in den Ausgabefunktionen zu verwendende Markierung (s.o.).

In den vordefinierten Zeichenfunktionen wird auf die projektiven geometrischen Daten der Anwendungsobjekte zugegriffen. Dabei wird das für das Teilfenster verwendete Modell berücksichtigt. In Kapitel 3.1.2 wurde schon zur Motivation der von HAMVIS eingeführten Modelle ein Beispiel für die Präsentation von räumlichen Objekten geschildert (siehe die Definition der `present`-Methode). Über das Modell wird die entsprechende Unterrelation zu `has-projective-form` bestimmt. Füller der entsprechenden Rolle werden (gesteuert durch die Modellklasse) von HAMVIS automatisch berechnet (siehe die Beispiele in Kapitel 3.1.6).

Die Ausgabefunktionen für Graphikfenster sehen weiterhin eine automatische Fokussierung und Skalierung der vorgesehenen Referenzrahmen innerhalb des Sichtbereichs der Teilfenster vor. Dadurch wird erreicht, daß die in Weltkoordinaten definierten geometrischen Projektionen der Anwendungsobjekte auf Koordinaten des rollbaren Sichtbereichs abgebildet werden.

Für die Ausgabe von Objekten muß dem Ausgabenverwalter noch der zu verwendende Präsentationstyp vorgegeben werden. Der zur Ausgabe zu verwendende Präsentationstyp ist durch die Benutzeraktion festgelegt und ergibt sich aus dem erweiterten Laufzeitnetz. Das Skelett der Ausgabefunktion sieht also wie folgt aus.

```
(let ((transition (get-transition-for-user-action place)))
  (dolist (object (get-objects-from-place place ...))
    (present (list object transition)
             :type (get-presentation-type place)
             :view (stream-model stream)
             :stream stream)
    ...))
```

Die Funktion `present` wird mit drei Parametern aufgerufen: das Anwendungsobjekt, der Präsentationstyp und der Ausgabestrom (das Teilfenster). Als „Anwendungsobjekt“ wird von HAMVIS ein Tupel mit dem „eigentlichen“ Anwendungsobjekt und der Transition verwendet, da dies für die Verwaltung der Interaktionsgesten benötigt wird. In vorigen Kapiteln wurde schon die Definition von speziellen `present`-Methoden für entsprechende Sichten diskutiert.

In ähnlicher Weise wie Unterlegungen und andere metagraphische Hervorhebungstechniken automatisch von HAMVIS realisiert werden, lassen sich in diesem Kontext auch anwendungsspezifische visuelle Notationen durch spezielle Ausgabetechniken berücksichtigen. Für nicht-geometrische Modelle sind in einer entsprechenden `present`-Methode entsprechende Abbildungsfunktionen zu berücksichtigen (siehe Kapitel 2.3.2 und Kapitel 3.1.11). Ich möchte aus Platzgründen die Ausführungen hierzu nicht vertiefen und hiermit die Darstellung der allgemeinen, anwendungsübergreifenden CLIM-Erweiterungen beenden. Neben den allgemeinen CLIM-Erweiterungen wird durch HAMVIS für eine konkrete Anwendung noch *anwendungsspezifischer* CLIM-Code generiert. Dieses wird im nächsten Abschnitt erläutert.

3.5.3 Automatische Codegenerierung für die XKL-Anwendung

Die verschiedenen Ebenen der Aktionenmodellierung müssen in entsprechende Konstrukte des Ziel-UIMS abgebildet werden. In den vorigen Abschnitten wurde schon angedeutet, daß dieser Schritt bei Verwendung von CLIM recht einfach ist. Eine Anbindung eines anderen UIMS scheint aber nicht unmöglich. Die zur Abbildung benötigten Informationen stehen im Aktionenmodell, in der Dialogstruktur, in der Präsentationsstruktur und im erweiterten Laufzeitnetz. Im Zusammenhang mit der Abbildung auf CLIM sind folgende Aspekte relevant (siehe auch Kapitel 2.1.4):

- Abbildung der Applikation auf einen CLIM-Anwendungsrahmen,
- Abbildung von Aktivitäten auf eine Konfiguration von Fenstern im Anwendungsrahmen,
- Abbildung von zusammengesetzten Benutzeraktionen (Handlungsalternativen innerhalb einer Aktivität) auf Kommandos,

- Festlegung von Interaktionsgesten durch Definition von und Präsentation-nach-Kommando-Abbildungen,
- Vorbereitung der Ausführung des Laufzeitnetzes durch Einfügung des entsprechenden Codes in den Rumpf eines Kommandos.

Beispiele für die Funktionalität der generierten XKL-Anwendung

Bevor in den folgenden Abschnitten die Abbildung auf CLIM-Konstrukte erläutert wird, betrachten wir einige Beispiele, die die letztendlich erzeugte XKL-Anwendung charakterisieren. Wir nehmen an, daß die notwendigen Berechnungs- und Speicherfunktionen durch den Anwendungsentwickler schon implementiert worden sind (siehe auch Abbildung 4). Dabei ist zu beachten, daß die Anwendung auch schon mit einer partiellen Implementierung von Berechnungs- und Speicherfunktionen getestet werden kann. HAMVIS ermöglicht also in diesem Kontext ein *prototypisches aber doch systematisches Vorgehen* bei der Systementwicklung. Weiterhin kann HAMVIS auch visuelle Oberflächen für *partielle Aktionenmodelle* erzeugen, so daß eine inkrementelle Entwicklung der Anwendung unterstützt wird. Trotz der im Vergleich zu Schnittstellenbaukästen (interface builder) z.T. abstrakten Modelle ist dadurch durchaus eine frühe Rückkopplung durch das tatsächliche Aussehen und Verhalten der generierten Anwendung möglich. Die in Abbildung 4 aufgeführten Entwicklungsphasen können mehrfach durchlaufen werden. Die nachfolgenden Abbildungen vermitteln einen Eindruck von der generierten XKL-Oberfläche.

Es soll hier noch einmal darauf hingewiesen werden, daß dem Benutzer einer mit HAMVIS generierten Anwendung die innerhalb des Interaktionszyklus möglichen Aktionen nicht in einem Nachrichtenfenster mitgeteilt werden. Wie in modernen Oberflächen üblich, kann der Benutzer die Maus auf ein Graphikobjekt bewegen und erhält in der Mausdokumentationszeile Informationen über die möglichen Interaktionsgesten und damit über die möglichen Handlungen. Bei Verwendung eines anderen Zielsystems würde statt einer Mausdokumentationszeile vielleicht eine Sprechblase erscheinen (siehe den Mechanismus des „balloon help“ auf dem Macintosh).

Es sei weiterhin angemerkt, daß in der Palette (im unteren Teil der Abbildungen) in der endgültigen Anwendung vermutlich Piktogramme zur besseren Kennzeichnung der zur Auswahl stehenden Objekte verwendet werden sollten. Das Zeichnen von Piktogrammen kann durch Spezialisierung der Darstellungsmethoden für Palettenobjekte erfolgen. Piktogramme sind jedoch nicht Gegenstand der Untersuchungen zu HAMVIS, so daß ich hier absichtlich auf eine optische „Verschönerung“ verzichtet habe und eine textuelle Standarddarstellung verwende.

Im linken Teil des Graphikfensters wird neben der Flugzeugkabine als weiteres Referenzsystem automatisch ein Koordinatensystem eingeblendet, das den Ursprung kennzeichnet. In der Anwendungsdomäne liegt der Koordinatenursprung der Weltkoordinaten vor der Nase des Flugzeugs.

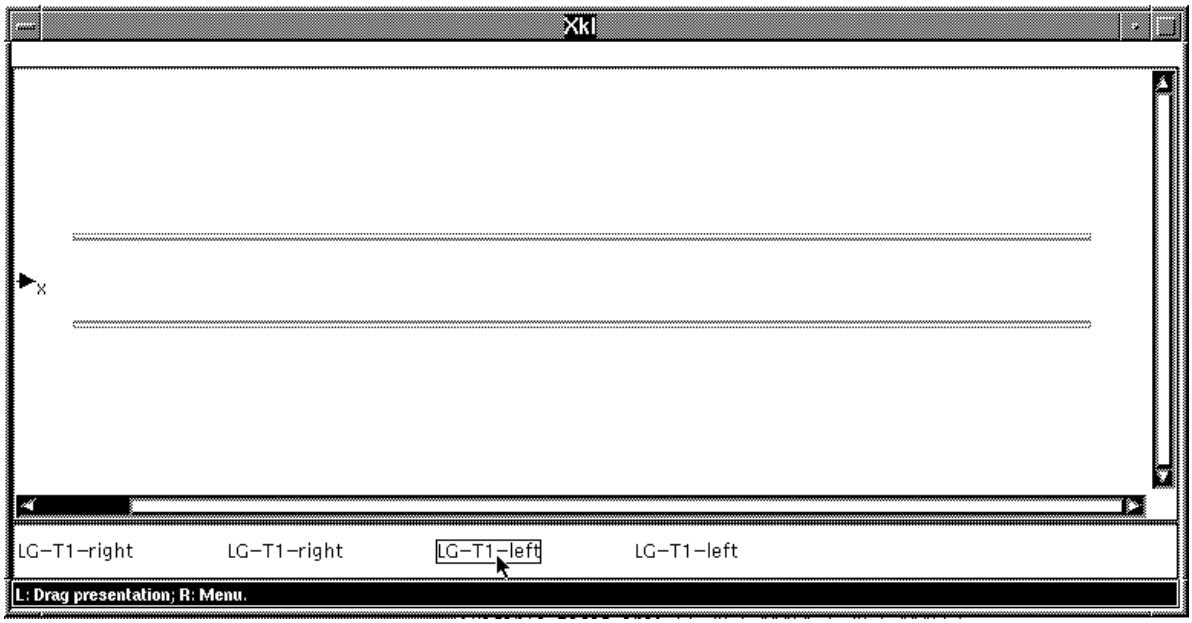


Abbildung 145. Der Anwendungsrahmen für XKL mit den beiden vorgesehenen Teilfenstern (ein Graphikfenster und eine Auswahlpalette). Die Objekte in der Palette (unten) sind mausssensitiv. CLIM beschreibt automatisch die möglichen Mausgesten in der schwarzen Mausdokumentationszeile.

Abbildung 146. Nach dem Anklicken von LG-T1-left erscheinen die möglichen Plazierungsbereiche, und die Küche kann auf einen der Bereiche geschoben werden. CLIM zeichnet automatisch ein gestricheltes Rechteck, um das gezogene Objekt zu kennzeichnen.

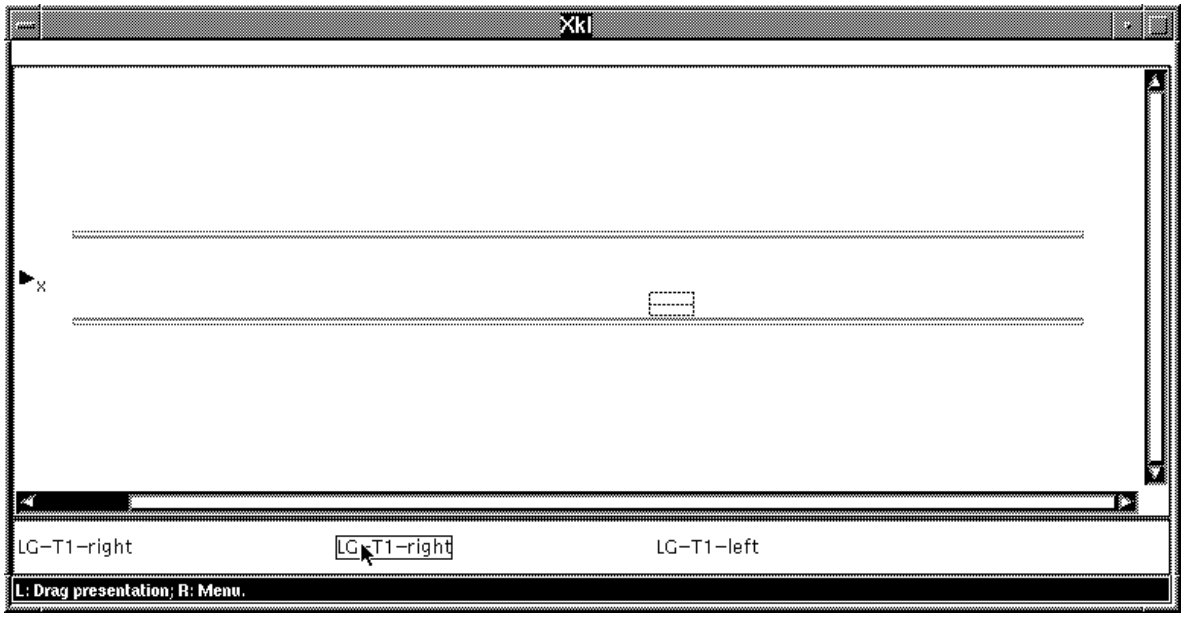


Abbildung 147. Nachdem die Küche auf den mittleren Plazierungsbereich geschoben und der linke Mausknopf gedrückt wurde, erscheint das plazierte Anwendungsobjekt. Ein neues Objekt kann aus der Palette ausgewählt werden.

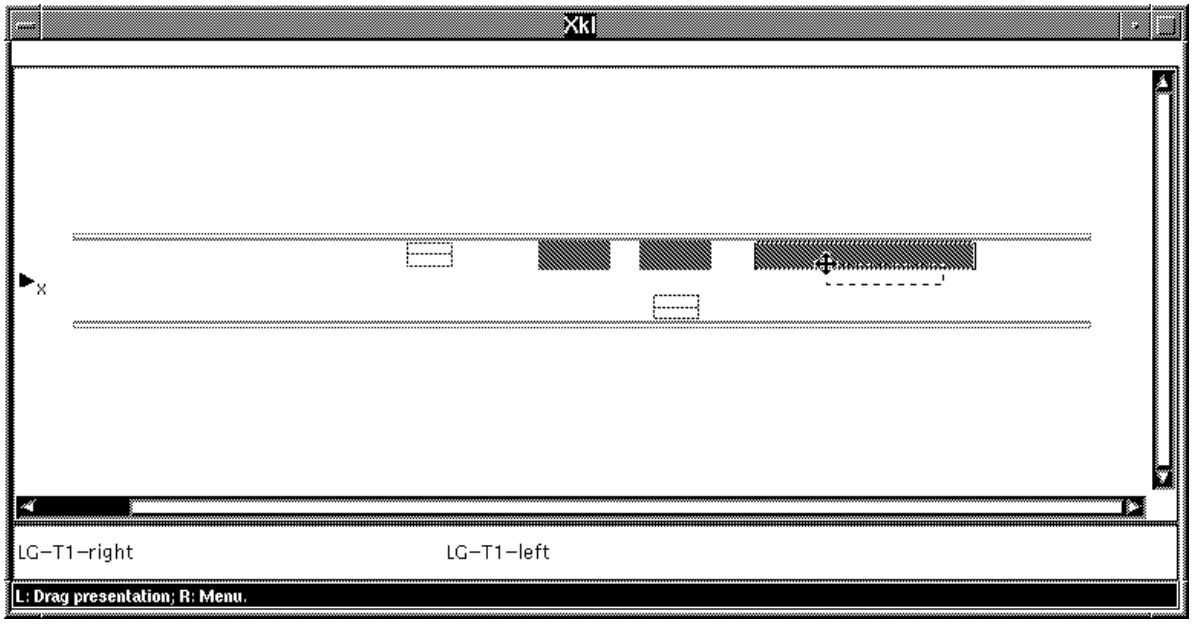


Abbildung 148. Zustand der XKL-Anwendung, nachdem einige Objekte in der Kabine plaziert wurden.

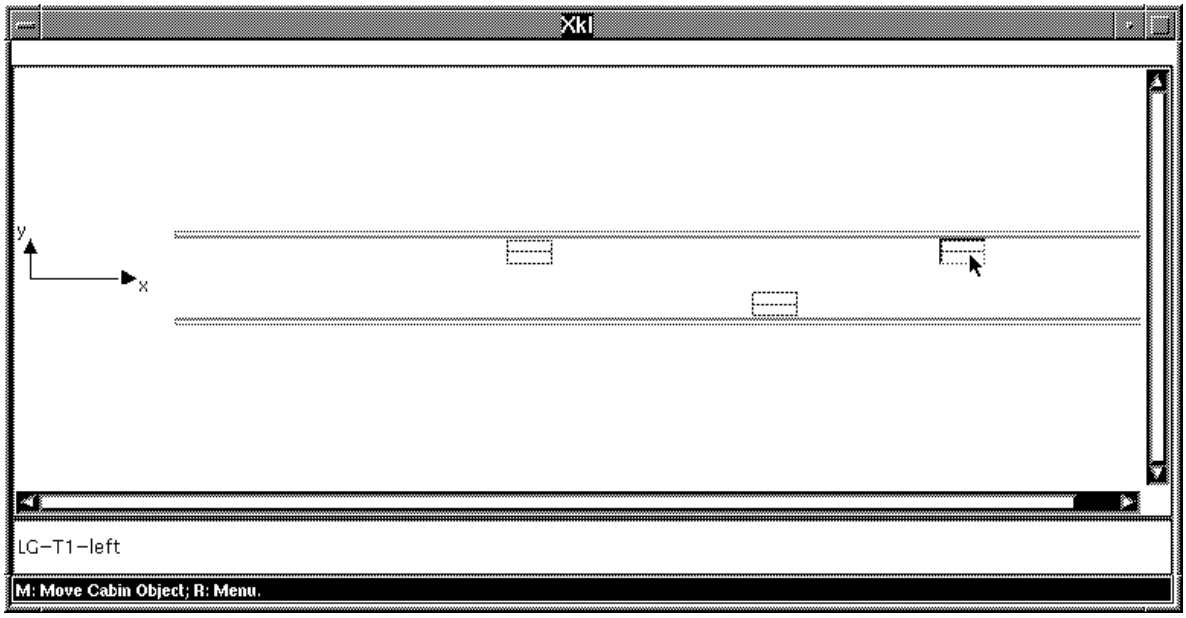


Abbildung 149. Plazierte Objekte sind maussensitiv und können verschoben werden (siehe die Mausdokumentationszeile).

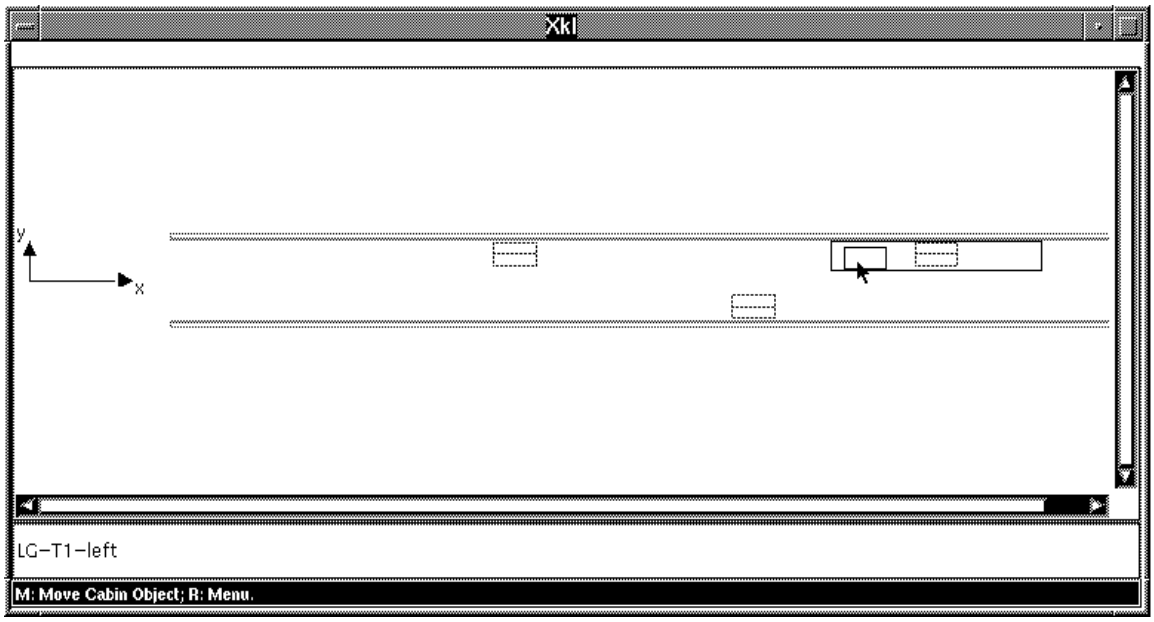


Abbildung 150. Eine Verschiebung ist jedoch nur innerhalb des rechteckigen Verschiebungsbereichs möglich, der automatisch durch Berechnungsfunktionen ermittelt wurde.

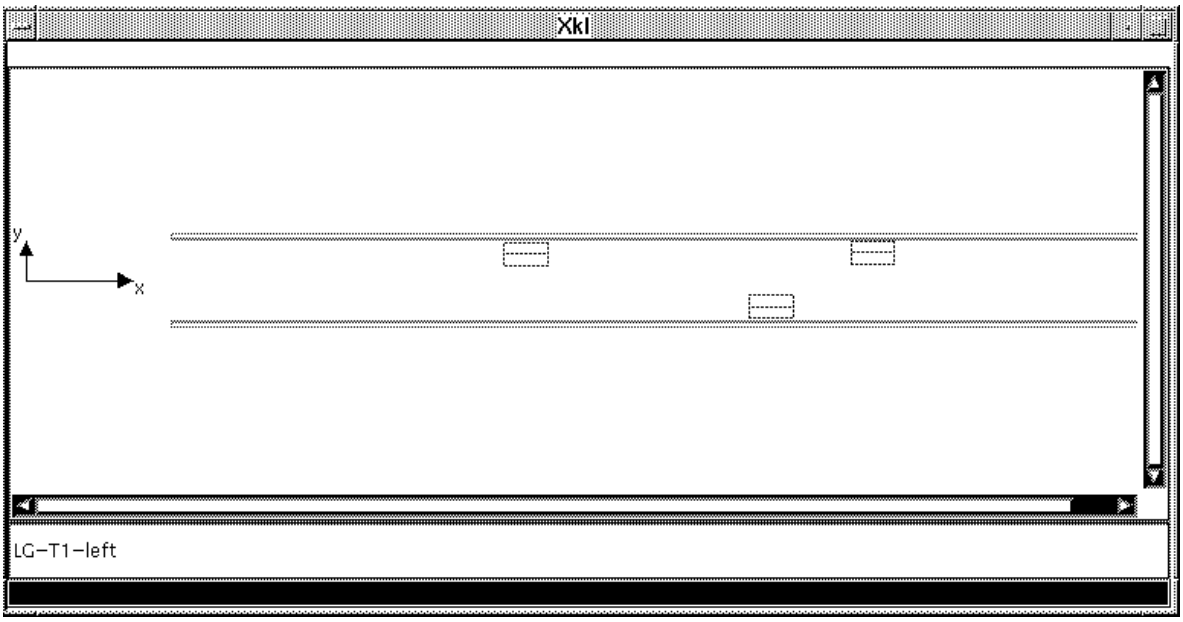


Abbildung 151. Nach der Verschiebung erscheint das bewegte Objekt an seinem neuen Platz.

Die Beispiele in den Abbildungen belegen, daß mit HAMVIS Anwendungen erstellt werden können, bei denen visuelle Darstellungen zur Realisierung einer direktmanipulativen Interaktionsform benötigt werden. Durch die Konzeption von HAMVIS mit einer Vorwegnahme von vielen Entscheidungen zur Entwicklungszeit kann ein gutes Laufzeitverhalten der Endanwendung erzielt werden.

Im folgenden möchte ich kurz auf die technischen Aspekte der Realisierung des Laufzeitsystems mit CLIM eingehen.

Abbildung einer „Applikation“ auf CLIM-Anwendungsrahmen

Für eine konkrete Anwendung muß z.B. eine Deklaration für einen Anwendungsrahmen (frame) erstellt werden. In der Form zur Deklaration des Anwendungsrahmens werden die Teilfenster (panes) und ihre jeweiligen Klassen vorgegeben (siehe Abbildung 152).

Abbildung von „Aktivitäten“ auf Layoutkonfigurationen von Teilfenstern

Für jede Aktivität wird ein spezielles Layout zur Anordnung der vorgesehenen Teilfenster angegeben. Zur Verwaltung der Benutzeraktionen zur Laufzeit wird für eine Aktivität eine Kommandotabelle deklariert. Zur Illustration der für die XKL-Anwendung resultierenden Deklarationsformen wird in den Abbildung 152 ein Beispiel gezeigt. Dieses Beispiel und die weiteren Beispiele mit CLIM-Code sollen andeuten, daß die Generierung des *anwendungsspezifischen* Laufzeitsystems bei Verwendung eines mächtigen UIMS wie z.B. CLIM nicht übermäßig aufwendig ist. Weiterhin zeigt Abbildung 152 die für die Teilfenster verwendeten Klassen (aus den UIMS-Erweiterungen von HAMVIS) und verdeutlicht, das für jedes Teilfenster ein bestimmtes anwendungsspezifisches Modell verwendet wird.

```
(clim:define-command-table construct-cabin :menu nil)
(clim:define-application-frame xkl
      (hamvis-frame)
  ()
  (:default-initargs
   :runtime-nets-for-layout-configurations
      '((construct-cabin #<runtime-net #x199fd8a>))
   :style (make-instance 'light-background-style)
   ...)
  (:command-table (xkl :inherit-from (construct-cabin) :menu nil))
  (:panes
     (pane-6/construction-floor-section-model
      (clim:make-clim-stream-pane
        :type 'hamvis-display-pane
        :name 'pane-6/construction-floor-section-model
        :initial-cursor-visibility :inactive
        :model (find-model 'construction-floor-section-model)
        :places ...
        :scroll-bars :both))
     (pane-2/floor-section-model
      (clim:make-clim-stream-pane
        :type 'singe-choice-palette-pane
        :name 'pane-2/floor-section-model
        :initial-cursor-visibility :inactive
        :model (find-model 'floor-section-model)
        :places ...
        :scroll-bars nil))
     (pointer-documentation-pane
      (clim:make-clim-stream-pane
        :type 'clim:pointer-documentation-pane
        :foreground clim:+white+
        :background clim:+black+
        :text-style (clim:make-text-style :sans-serif :bold :small)
        :scroll-bars nil
        :min-height '(1 :line)
        :max-height '(1 :line)
        :height '(1 :line))))
  (:layouts
     (construct-cabin
      (clim:vertically ()
         pane-6/construction-floor-section-model
         pane-2/floor-section-model
         pointer-documentation-pane))))
```

Abbildung 152. Definition des Anwendungsrahmens für XKL. Die Klassen der Fenster und die in den Teilfenstern verwendeten Modelle sind hervorgehoben.

Das Layout der Teilfenster wird in der aktuellen Version von HAMVIS willkürlich gesetzt. Teilfenster werden einfach untereinander gesetzt (siehe Kapitel 3.3.13).

Realisierung von zusammengesetzten Aktionen durch Kommandos

Die Realisierung von zusammengesetzten Aktionen erfolgt durch Kommando-Deklarationen, wobei die Speicherfunktionen (elementare Aktionen ohne Ausgangsparameter) einer zusammengesetzten

Aktion den Rumpf der Kommandodefinition bilden. Abbildung 153 zeigt die Kommandodefinition für die zusammengesetzte Aktion `move-cabin-object`.

```
(progn (clim:define-command
        (com-move-cabin-object :menu nil
                               :name nil
                               :command-table construct-cabin)
           ((has-moved-entity '(and non-overlapped-spatial-object
                                    cabin-object))
            (has-new-position-constraint
                '(and position-constraint-with-default
                      bounding-rectangle-position-constraint-with-xy-default)))
          (put-parameters-into-places
             '(#<place moved-object<->has-moved-entity @ #x199fe2a>
               #<place new-position<->has-new-position-constraint @ #x199fe3a>)
          (list has-moved-entity has-new-position-constraint)))

       (clim:define-presentation-to-command-translator
         com-move-cabin-object-translator
         (movable-object com-move-cabin-object
          construct-cabin :gesture :describe)
         (object)
         (let* ((application-object (first object))
                (transition (second object))
                (display-environment
                 (display-environment
                   '#<place-with-visual-information
                      has-moved-entity<->movable-objects @ #x199fdea>))
                (result
                 (realize-user-action transition
                   application-object display-environment)))
           (arrange-result result transition))))
```

Abbildung 153. Kommando und Präsentation-nach-Kommando-Abbildung für die zusammengesetzte Aktion `move-cabin-object`.

Aktivierung von Kommandos durch Präsentation-nach-Kommando-Abbildungen

Ein bewegliches Objekt wird durch den Ausgabenverwalter unter dem Präsentationstyp `movable-object` ausgegeben (der Präsentationstyp ist beim Aktionenkonzept vermerkt). Für die Auslösung eines Kommandos durch Interaktionsgesten wird eine Präsentation-nach-Kommando-Abbildung erzeugt (siehe die Deklaration `clim:define-presentation-to-command-translator` in Abbildung 153). Objekte, die mit dem Präsentationstyp `movable-object` ausgegeben worden sind, sind durch CLIM automatisch maussensitiv (siehe das Beispiel in Abbildung 149). Wenn sich der Mauszeiger auf der graphischen Präsentation eines solchen Anwendungsobjekts befindet, werden die möglichen Kommandos und ihre Auslösung automatisch in der Mausdokumentationszeile dokumentiert (und sind ggf. auch über ein Aufklappmenü zugängig). In dem Beispiel aus Abbildung 153 wird die Präsentation-nach-Kommando-Abbildung durch Mittelklick aktiviert. Das angeklickte Objekt der Anwendung wird zusammen mit der Aktionstransition als Tupel an den Aktualparameter `object` übergeben. Innerhalb des Rumpfes der Präsentation-nach-Kommando-Abbildung ist dadurch das manipulierte Objekt und die Transition des Laufzeitnetzes verfügbar (`application-object` bzw. `transition`). Mit der Transition des Laufzeitnetzes ist die Realisierungsfunktion der Aktion verbun-

den (siehe Kapitel 3.5.1). Der Aufruf dieser Realisierungsfunktion erfolgt innerhalb von `realize-user-action` (siehe Abbildung 153). Die Ergebnisse des Aufrufs werden sortiert (`arrange-result`) und als Wert der Präsentation-nach-Kommando-Abbildung zurückgegeben. CLIM übergibt die Werte dann automatisch als Parameter an das Kommando (s.o.) und führt den Kommandorumpf aus.

In Abbildung 154 werden die CLIM-Deklarationen zur Umsetzung der zusammengesetzten Aktion `create-cabin-object` gezeigt.

```
(progn (clim:define-command
         (com-create-cabin-object :menu nil
                                  :name nil
                                  :command-table construct-cabin)
           ((has-new-position-constraint
             '(and xy-bounding-rectangle-position-constraint))
            (has-localized-entity '(and cabin-object)))
          (put-parameters-into-places
            '(#<place new-position-constraint<->has-new-position-constraint
                @ #x199fdca>
              #<place-with-visual-information
                localized-object<->has-localized-entity @ #x199fdda>)
            (list has-new-position-constraint has-localized-entity)))

       (clim:define-presentation-type #:dragging-source2439 nil
          :inherit-from 'dragging-source-presentation-type)

       (clim:define-presentation-type #:dragging-destination2440 nil
          :inherit-from 'dragging-destination-presentation-type)

       (clim:define-drag-and-drop-translator
          com-create-cabin-object-translator
          (#:dragging-source2439 clim:command #:dragging-destination2440
           construct-cabin)
          (object destination-object)
          (let* ((application-object
                  (first destination-object))
                 (transition (second destination-object))
                 (display-environment
                  (display-environment
                    '#<place-with-visual-information
                       choice-set<->prototype-objects @ #x19a2202>))
                 (result
                  (realize-user-action transition
                    application-object display-environment)))
            (cons 'com-create-cabin-object
                  (arrange-result result transition)))))
```

Abbildung 154. Kommando und Ziehen-und-Fallenlassen-Abbildung
für die zusammengesetzte Aktion `create-cabin-object`.

Der Kommandorumpf wird so gestaltet, daß die durch eine Benutzeraktion gelieferten Werte in die vorgesehenen Stellen im Laufzeitnetz gelegt werden (`put-parameters-into-places`). Für `create-cabin-object` wird eine spezielle Präsentation-nach-Kommando-Abbildungsfunktion definiert: eine Ziehen-und-Fallenlassen-Abbildung. Die Ziehen-und-Fallenlassen-Abbildung ist ähnlich aufgebaut wie die Präsentation-nach-Kommando-Abbildung aus Abbildung 153, faßt aber im

Prinzip zwei Interaktionsgesten und damit auch zwei Benutzeraktionen zusammen. Wie in Kapitel 3.3.11 erläutert, ist dieses nur unter ganz bestimmten Bedingungen möglich. Für die zusammengesetzte Aktion `create-cabin-object` sind die Bedingungen allerdings erfüllt (siehe die Erläuterung der Schlußfolgerungen in Kapitel 3.3.11). Die erste Aktion `select-cabin-object` wird durch den ersten Mausklick realisiert, das Fallenlassen auf das Zielobjekt realisiert die zweite Benutzeraktion der zusammengesetzten Aktion (in diesem Fall also `localize-cabin-object`).

Bei der Codegenerierung wird dem Ausgabenverwalter der Palette noch der Präsentationstyp für die Ausgaben zugeordnet (`#:dragging-source2439`). Da eine entsprechende Ziehen-und-Fallenlassen-Abbildung mit diesem „Start-Präsentationstyp“ definiert wurde, ist also durch CLIM ein Objekt in der Auswahlpalette automatisch maussensitiv. Dieses wird an der Oberfläche durch ein kleines Rechteck angezeigt (siehe Abbildung 145). Wenn auf ein Palettenelement geklickt wird, aktiviert CLIM die Ziehen-und-Fallenlassen-Abbildung, d.h. es wird automatisch ein gestricheltes Rechteck gezeichnet, das der Mausbewegung folgt. Der Rumpf der Ziehen-und-Fallenlassen-Abbildung wird evaluiert, wenn auf ein Objekt geklickt wird, das mit dem Zielpräsentationstyp ausgegeben wurde. Graphische Objekte, die als Zielobjekte der Ziehen-und-Fallenlassen-Abbildung dienen können, werden ebenfalls durch CLIM automatisch hervorgehoben (siehe Abbildung 146).[111]

Als Wert der Ziehen-und-Fallenlassen-Abbildung wird ein Kommandoobjekt zurückgegeben. Ein Kommandoobjekt wird als Liste bestehend aus Kommandonamen und aktuellen Parametern repräsentiert (siehe die `cons`-Anweisung am Ende der Definition in Abbildung 154). CLIM sorgt dafür, daß die entsprechende Kommandodefinition aus der Kommandotabelle herausgesucht und mit den angegebenen Aktualparametern aufgerufen wird.

3.5.4 Zusammenfassung: Abbildung auf CLIM

In diesem Kapitel wurde ein kurzer Abriß über die technische Anbindung eines UIMS an HAMVIS anhand des Beispiels von CLIM gegeben. Viele Dienste, die bei anderen Autoren Gegenstand der Aufgaben- und Aktionenmodellierung sind (vgl. die Ausführungen in Kapitel 2.2.3) sind im HAMVIS-Kontext durch das zugrundliegende UIMS abgedeckt (bzw. abzudecken falls ein anderes UIMS als CLIM verwendet werden soll).

In der aktuellen Version des Zielsystemübersetzers sind noch nicht alle möglichen Kombinationen von Benutzeraktionen erfaßt. Es lassen sich komplexere Präsentation-nach-Kommando-Abbildungsfunktionen synthetisieren. Hierzu müssen auch für zusammengesetzte Aktionen ggf. noch Parameter zur detaillierteren Beschreibung eingeführt werden. Die bestehende Funktionalität reicht jedoch aus, um das Grundprinzip der CLIM-Übersetzung zu verdeutlichen.

---

111. Zu beachten ist noch, daß die Details der Mausbehandlung bei einer Ziehen-und-Fallenlassen-Interaktionsgeste durch das Wirtssystem festgelegt werden. In dieser Arbeit wurde CLIM in einer Motif-Umgebung eingesetzt. Ziehen-und-Fallenlassen-Gesten werden durch zwei Mausklicks realisiert. Während der Geste wird kein Mausknopf gedrückt. Wenn CLIM in einer anderen Umgebung eingesetzt wird, so wird evtl. eine Ziehen-und-Fallenlassen-Abbildungsfunktion ebenfalls dadurch ausgelöst, daß eine Maustaste gedrückt wird. Während der Interaktionsgeste muß aber die Maustaste gedrückt gehalten werden. Wenn der Mauszeiger auf ein Zielobjekt zeigt, kann der Mausknopf losgelassen werden. Diese Unterschiede werden aber auf der Ebene von CLIM berücksichtigt und brauchen nicht von HAMVIS berücksichtigt zu werden.

3.6 Zusammenfassung: Entwurf von Visualisierungen für Benutzungsoberflächen mit HAMVIS

Am Anfang der Arbeit zu HAMVIS wurden für die Anwendungsdomäne XKL bestehende handgezeichnete Visualisierungen aus einem Handbuch des Flugzeugherstellers (Cabin Configuration Guide) analysiert (siehe auch die Beispiele aus der Einleitung). Den handgezeichneten Darstellungen lag jeweils eine bestimmte Sicht auf die Dinge der realen Welt zugrunde. Auffällig war, daß im ganzen Buch nur wenige Sichten verwendet wurden. Mit der geometrischen Sicht war immer auch eine begrifflich-logische Sicht der Weltmodellierung verbunden (insbesondere Teil-von-Beziehungen). Es wurden z.B. Detaildarstellungen und überblicksartige Darstellungen präsentiert. In den Übersichtsdarstellungen werden z.B. bestimmte Objekte, die in detaillierteren Darstellungen immer vorkommen, grundsätzlich nicht verwendet. Zu beachten ist, das Objekte nicht wegen der Größe weggelassen werden, sondern weil sie für den Zweck, den eine Visualisierungssicht erfüllen sollte, nicht erforderlich waren.

Aus der Analyse des Cabin Configuration Guide reifte der Gedanke, den Prozeß der Visualisierungsgenerierung bzw. -komposition als Auswahlprozeß über Sichten zu definieren. Eine Sicht im Sinne von HAMVIS wird als *Modell* bezeichnet und verknüpft über die Modellklasse die geometrische Modellierung von Objekten mit der begrifflich-logischen bzw. „funktionalen“ Modellierung durch Konzept- und Rollendefinitionen. Anstatt also den Inhalt (zur Laufzeit) durch Schlußfolgerungsprozesse nach gewissen Kriterien auszuwählen, wird durch HAMVIS (zur Entwicklungszeit) ein Modell bestimmt, das eine Weltmodellierung mit der erforderlichen Granularität vornimmt. Oder, anders gesagt, es werden zur Entwicklungszeit Anforderungen zur Bildung von Modellen abgeleitet. Die Anforderungen ergeben sich aus den Aktionen, die durch Visualisierungen unterstützt werden sollen. HAMVIS geht also nicht von einer festen Weltmodellierung aus, sondern leitet während der Generierungsphase ab, ob schon deklarierte Modelle für notwendige Benutzeraktionen verwendet werden können, ob bestehende Modelle modifiziert werden müssen oder ob neue Modelle für zu unterstützende Benutzeraktionen deklariert werden sollten. Die in den vorigen Kapiteln vorgestellte Formalisierung dieser Inferenzschritte basiert insbesondere auf dem „Weglassen von Informationen“ in einem Modell (siehe die Ausführungen zur Sichtbarkeit von Konzepten und zur Verfeinerung von Konzepten in einem Modell). Für die Visualisierungsgenerierung wurde eine Charakterisierung von Konzepten eingeführt (Basiskonzepte, Primärkonzepte, kategoriale Oberkonzepte). Basiskonzepte bilden die Grundlage für die zweidimensionale Darstellung von Objekten in einer Visualisierung. Die Wirkung des „Weglassens von Informationen“ besteht in einer unterschiedlichen Charakterisierung von Konzepten in verschiedenen Modellen, d.h. in verschiedenen Modellen können unterschiedliche Basiskonzepte auftreten und dadurch unterschiedliche Objekte dargestellt werden. Die Charakterisierung von Konzepten basiert auf der Verwendung von Konzepten und Relationen aus einem HAMVIS-Grundmodell.

Nach den Generierungsbeispielen und den technischen Ausführungen zur Realisierung des HAMVIS-Systems in diesem Kapitel möchte ich zur Zusammenfassung einen Grobüberblick über die einzelnen Phasen der Entwicklung einer Oberfläche mit HAMVIS geben und noch einmal das Zusammenspiel der Teilmoduln erläutern (siehe die Übersicht in Abbildung 52.). Ein Vergleich des Ansatzes mit anderen Arbeiten und eine Bewertung des Beitrags von HAMVIS zur Forschung wird in Kapitel 4 vorgenommen. In Abbildung 155 wird die Verzahnung von anwendungsspezifischen und vordefinierten TBox-Teilmodellen aufgezeigt (weiße bzw. graue Kästen mit dickerem Rand). Anwendungsspezifi-

sche Strukturen, die mittels ABox-Assertionen definiert werden, sind durch einen dünneren Rand gekennzeichnet.

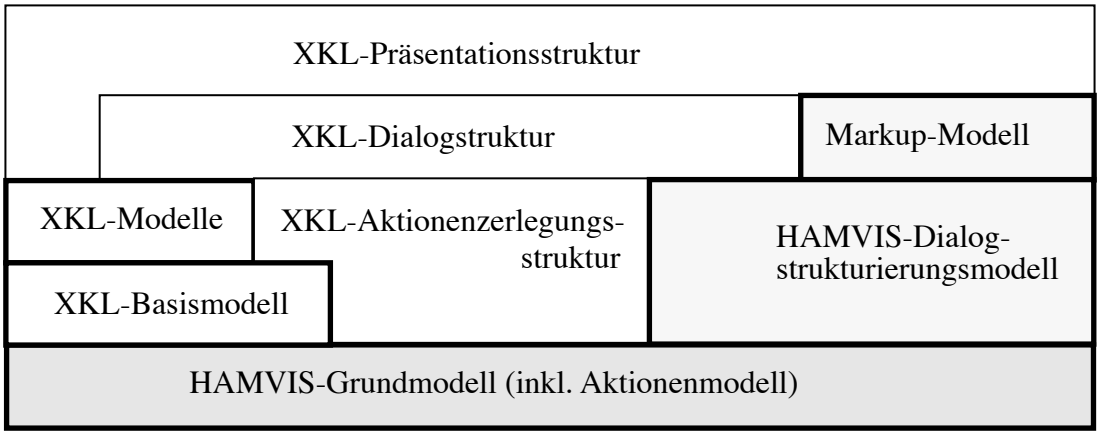


Abbildung 155. Verzahnung und Schichtung von Modellen innerhalb des HAMVIS-Szenarios. Höhere Schichten bauen auf unteren Schichten auf.

Unter Rückgriff auf das HAMVIS-Grundmodell wird durch das Systementwicklungsteam zunächst ein Anwendungsmodell unter Verwendung von Konzepten bzw. Rollen des HAMVIS-Grundmodells definiert (genannt Basismodell der Anwendung, siehe Abbildung 155). Auf der Basis dieses Modells, das eine dreidimensionale geometrische Weltmodellierung unterstützt, wird eine *anwendungsspezifische* Aktionen*zerlegung* festgelegt.

Die Aktionenzerlegung wird mit einer interaktiven Oberfläche unter Verwendung des HAMVIS-Aktionenmodells durchgeführt. Aktionen des Benutzers (genauer: Aktionsmöglichkeiten des Benutzers zur Laufzeit) werden durch das Systementwicklungsteam mit Hilfe von vordefinierten Konzepten aus dem HAMVIS-Aktionenmodell beschrieben. Aktionsmöglichkeiten des Benutzers müssen so genau beschrieben werden, daß eine Visualisierungsgenerierung möglich wird. Die Beschreibung von Aktionen erfolgt durch schrittweise Verfeinerung (Spezialisierung) von Aktionenkonzepten.

Das Aktionenzerlegungsmodell einer Anwendung strukturiert die Anwendung und legt zudem fest, welche Objekte dem Benutzer mindestens präsentiert werden müssen, damit er die Aktionen zur Laufzeit durchführen kann (manipulierte Objekte). Da die Objekte zur Entwicklungszeit nicht als Instanzen bekannt sind, wird ein Formalismus zur Beschreibung von konzeptuellen Informationen benötigt. Dieses war eine wesentliche Motivation zur Verwendung des beschreibungslogischen Formalismus.

Wir haben weiterhin gesehen, daß sich die Abhängigkeiten zwischen Benutzeraktionen und Berechnungs- und Speicherfunktionen über ein Petri-Netz beschreiben lassen, das aus der Aktionenzerlegung abgeleitet werden kann. Das Petri-Netz wird automatisch aus dem Aktionenmodell abgeleitet. Stellen im Petri-Netz enthalten zur Laufzeit Mengen von Anwendungsobjekten. Transitionen entsprechen entweder Benutzeraktionen oder Anwendungsfunktionen (Berechnungs- und Speicherfunktionen). Das Verhalten einer Anwendung in Bezug auf die Evaluierung von Anwendungsfunktionen ist durch den Petri-Netz-Formalismus eindeutig beschrieben. Funktionen werden dann evaluiert, wenn ihre Eingangsstellen belegt und die Ausgangsstellen frei sind. Benutzeraktionen verändern nicht (direkt) den Zustand der Welt sondern liefern neue Werte, die dann in die Ausgangsstellen der korrespondierenden Transitionen gelegt werden.

Neben den für eine Anwendungsklasse vordefinierten Aktionenkonzepten stellt HAMVIS zur Entwicklungszeit eine Reihe von Inferenzdiensten zur Modellierung einer Aktionenzerlegung bereit. Durch die Propagierung von Objektkonzepten innerhalb des Petri-Netzes wird aus der Spezifikation der „Typen“ der Parameter und Werte von Berechnungs- bzw. Speicherfunktionen zur Entwicklungszeit abgeleitet, ob die Objekte in den Ein- oder Ausgangsstellen von Transitionen, die für Benutzeraktionen stehen, überhaupt zu den Aktionen „passen“ (Konsistenzprüfung).

Durch die Typpropagierung läßt sich eine partielle Beschreibung einer vom Systementwicklungsteam gewünschten Benutzeraktion automatisch weiter einschränken (Schlüsse über definierte Aktionenkonzepte). Falls eine automatische Spezialisierung nicht möglich ist, können speziellere Aktionskonzepte mit Hilfe der interaktiven Oberfläche ausgewählt werden. Durch Propagierung der Konzepte entlang der Kanten des Laufzeitnetzes werden bei Auswahl eines möglichen Aktionenkonzeptes auch die Parameter und Werte von Berechnungs- und Speicherfunktionen weiter eingeschränkt. Dadurch wird erreicht, daß die „Typsignatur“ von Anwendungsfunktionen automatisch an die Erfordernisse der Oberflächengestaltung angepaßt wird.

Über die Konzepte der Benutzeraktionen aus dem HAMVIS-Aktionenmodell ist definiert, welche Stellen Objekte enthalten, die an den Ausgabenverwalter übermittelt werden müssen. Weiterhin sind durch die Konzepte der Benutzeraktionen Anforderungen bzw. Einschränkungen für die Darstellungsform der in den Stellen auftretenden Objekte schon zur Entwicklungszeit der Anwendung festgelegt. Einschränkungen sind entweder durch erforderliche Modellklassen gegeben oder über Konzepte definiert, die für die Objekte in den entsprechenden Stellen in mindestens einem Modell gelten müssen.

Modelle der erforderlichen Modellklassen können durch die von HAMVIS bereitgestellten Erweiterungen zu KRSS-CLASSIC definiert werden (siehe Abbildung 155). HAMVIS bestimmt für jedes Modell die entsprechenden Konzeptcharakterisierungen, berechnet auf dieser Grundlage die jeweiligen Objektzerlegungsgraphen (Zerlegung von Basiskonzepten in Basiskonzepte) und zeigt die Ergebnisse auf Wunsch graphisch an. Für die Anwendung können neben den in dieser Arbeit vorgestellten Modellen, die insbesondere für Visualisierungszwecke ausgelegt sind, auch Modelle verwendet werden, die keine direkte Visualisierung erlauben.

Mögliche Strukturierungen des Dialogs einer Anwendung werden aus generischem Dialogstrukturierungswissen hergeleitet. HAMVIS formalisiert dieses Wissen ebenfalls mit Hilfe der Beschreibungslogik. Eine wichtige Rolle spielen dabei die Konzepte des HAMVIS-Grundmodells und die Konzepte von Benutzeraktionen aus dem HAMVIS-Aktionenmodell. Es ergibt sich folgendes Zusammenspiel der Teilmodelle: Wenn in der Aktionenzerlegung eine bestimmte Benutzeraktion vorgesehen ist, so wird über das verwendete Beschreibungskonzept aus dem HAMVIS-Aktionenmodell festgelegt, welche Konzepte die zu präsentierenden Objekte haben müssen. Da die zu präsentierenden Objekte (zur Laufzeit) von Anwendungsfunktionen geliefert werden, müssen für die Anwendungsfunktionen entsprechende Einschränkungen für die Parameter und Werte zugesichert werden, d.h. anwendungsspezifische Konzepte müssen von bestimmten Konzepten des HAMVIS-Grundmodells subsumiert werden, damit eine Visualisierung möglich ist. Das Wissen zur Dialogstrukturierung ist nun im HAMVIS-Dialogstrukturierungsmodell in Bezug auf diese Konzepte des HAMVIS-Grundmodells spezifiziert (siehe Abbildung 155). Wir haben in diesem Kapitel gesehen, daß über den Vererbungsmechanismus der Beschreibungslogik bestimmte Modifikationsanforderungen ausgelöst werden, die durch neue Objekte erfüllt werden müssen. Die Bestimmung (bzw. Herleitung) dieser Modifikatorobjekte erfolgt

automatisch durch berechnete Assertionen, die über Ableitungen über die Struktur der Objektzerlegungsgraphen der verschiedenen Modelle der Anwendung bestimmt werden.

Mit Hilfe des Objektzerlegungsgraphen, den HAMVIS automatisch für jedes Modell der Klasse `geometry-model-class` aus den beschreibungslogischen Termen bestimmt, lassen sich die Modifikationsfunktionen als Traversierungsfunktionen der Teile-Zerlegung automatisch zusammensetzen. Es kommen für die Darstellung nur solche Modelle in Frage, die die notwendigen Modifikatoren auch bestimmbar machen. Der Modifikationsmechanismus wird durch berechnete ABox-Assertionen modelliert. Durch die hinzukommenden ABox-Assertionen wird eine hierarchische Beschreibung der Dialogstruktur aufgebaut. Wir haben gesehen, daß es notwendig ist, verschiedene Varianten zu verwalten.

Für den Schnittstellenentwickler ist eine interaktive Oberfläche vorgesehen, mit der er die möglichen Varianten vergleichen kann. Durch die interaktive Oberfläche braucht der Entwickler die internen Ableitungsmechanismen und die internen Dialogstrukturierungskonzepte der HAMVIS-Wissensbasis nicht zu kennen. Falls z.B. notwendige Modifikatoren nicht ableitbar sind, so wird dieses dem Entwickler mit Hilfe der Oberfläche angezeigt. Sollte dieses der Fall sein, so müssen neue Modelle für die Anwendung definiert oder bestehende Modelle angepaßt werden. Es wird hier auch deutlich, daß sich Anforderungen, die sich durch die Darstellung ergeben, auf die Definition von Domänenmodellen auswirken. Obwohl Darstellungsanforderungen durch bestehende Modelle eventuell schon abgedeckt werden, können dennoch neue Modelle definiert werden, in denen z.B. weniger Basiskonzepte definiert sind (siehe auch die Feiner-Relation zwischen Modellen), so daß der Entwickler die Kontrolle über den dargestellten Inhalt behält.

Die abgeleiteten Informationen zur Ausgestaltung der Präsentation müssen zur Laufzeit durch Evaluierung der Modifikationsfunktionen bestimmt werden. Die Einbindung der Modifikationsfunktionen erfolgt in dem von HAMVIS generierten erweiterten Laufzeitnetz. Bei der Herleitung der Dialogstrukturierung wird automatisch die Funktion von Teilkomponenten in einer Darstellung hergeleitet und im erweiterten Laufzeitnetz an den Stellen explizit gemacht (Assertionen zur Modellierung des Diskurszwecks und der damit verbundene Diskursposition). Auch hierzu werden beschreibungslogische Konzepte aus der HAMVIS-Wissensbasis eingesetzt. Über diese Konzepte sind Einschränkungen für die Darstellung von Teilkomponenten definiert. Je spezieller die Diskurszweckkonzepte, desto spezieller sind die Darstellungseinschränkungen.

Zur Bestimmung von Zeichenattributen werden Objekte zur Modellierung von Diskurszwecken in Markierungen umgesetzt. Markierungen bieten eine software-technische Realisierung der Einschränkungsverwaltung für die Bestimmung von Zeichenattributen und die Einbindung von UIMS-Strukturen. Wir haben gesehen, daß auch anwendungsspezifische visuelle Notationen, die durch den Schnittstellenentwickler definiert werden, in einen solchen Mechanismus integriert werden können. Durch Markierungen wird die Präsentationsstruktur einer Anwendung festgelegt. Aus dem erweiterten Laufzeitnetz, der Dialogstruktur und der Präsentationsstruktur kann der Code für ein UIMS automatisch generiert werden.

Die Beispiele aus dem vorigen Unterkapitel belegen, daß HAMVIS auch aus unvollständigen Aktionenmodellen schon Anwendungen generieren kann. Zur Unterstützung eines schnellen Tests der lauffähigen Anwendung können Berechnungs- und Speicherfunktionen prototypisch implementiert werden. HAMVIS unterstützt also eine inkrementelle Programmentwicklung, vermeidet aber durch die Strukturierung des Vorgehens die Nachteile einer unsystematischen Prototypentwicklung.

4
Erzielte Ergebnisse und mögliche Erweiterungen

Der Aufwand zur Erstellung von Oberflächen mit HAMVIS erscheint vielleicht recht hoch. Verschiedentlich wurde geäußert, daß Visualisierungen für „praktische Anwendungen“ doch aus speziell erfaßten geometrischen Daten extrahiert werden und in der (nachträglich aufgesetzten) Oberfläche „einfach in ein Fenster eingeblendet werden können“. Die Behandlung der Interaktion und die notwendige „Schnittstellenprogrammierung“ sei „schnell erledigt“. Die Praxis zeigt, daß dieses nicht der Fall ist. Nach meiner Ansicht ist die Entwicklung von Anwendungen wie XKL primär durch Gestaltungsaspekte der Benutzungsschnittstelle geprägt. In mehreren Projekten, in denen ich mitgewirkt habe, kamen von Anwendern Vorschläge für neue Dienste der Anwendung häufig aus der Perspektive der Benutzungsschnittstelle. Es kommen Wünsche für neue Aktivitäten oder neue Aktionen. Oberflächenaspekte bestimmen in nicht zu unterschätzender Weise die automatisierten Dienste einer Anwendung.

Obwohl der Anwender vielleicht nicht direkt die Terminologie der HAMVIS-Aktionenmodellierung verwendet, so lassen sich vom Systementwicklungsteam mit Hilfe der vordefinierten Aktionenkonzepte und der vorgesehenen graphischen Oberfläche schnell die notwendigen Anforderungen an Berechnungsfunktionen ermitteln. Durch die Propagierung der Konzepteinschränkungen von Aktionenkonzepten an die Parameter und Werte von Berechnungs- und Speicherfunktionen ergeben sich Spezifikationen für automatische Dienste der Anwendung. Durch die Dialogstrukturierungswissensbasen und die interaktiven Oberflächen von HAMVIS lassen sich anschließend notwendige Darstellungsobjekte und Darstellungsfenster erfassen. Es kann sich hierbei ergeben, daß für eine gewünschte, neue Funktionalität schon bestehende Anwendungsmodelle verwendet werden können. Es kann auch der Fall sein, daß zur Realisierung einer neuen Funktionalität der Anwendung neue Modelle erstellt und feinere Konzepte deklariert werden müssen.

Das Laufzeitsystem einer Anwendung wird durch ein Laufzeit-Petri-Netz spezifiziert, das aus der anwendungsspezifischen Aktionenmodellierung durch HAMVIS bestimmt wird. Durch die Festlegung von Darstellungsattributen über Markierungen wird das Laufzeitsystem vervollständigt. Wenn die für die Aktionen benötigten Anwendungsfunktionen prototypisch implementiert werden, so ist eine schnelle Rückkopplung mit dem Anwender möglich.

Die HAMVIS-Architektur ermöglicht es, den wissensbasierten Modellierungsansatz auf die Konzeption und Implementation der Gesamtanwendung anzuwenden, d.h. Oberflächenkomponenten werden nicht im nachhinein auf eine „fertige“ Anwendung aufgesetzt, sondern sind integraler Bestandteil der Anwendungskonzeption. In dieser Arbeit wurden hierzu die wesentlichen Aspekte der Spezifikation und Generierung von Visualisierungen innerhalb der Oberfläche analysiert.

4.1 Der wissenschaftliche Beitrag

Die vorliegende Dissertation ist eine anwendungsorientierte Arbeit, in der ein Rahmensystem zur Entwicklung von Oberflächen mit interaktiven, modellbasierten Visualisierungen konzipiert wurde. Ein wesentliches Problem bei der Konzeption eines Rahmensystems besteht darin, „Teilstrukturen“ festzulegen, die in koordinierter Weise bei der Lösung eines Gesamtproblems zusammenspielen. Das in den Teilmoduln verwendete Wissen muß in strukturierter Weise erweitert werden können. Weiterhin müssen „Schnittstellen“ definiert werden, so daß in das Rahmensystem anwendungsspezifische Modelle integriert werden können. Mit dem in dieser Arbeit vorgestellten Architekturansatz wurden diese Kriterien erfüllt. Die vordefinierten Modelle (Wissensbasen) von HAMVIS zeigen klar auf, in welcher Weise das Wissen für die Teilschritte strukturiert und mit Hilfe einer Beschreibungslogik formalisiert werden kann. Die Teilmodelle sind jedoch jeweils noch nicht so weit entwickelt, daß eine vollständige XKL-Anwendung schon mit dem HAMVIS-Prototypen realisiert werden kann.

Ein zentraler Beitrag von HAMVIS ist durch die konsequente Abgrenzung von Entwicklungszeit- und Laufzeitaktivitäten gegeben. Es wurde herausgearbeitet, daß für die Visualisierungsgestaltung und die Oberflächengestaltung allgemein durch Betrachtung der gesamten Aktionen- und Dialogstruktur wesentliche Gestaltungsmerkmale einer Anwendung schon zur Entwicklungszeit festgelegt werden können. Grundlegend hierfür waren die durch beschreibungslogische Schlüsse formalisierten Ableitungen mit konzeptuellen Informationen über Anwendungsobjekte. Mit der Prototypimplementation von HAMVIS wurde die Tragfähigkeit des entwickelten Ansatzes aufgezeigt.

In der Arbeit wurden theoretische und praktische Ergebnisse aus verschiedenen Forschungsrichtungen berücksichtigt. Innerhalb des HAMVIS-Rahmensystems wurden sowohl praktische, softwaretechnische Arbeiten aus den Bereichen Geometriemodellierung, Programmierung von Benutzungsschnittstellen, KI-Repräsentationssprachen als auch eher theoretische Arbeiten aus den Bereichen Mensch-Computer-Interaktion, Automatische Generierung von graphischen Präsentationen und Intelligente Multimedia Präsentationssysteme integriert. Theorien z.B. aus der kognitiven Psychologie dienen als „Ideenspender“ (siehe z.B. die Konzeptcharakterisierungen aus Kapitel 3.1.6) wurden aber im Sinne der ingenieursorientierten KI formalisiert und in den Kontext des HAMVIS-Rahmenwerks integriert. HAMVIS liefert für die angesprochenen Forschungsgebiete seinerseits theoretische und praktische Beiträge. Obwohl sich theoretische und praktische Arbeiten ergänzen, möchte ich die jeweiligen Teilaspekte getrennt verdeutlichen.

4.1.1 Die theoretischen Beiträge

Mit der in HAMVIS realisierten Aktionenmodellierung wurde gezeigt, daß es möglich ist, durch beschreibungslogische Schlußverfahren zur Entwicklungszeit Inferenzen über mögliche Benutzeraktionen zu formalisieren. Durch die konzeptuellen Informationen über die für die Aktionen relevanten Anwendungsobjekte können grobe Aktionenschemata zur Entwicklungszeit automatisch verfeinert und auf die zur Laufzeit generierten Anwendungsobjekte abgestimmt werden. Durch schrittweise Verfeinerung der Konzepte von Benutzeraktionen können die Dienste einer UIMS-„Infrastruktur“ (für eine bestimmte Anwendungsklasse) systematisch zugängig gemacht werden.

Das HAMVIS-Grundmodell greift aktuelle Teilfragen der KI-Forschung auf (z.B. Teil-Ganzes-Relationen, Kopplung von geometrischen Berechnungen und begrifflichen Inferenzen) und bietet für die für HAMVIS relevanten Teilaspekte eine beschreibungslogische Formalisierung. Es wurde weiterhin

aufgezeigt, wie beschreibungslogische Repräsentationsformalismen angewendet und erweitert werden können, um multiple Modelle zu verwalten. Die Verwaltung von multiplen Modellen ist nicht nur zur Organisation der Oberflächeneinheiten wichtig, sondern etabliert sich als notwendige Basistechnik in verschiedenen Anwendungskontexten für komplexe wissensbasierte Systeme. In HAMVIS wurde insbesondere die Behandlung von geometrischen Daten (sog. projektive Formen) und begriffliche Teil-von-Zerlegungen gekoppelt. Aggregierungsprobleme wurden bei der Herleitung der Dialogstruktur im Bottom-up-Verfahren behandelt. HAMVIS zeigt damit, daß zum Aufbau einer Dialogstruktur nicht nur ein Top-Down-Ansatz mit Planoperatoren bzw. Präsentationszielen oder Schemata geeignet ist. Anstatt allgemeine, kombinatorische Aggregierungsprinzipien zu realisieren, wurde domänenspezifisches Wissen über die erweiterten beschreibungslogischen Terme mit K-Operatoren und anwendungsspezifischen Funktionen integriert. Die Arbeit zeigt auf, daß Beschreibungslogiken sich – entsprechend erweitert – sehr gut eignen, die wesentlichen Inferenzschemata adäquat zu formalisieren.

HAMVIS behandelt im Gegensatz zu modellbasierten Schnittstellenwerkzeugen (in formaler Weise) die Komposition von modellbasierten Visualisierungen für Anwendungsobjekte und definiert nicht nur eine Menge von „Werkzeugen“, die auf expliziten Datenstrukturen operieren.

Die in dieser Arbeit vorgestellte Konzeption der Architektur des Rahmensystems wurde durch modellbasierte Schnittstellenwerkzeuge und IMMP-Systeme motiviert. Als Ergänzung zu IMMP-Systemen wurde untersucht, wie sich die direktmanipulative Realisierung von Benutzeraufgaben auf die Gestaltung und Komposition der Darstellungen auswirkt (interaktive Visualisierungen). Die strikte Trennung von Entwicklungszeit- und Laufzeitaktivitäten ermöglicht es, zur Entwicklungszeit möglichst viel Wissen für die Oberflächenkonzeption auszunutzen. In dieser Arbeit werden modellbasierte Schnittstellenbaukästen und IMMP-Systeme nicht als Gegensatz betrachtet, sondern als sich ergänzende Perspektiven angesehen. Es wäre sehr interessant, für die Laufzeitaktivitäten ein IMMPS zu integrieren, das auf die zur Entwicklungszeit getroffenen Designentscheidungen abgestimmt ist (siehe Abschnitt 4.2).

4.1.2 Die praktischen Beiträge

Die Konzeption der HAMVIS-Architektur wurde durch die schrittweise Entwicklung der Prototypimplementierung stark gefördert. Insbesondere durch die Entwicklung der interaktiven Oberflächenkomponenten von HAMVIS wurden Anforderungen an die Wissensmodellierung in Bezug auf Erklärungs- und Kommunikationsmöglichkeiten von Teilergebnissen und damit auch Anforderungen an die HAMVIS-Gesamtarchitektur aufgezeigt. Es wurden dabei durch praktische Arbeiten zur KI-Softwaretechnik neben den Wissensbasen und Interaktionskomponenten von HAMVIS, die dieser Arbeit detailliert besprochen wurden, noch folgende Teilsysteme erstellt, die auch in anderen Anwendungskontexten eingesetzt werden können:

- Anbindung des logischen Programmiersystems CLP(R) an Common Lisp und Interpretation von Einschränkungsbescheibungen in Lösungen von CLP(R) durch eine Lisp-Schnittstelle,
- Anbindung des geometrischen CSG-Modellierers VANTAGE mit Realisierung einer CLOS-basierten objektorientierten Schnittstelle,

- Erweiterung von CLASSIC um Möglichkeiten zur Verwaltung von multiplen Modellen, K-Operatoren und objektorientierten Programmierkonzepten (generische Funktionen und Methodenkomposition für CLASSIC-Individuen und CLOS-Objekte, siehe [215]),
- Oberflächenkomponenten als Erweiterung von CLIM (Darstellung von Teilklassen von Petri-Netzen, Präsentation von geometrischen VANTAGE-Objekten).

Naturgemäß können weitere Arbeiten in das erstellte Rahmensystem HAMVIS integriert werden. In einigen Abschnitten dieser Arbeit wurde schon auf mögliche Erweiterungen hingewiesen. Ich möchte im folgenden Abschnitt die wesentlichen Aspekte zusammenfassen.

4.2 Mögliche Erweiterungen und neue Anwendungsgebiete

Das bestehende HAMVIS-System besitzt eine klare Gesamtstruktur und trägt damit zur Strukturierung der verschiedenen Teilaufgaben bei der Visualisierungsgenerierung für interaktive Oberflächen bei. Zur Vervollständigung der in dieser Arbeit aufgezeigten Funktionalität sind für die einzelnen Teilbereiche noch folgende Arbeiten möglich:

- Vervollständigung der Menge der Aktionenkonzepte für die Anwendungsklasse der Layoutprobleme,
- Ergänzung der Wissens zur Dialogstrukturierung und zum Aufbau von Visualisierungen (z.B. besondere Berücksichtigung von Landmarken),
- Weitere Abbildungen von komplexeren zusammengesetzten Handlungen auf Interaktionsgesten,
- Generierung von Layoutbeschreibungen, die auf die Einschränkungen der Dialogstruktur angepaßt werden,
- Erweiterung der Präsentationstrukturierungskomponente.
- Erweiterung der anwendungsübergreifenden Komponenten für das Zielsystem (backend) CLIM.

Weiterhin könnte auch die Funktionalität von HAMVIS noch erweitert werden. In der XKL-Anwendungsklasse beispielsweise ist die Lokalisierung von Einrichtungsgegenständen von speziellen räumlichen Konstellationen abhängig (siehe Abbildung 156). Um Platz zu sparen, ist es z.B. vorteilhaft, den vor Küchen notwendigen Arbeitsraum (für die Handhabung von Trolleys) möglichst überlappend zu gestaltend, so daß der Freiraum für verschiedene Aufgaben genutzt werden kann (bspw. auch als Gang).

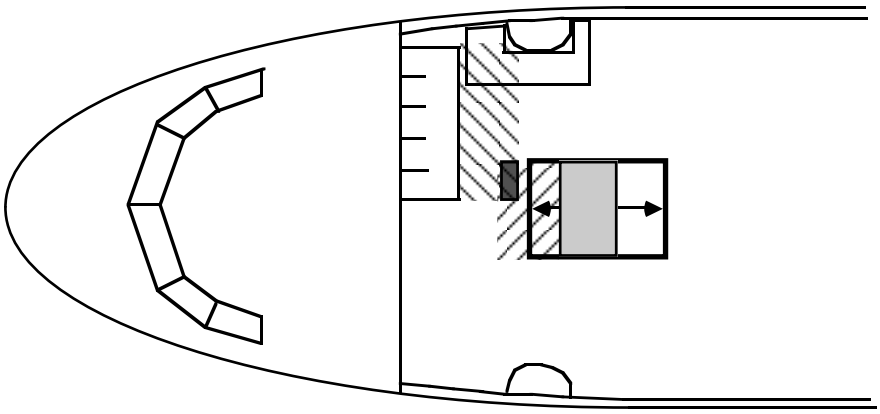


Abbildung 156. Skizze zur Visualisierung einer speziellen räumlichen Konstellation mit Überlappung der Arbeitsbereiche von zwei im Flugzeug installierten Küchen.

Eine spezielle räumliche Konstellation ist also für die Entscheidungsfindung bei der Durchführung von Aktionen zu berücksichtigen und sollte daher in der Visualisierung im Interaktionszyklus auch entsprechend dargestellt werden. Zur Erkennung von räumlichen Konstellationen kann z.B. die von Schick et al. entwickelte „Spacebox“ verwendet werden [273]. Wie schon angesprochen, ließen sich für die Organisation der Darstellung zur Laufzeit die in Kapitel 2 diskutierten Dienste eines IMMPS vorsehen. Für spezielle räumliche Konstellation könnten auch spezielle, domänenspezifische Aktionenkonzepte definiert werden wie z.B. die Beseitigung einer ungewünschten Überlappung statt einer einfachen Verschiebung. Im Gegensatz zu der zur Zeit realisierten Funktionalität der Klassifikation von Aktionsmöglichkeiten zur Entwicklungszeit ist also in diesem Fall auch eine Klassifikation von Aktionen zur Laufzeit zu unterstützen. Weitere Untersuchungen müssen das Zusammenspiel von Inferenzmöglichkeiten zur Entwicklungszeit und intelligente Präsentationsgestaltung zur Laufzeit klären.

Interessant wäre auch die Weiterentwicklung von HAMVIS zu einem Entwicklungssystem für die Spezifikation von dynamisch generierten graphischen WWW-Dokumentationen mit interaktiven Eingabe- und Interaktionsmöglichkeiten. Die Grundlagen für eine Beschreibungssprache für Markierungen (im Sinne eines graphischen Pendants zu HTML) wurden durch die Dialog- und Präsentationsstrukturierungskonzepte von HAMVIS gelegt.

Literaturverzeichnis

[1] Abowd, G.D., *Formal Aspects of Human-Computer Interaction*, Dissertation, Oxford University, Computer Laboratory, Programming Research Group, 1991.

[2] Allgayer, J., Harbusch, K., Kobsa, A., Reddig, C., Reithinger, N., Schmauks, D., *XTRA: A Natural-Language Access System to Expert Systems*, Int. Journal of Man-Machine Studies, No. 31, 1989, pp. 161-195.

[3] André, E., Rist, T., *The Design of Illustrated Documents as a Planning Task*, DFKI Research Report RR 92-45, September 1992.

[4] André, E., Rist, T., *Multimedia Presentations: The Support of Passive and Active Viewing*, DFKI Research Report RR 94-01, January 1994.

[5] André, E., Rist, T. , *Generating Coherent Presentations Employing Textual and Visual Material*, in: AI Review 9, pp. 147-165, 1995.

[6] André, E., *Ein planbasierter Ansatz zur Generierung multimedialer Präsentationen*, Dissertation, Universität des Saarlandes, 1995.

[7] Appelt, D.E., *Planning English Sentences*, Cambridge University Press, 1985.

[8] Arend, U., *Wissenserwerb und Problemlösen bei der Mensch-Computer Interaktion*, in German, S. Roderer Verlag, Regensburg, 1990.

[9] Arens, Y., Miller, L., Shapiro, S.C., Sondheimer, N.K., *Automatic Construction of User-Interface Displays*, in: Proceedings AAAI'88, Seventh National Conference on Artificial Intelligence, August 1988, pp. 808-813.

[10] Arens, Y., Miller, L., Sondheimer, N.K., *Presentation Design Using an Integrated Knowledge Base*, in: [299], pp. 207-240.

[11] Arens, Y., Hovy, E.H., Vossers, M., *On the Knowledge Underlying Multimedia Presentations*, in: [204] pp. 280-306.

[12] Arens, Y., Hovy, E.H., van Mulken, S., *Structure and Rules in Automated Multimedia Presentation Planning*, in: Proceedings IJCAI'93, August 1993.

[13] Artale, A., Franconi, E., *A Computational Account for a Description Logic of Time and Action*, in: Proceedings KR'94 Principles of Knowledge Representation and Reasoning, Doyle, J., Sandewall, E., Torasso, P. (Hrsg.) Morgan-Kaufmann Publishers, Inc., 1994.

[14] Artale, A., Franconi, E., Guarino, N., Pazzi, L., *Part-Whole Relations in Object-Centered Systems: An Overview*, erscheint in Data and Knowledge Engineering, North-Holland, Elsevier.

[15] Baader, F., Hanschke, P., *A Scheme for Integrating Concrete Domains into Concept Languages*, DFKI Research Report RR-91-10, April 1991.

[16] Bandyopadhyay, S., *Towards an Understanding of Coherence in Multi-Modal Discourse*, Technical Memo TM-90-01, Deutsches Forschungszentrum für Künstliche Intelligenz GmbH, Saarbrücken, 1990.

[17] Barnard, P., May, J., *Interactions with Advanced Graphical Interfaces and the Deployment of Latent Human Knowledge*, in: [245], pp. 15-49.

[18] Bateman, J.A., Kasper, R.T., Moore, J.D., Whitney, R.A., *A General Organization of Knowledge for Natural Language Processing: the Penman Upper Model*, March, 1990.

[19] Bernardi, A., Klauck, C., Legleitner, R., *STEP – Überblick über eine zukünftige Schnittstelle zum Produktdatenaustausch,* DFKI-Document Nr. D-90-04, Deutsches Forschungszentrum für Künstliche Intelligenz, September 1990.

[20] Bernardi, A., Klauck, C., Legleitner, R., *Ein Formalismus zur Repräsentation von Geometrie- und Technologieinformationen als Teil eines Wissensbasierten Produktmodells,* DFKI-Document Nr. D-90-05, Deutsches Forschungszentrum für Künstliche Intelligenz, Dezember 1990.

[21] Bertin, J., *Graphics and Information Processing*, de Gruyter, 1981.

[22] Bobrow, D.G., Stefik, M., *The LOOPS Manual*, Xerox Corporation, 1983.

[23] Bodart, F., Hennebert, A.-M., Leheureux, J.-M., Vanderdonckt, J., *A Model-Based Approach to Presentation: A Continuum form Task Analysis to Prototype*, in: [245], pp. 77-94.

[24] Bonar, J., Liffick, B., *Communicating with High-Level Plans*, in: [299], pp. 129-156.

[25] Borgida, A., Brachman, R.J., McGuiness, D.L., Resnick, L.A., *CLASSIC: A Structural Data Model for Objects*, in: Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data, Portland, Oregon, May-June, 1989.

[26] Borgida, A., *Towards the Systematic Development of Description Logic Reasoners: CLASP Reconstructed*. in: Proceedings KR'92 Principles of Knowledge Representation and Reasoning, Nebel, B., Rich, C., Swartout, W. (Eds.) Morgan Kaufmann Publishers, Inc., 1992, pp. 259-269.

[27] Borgida, A., Patel-Schneider, P.F., *A Semantics and Complete Algorithm for Subsumption in the CLASSIC Description Logic*, Journal of Artificial Intelligence Research, No. 1, Morgan Kaufmann Publishers, 1994, pp. 277-308.

[28] Borning, A., Duisberg, R., *Constraint-Based Tools for Building User Interfaces*, in: ACM Transactions on Graphics, 5(4), Oct. 86, pp. 345-374.

[29] Bos, E., Huls, C., Classen, W., *EDWARD: Full Integration of Language and Action in a Multimodal User Interface*, Int. J. Human-Computer Studies, 1994, 40, pp. 473-495.

[30] Bowman, W.J., *Graphic Communication*, John Wiley & Sons, 1968.

[31] Brachman, R.J., McGuiness, D.L., Patel-Schneider, P.F., Resnick, L.A., Borgida, A., *Living with CLASSIC: When and How to Use a KL-ONE-Like Language*, in: [290], pp. 401-456.

[32] Brachman, R.J., *"Reducing" CLASSIC to Practice: Knowledge Representation Theory Meets Reality*, in: Proceedings KR'92 Principles of Knowledge Representation and Reasoning, Nebel, B., Rich, C., Swartout, W. (Eds.) Morgan Kaufmann Publishers, Inc., 1992, pp. 247-258.

[33] Brandenburg, F.J., *Nice Drawings of Graphics are Computationally Hard*, Proceedings Visualization in Human-Computer Interaction, Seventh Interdisciplinary Workshop on Informatics and Psychology, Schärding, Austria, May 1988, Gorny, P., Tauber, M.J. (Eds.), Springer-Verlag, 1990, pp. 1-15.

[34] Breuker, J., de Greef, P., *Modelling System-User Cooperation* in KADS, in: [276], pp. 47-70.

[35] Brüning, B., *Modellierung von Interaktionshandlungen zur Generierung von Benutzungsschnittstellen*, Diplomarbeit, Universität Hamburg, Fachbereich Informatik, 1995.

[36] Butz, U., *Grafische Wissensrepräsentation: das BMFT-Verbundvorhaben GRAWIS*, in German, Heft 3, 1992.

[37] Carberry, S., Kazi, Z., Lambert, L., *Modeling Discourse, Problem-Solving, and Domain-Goals Incrementally in Task-Oriented Dialogue*, in: Proceedings UM'92, 3rd International Workshop on User MOdeling, André, E., Cohen, R., Graf, W., Kass, B., Paris, C., Wahlster,W. (Eds.), Dokument D-92 17 Deutsches Forschungszentrum für Künstliche Intelligenz, August 1992, pp. 192-201.

[38] Card, S., Moran, T., Newell, A., *The Psychology Of Human-Computer Interaction*, Erlbaum, Hillsdale, 1983.

[39] Casner, S.M., *A Task-Analytic Approach to the Automated Design of Graphic Presentations*, ACM Transactions on Graphics, Vol. 10, No. 2., April 1991, pp. 111-151.

[40] Casner, S.M., *Automatic Design of Efficient Visual Problem Representations*, AAAI Spring Symposium on Reasoning with Diagrammatic Representations, Stanford University, March 1992.

[41] Cawsey, A., *Explanation and Interaction – The Computer Generation of Explanatory Dialogues*, The MIT Press, 1992.

[42] Cawsey, A., *User Modelling in Interactive Explanations*, in: User Modeling and User-Adapted Interaction, 3, 1993, pp. 221-247.

[43] Chandrasekaran, B., Johnson, T.R., Smith, J.W., *Task-Structure Analysis for Knowledge Modeling*, in: Knowledge-Oriented Software Design, Cuena, J. (Ed.), Elsevier Science Publishers B. V. (North Holland), 1993.

[44] Chang, S. (Hrsg.), Ichikawa, T. (Hrsg.), Ligomenides, P. (Hrsg.), *Visual Languages*, Invited and Selected Papers of Two IEEE Workshops: Languages and Automation - Cognitive Aspects in Information Processing (1985) and Visual Languages, New York: Plenum Press, 1986, 460 S. (Management and Information Systems).

[45] Chang, S.K., *Visual Languages: A Tutorial and Survey*, IEEE Software, January 1987, pp. 29-39.

[46] Chang, S.K., *Principles of Visual Programming Systems*, Prentice Hall, 1990.

[47] Chang, S.K. (Ed.), *Visual Languages and Visual Programming*, Plenum Press, 1990.

[48] Chesley, H.R., *Walt Disney and Computer User Interface Design*, December 1988.

[49] Ciccarelli, E.C., *Presentation Based User Interfaces*, MIT Ariticial Intelligence Laboratory, Technical Report 794, August 1984.

[50] Cleveland, W.S., McGill R., *Graphical Perception: Theory, Experimentation and Application to the Development of Graphical Methods*, Journal American Statistics Association, 79, 387, September 1984, pp. 531-554.

[51] Cleveland, W.S., McGill, R., *An Experiment in Graphical Perception*, Int. Journal Man-Machine Studies, 25, 1986, pp. 491-500.

[52] *Common Lisp Interface Manager, User Guide*, Franz Inc., 1994.

[53] Cockton, G., Draper, S.W., Weir, G.R.S., *People and Computer IX*, Proceedings of HCI'94, Glasgow, August 1994.

[54] Cohen, P.R., Levesque, H.J., *Speech Acts and Rationality*, Proceedings of the 23rd ACL Conference, Chicago 1985, pp. 49-59.

[55] Cohen, P.R., Perrault, C.R., *Elements of a Plan-Based Theory of Speech Acts*, in: Readings in Artificial Intelligence, Webber, B.L., Nilsson, N.J. (Eds.), Morgan Kaufmann, 1981, pp. 477-496.

[56] Conati, C., Slack, J., *Accessing Information Through Graphics*, Proceedings ECAI'92, 10th European Conference on Artificial Intelligence, Vienna, Austria, Neumann, B. (Ed.), 1992.

[57] Copas, C.V., Edmonds, E.A., *Executable Task Analysis: Integration Issues*, in: [53], pp. 339-352.

[58] Cramer, N., *MIRAGE: A CLIM-based Editor for Building Gadget-Oriented Graphical User Interfaces*, in: Proceedings Lisp Users and Vendors Conference, 1995.

[59] Cuena, J. (Ed.), Knowledge-Oriented Software Design, IFIP, North-Holland, 1993.

[60] Cunis, R., *Modellierung technischer Systeme in der Begriffshierarchie*, in: Das PLAKON-Buch, Cunis, R., Günter, A., Strecker, H. (Hrsg.), Springer-Verlag, Reihe Informatik-Fachberichte, Band 266, 1991.

[61] Cypher, A., Stelzner, M, *Graphical Knowledge-Based Model Editors*, in: [299], pp. 403-420.

[62] Cypher, A (Ed.), *Watch What I do: Programming by Demonstration*, Mit, Cambridge, Ma, 1993.

[63] David, J.-M., Krivine, J.-P., Simmons, R. (Eds.), *Second Generation Expert Systems*, Springer-Verlag, 1993.

[64] Davidson, R., Harel, D., *Drawing Graphs Nicely Using Simulated Annealing*, Department of Applied Mathematics & Computer Science, The Weizman Institute of Science, Rehovot 76100, Israel, CS89-13, July 1989.

[65] Dearden, A.M., Harrison, M.D., *Modelling Interaction Properties for Interactive Case Memories*, in: [245], pp. 301-316.

[66] Deisel, W., *Anforderungen an neuartige Graphiksysteme*, in German, TASSO-Report Nr. 29, Gesellschaft für Mathematik und Datenverarbeitung mbH, 1990.

[67] Deisel, W., *Objektorientierte Diagrammgestaltung mit EPICT und COMPOUND*, in German, TASSO-Report Nr. 43, Gesellschaft für Mathematik und Datenverarbeitung mbH, December 1992.

[68] Devanbu, P.T., Litman, D.J., *Plan-Based Terminological Reasoning*, in: Proceedings KR'91 Principles of Knowledge Representation and Reasoning, Allen, J., Fikes, R., Sandewall, E. (Hrsg.), Morgan Kaufmann Publishers, Inc., 1991, pp. 128-138.

[69] Di Battista, G., Pietrosanti, E., Tamassia, R., Tollis, I.G., *Automatic Layout of PERT Diagrams with X-PERT*, 1989 IEEE Workshop on Visual Languages, IEEE Computer Society Press, 1989, pp. 171-176.

[70] Di Eugenio, B., *Action Representation for Intepreting Purpose Clauses in Natural Language Instruction*, in: Proceedings KR'94 Principles of Knowledge Representation and Reasoning, Doyle, J., Sandewall, E., Torasso, P. (Hrsg.) Morgan-Kaufmann Publishers, Inc., 1994, pp. 158-169.

[71] Di Primio, F. (Hrsg.), *Methoden der Künstlichen Intelligenz für Grafikanwendungen*, Addison-Wesley, 1995.

[72] Ding, Ch., Mateti, P., *A Framework for the Automated Drawing of Data Structure Diagrams*, IEEE Transactions on Software Engineering, Vol. 16, No. 5, May 1990.

[73] Dix, A., Finlay, J., Abowd, D., Beele, R., *Human-Computer Interaction*, Prentice Hall, 1993.

[74] Dix, A., *LADA – A Logic for the Analysis of Distributed Actions*, in: [245], pp. 317-332.

[75] Djeraba, C., *Composite Objects and Dependency Relationssships in a Knowledge-Based System*, in: Applications of Artificial Intelligence in Engineering VIII, Vol. 1: Design, Methods and Techniques, Rzevski, G., Pastor, J., Adey, R.A., (Eds.).

[76] Donini, F.M., Lenzerini, M., Nardi, D., Schaerf, A., Nutt, W., *Adding Epistemic Operators to Concept Languages*, in: Proc. KI'92 Principles of Knowledge Representation and Reasoning, Proceedings of the Third International Conference, Morgam Kaufmann, 1992, pp. 342-353.

[77] Duke, D.J., Harrison, M.D., *Folding Human Factors Into Rigorous Development*, in: [245], pp. 333-347.

[78] Eades, P., Xuemin, L., *How to Draw a Directed Graph*, Proceedings, 1989 IEEE Workshop on Visual Languages, IEEE Computer Society Press, 1989, pp. 13-17.

[79] Ehn, P., *Work-Oriented Design of Computer Artifacts*, Gummessons, Falköping, Schweden, 1988.

[80] Eschenbach, C., Heydrich, W. (Hrgs.), *Parts and Wholes – Integrity and Granularity*, Graduiertenkolleg Kognitionswissenschaft, Bericht Nr. 29, August 1995.

[81] Feiner, S., *APEX: An Experiment in the Automated Creation of Pictorial Explanations*, IEEE Computer Graphics & Applications, November 1985.

[82] Feiner, S., McKeown, K.R., *An Architecture for Knowledge-Based Graphical Interfaces*, in: [299], pp. 259-279.

[83] Feiner, S., *Coordinating Text and Graphics in Explanation Generation*, in: Proceedings AAAI-90, Eigth National Conference on Artificial Intelligence, AAAI Press/The MIT Press, 1990, pp. 443-449.

[84] Feiner, S., McKeown, K.R., *Automating the Generation of Coordinated Multimedia Explanations*, Computer, October 1991, pp. 33-41.

[85] Fischer, G., McCall, R., Morch, A., *Design Environment for Constructive and Argumentative Design*, in: Proceedings CHI'89 Human Factors in Computing Systems, ACM Press, 1989, pp. 269-275.

[86] Fischer, G., Nieper-Lemke, H., *HELGON: Extending the Retrieval by Reformulation Paradigm*, in: Proceedings CHI'89 Human Factors in Computing Systems, ACM Press, 1989, pp. 357-362.

[87] Fischer, G., Lemke, A., McCall, R., *Towards a System Architecture Supporting Contextualized Learning*, in: Proceedings AAAI'90, 8th National Conference on Artificial Intelligence, AAAI Press/ The MIT Press, 1990, pp. 420-425.

[88] Fischer, G., Nakakoji, K., *Making Design Objects Relevant to the Task at Hand*, Proceedings AAAI'91, 9th National Conference on Artificial Intelligence, AAAI Press/The MIT Press, 1991, pp. 67-73.

[89] Fischer, G., Nakakoji, K., *Empowering Designers with Integrated Design Environments*, in: Proceedings of the First International Conference on Artificial Intelligence in Design, Royal Museum of Scotland, Edinburg, 1992.

[90] Fischer, G., McCall, R., Ostwald, J., Reeves, J., Shipman, F., *Seeding, Evolutionary Growth and Reseedings: Supporting the Incremental Development of Design Environments*, in: Proceedings CHI'94 Human Factors in Computing Systems, ACM Press, 1994, pp. 292-298.

[91] Foley, J.D., Wallace, V.L., Chan, P., *The Human Factors of Computer Graphics Interaction Techniques*, IEEE Computer Graphics, Vol. 4, No. 11, November 1984, pp. 11-44.

[92] Foley, J.D., Kim, W.C., Kovacevic S., Murray, K., *UIDE – An Intelligent User Interface Design Environment*, in: [299].

[93] Foley, J.D., Sukaviriya, P.N., *History, Results, and Bibliography of the User Interface Design Environment (UIDE), an Early Model-Based System for User Interface Design and Implementation*, in: [245], pp. 3-14.

[94] Friedell, M., *Automatic Graphics Environment Synthesis*, Technical Report CCA-83-03, Computer Corporation of America, 1983.

[95] Friedell, M., *Automatic Synthesis of Graphical Object Descriptions, Computer Graphics*, Volume 18, Number 3, July 1984.

[96] Geller, J., Shapiro, S.C., *Graphical Deep Knowledge for Intelligent Machine Drafting*, in: Proceedings IJCAI'87, International Joint Conferences on Artificial Intelligence, McDermott, J., Morgan Kaufmann Publishers, Inc., 1987, pp. 547-551.

[97] Gerstl P., Pribbenow, S., Midwinters, End Games, and Bodyparts: A Classification of Part-Whole Relations, in: [80], auch in: *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Guarino, N., Poli, R. (Hrsg.), Kluwer, Dordrecht, 1994.

[98] Gnanamgari, S., *Information Presentation Through Default Displays*, Ph.D. dissertation, Univ. of Pennsylvania, May 1981.

[99] Goetze, R., *Dialogmodellierung für multimediale Benutzerschnittstellen*, in German, Dissertation, Univerity of Oldenburg, Mar. 1994.

[100] Goossens, M., Mittelback, F., Samarin, A., *The LaTeX Companion*, Addison-Wesley, 1994.

[101] Gould, J.D., Lewis, C., *Designin for Usability: Key Principles and What Designers Think*, in: Readings in Human-Computer-Interaction: A Multidisciplinary Approach, Baecker, R.M., Buxton, W.A. (Eds.), Morgan Kaufmann Publishers, Inc., 1985, pp. 528-539.

[102] Graf, W., *Constraint-Based Graphical Layout of Multimodal Presentations*, Research Report, DFKI, Deutsche Forschungszentrum für Künstliche Intelligenz, RR-92-15,Jan. 1992.

[103] Grice, H.P., *Logic and Conversation, in: Syntax and Semantics*: 3. Speech Acts, Walker, D. (Ed.), Academic Press, 1975.

[104] Grosz, B.J., Sidner, C.L., *Attention, Intentions, and the Structure of Discourse*, J. Computational Linguistics, 13 (3), 1986, pp. 175-204.

[105] Gruber, Th., *A Translation Approach to Portable Ontology Specification*, Knowledge Systems Laboratory, Technical Report KSL 92-71, April 1993.

[106] Günter, A. (Hrsg.), *Wissenbasiertes Konfigurieren, Ergebnisse aus dem Projekt PROKON*, Infix-Verlag, 1995.

[107] Gugerty, L., *The Use of Analytical Models in Human-Computer-Interface Design*, Int.Journal Man-Machine Studies, 38, 1993, pp. 625-660.

[108] Haarslev, V., Möller, R., *VIPEX: Visual Programming of Experimental Systems, in: Visual Languages and Visual Programming*, Chang, S.-K. (Hrsg.), Plenum Publishing Corporation, 1990.

[109] Haarslev, V., Möller, R., *A Declarative Formalism for Specifying Graphical Layout*, in: Proc. 1990 IEEE Workshop on Visual Languages, Skokie/IL, Oct. 4-6, 1990, IEEE Computer Society Press, 1990.

[110] Haarslev, V., Möller, R., *A Framework for Visualizing Object-Oriented Systems*, in: Proceedings OOPSLA'90, SIGPLAN Notices, Volume 25, No. 10, October 1990.

[111] Haarslev, V., Möller, R., *Visualization and Graphical Layout in Object-Oriented Systems*, Journal of Visual Languages and Computing, Nr. 3, 1992, pp. 1-23.

[112] Haarslev, V., Möller, R., Schröder, C., *Combining Spatial and Terminological Reasoning*, in: KI-94: Advances in Artificial Intelligence - Proc. 18th German Annual Conference on Artificial Intelligence, Saarbrücken, Spet. 18-23, 1994, Volume 861 of Lecture Notes in Artificial Intelligence, pp. 142-153, Springer-Verlag, 1994.

[113] Haber, R.N., Wilkinson, L., *Perceptual Components of Computer Displays*, in: IEEE Computer Graphics and Applications, May 1992, pp. 23-35.

[114] Hanschke, P., *Specifying Role Interaction in Concept Languages*, in: Proc. KI'92 Principles of Knowledge Representation and Reasoning, Proceedings of the Third International Conference, Morgan Kaufmann, 1992.

[115] Hanschke, P., Hinkelmann, K., *Combining Terminological and Rule-Based Reasoning for Abstraction Processes*, Research Report RR-92-40, DFKI, November 1992.

[116] Hanschke, P., *A Declarative Integration of Terminological, Constraint-Based, Data-Driven and Goal-Directed Reasoning*, DFKI Research Report RR-93-46, Oktober 1993.

[117] Harel, D., *On Visual Formalisms*, Communications of the ACM, Vol. 31, May 1988, pp. 514-530.

[118] Hartson, H.R., Siochi, A.C., Hix, D., The UAN: *A User-Oriented Representation for Direct-Manipulation Interface Design, ACM Transactions on Information Systems*, Vol. 8, Nr. 3, July 1990, pp. 181-203.

[119] Hartson, H.R., Brandenburg, J.L., Hix, D., *Different Languages for Different Development Activities: Behavioral Representation Techniques for User Interface Design*, in: [227], pp. 303-325.

[120] Hartson, H.R., Mayo, K.A., *A Framework for Precise, Reusable Task Abstraction*, in: [245], pp. 279-297.

[121] Hayball, C., *Dialogue Specification for Knowledge-Based Systems*, in: *User Interface Management and Design*, Duce, D.A., Gomes, M.R., Hopgood, F.R.A., Lee, J.R. (Eds.), Eurographics Seminars, Springer Verlag, 1991, pp. 169-178.

[122] Heinsohn, J., Kudenko, D., Nebel, B., Profitlich, H.-J., *RAT – Representation of Actions in Terminological Logics*, in: DFKI Workshop ion Taxonomic Reasoning, DFKI-Report Nr. D-92-08, pp. 16-22.

[123] Heintze, N.C., , Jaffar, J., Michaylov, S., Stuckey, P.J., Yap, H.C., *The CLP(R) Programmer's Manual*, IBM T.J., Watson Research Center, Yorktown Height, NY 10598.

[124] Helander, M. (Ed.), *Handbook of Human-Computer Interaction*, Elsevier Science Publishers, B.V. (North Holland), 1988.

[125] Helpenstein, H.J. (Hrsg.), *CAD Geometry Data Exchange Using STEP – Realisation of Interface Processors*, Research Reports ESPRIT, Springer Verlag, 1991.

[126] Hesse, J., *Flexible Graphformatierung mit hybriden Methoden*, in German, TASSO-Report Nr. 30, Gesellschaft für Mathematik und Datenverarbeitung, Oct. 1991.

[127] Hightower, D.W., *A Solution to Line-Routing Problems on the Continuous Plane*, Proceedings Sixth Design Automation Workshop, 1969, pp. 1-24.

[128] Hollan, J., Rich, E., Hill, W., Wroblewski, D., Wilner, W., Wittenburg, K., Grudin, J., *An Introduction to HITS: Human Interface Tool Suite*, in: [299], pp. 293-337.

[129] Hollunder, B., Baader, F., *Qualifying Number Restrictions in Concept Languages*, in: KR'91 Principles of Knowledge Representation and Reasoning, Allen, J., Fikes, R., Sandewall, E. (Hrsg.), Morgan-Kaufmann Publishers, Inc., 1991, pp. 335-346.

[130] Holz, D., *Über das Entwerfen von Gebrauchssoftware*, Dissertation, Universität Basel, 1995.

[131] Hovy, E.H., *A Brief Overview of the Penman Project*, Information Sciences Institute of the University of Southern California, January 1990.

[132] Hovy, E.H., *Approaches to the Planning of Coherent Text*, in: Natural Language Generation in Artificial Intelligence and Computational Linguistics, Paris, C.L., Swartout, W.R., Mann, W.C. (Eds.), Kluwer Academic Publishers, 1991, pp. 83-102.

[133] Hovy, E.H., Arens, Y., *Automatic Generation of Formatted Text*, in: Proceedings of AAAI-91, Ninth National Conference on Artificial Intelligence, Anaheim, CA, AAAI Press/MIT Press, 1991, pp. 92-97.

[134] Hovy, E.H., *Automated Discourse Generation Using Discourse Structure Relations*, Artificial Intelligence, 63, 1993, pp. 341-385.

[135] Hudson, S.E., Mohamed, S.P., *Interactive Specification of Flexible User Interface Displays*, ACM Transactions on Information Systems, Vol. 8, No. 3, July 1990, pp. 269-288.

[136] Intellicorp, *KEE ActiveImages3 Reference Manual, KEE Software Development System*, Intellicorp 1986.

[137] Jaffar, J., Michaylov, S., Stuckey, P., Yap, R.H., *The CLP(R) Language and System*, in: ACM Transactions on Programming Languages and Systems, Vol. 14, No. 3, July 1992, pp. 339-395.

[138] Johannsen, G., *Mensch-Maschine-Systeme*, in German, Springer-Verlag, 1993.

[139] Johnson, P., *Towards a Task Model of Messaging: An Example of the Application of TAKD to User Interface Design*, in: Johnson, P., Cook, S., People and Computers: Designing the Interface, Proceedings of the Conference of the British Computer Society Human Computer Interaction Specialist Group, Cambridge University Press, 1985, pp. 46-62.

[140] Johnson, P, Wilson, S. Markopoulos, P., Pycock, J., *ADEPT - Advanced Design Environment for Prototyping with Task Models*, in: Proceedings, INTERCHI'93.

[141] Johnson, P., Wilson, S., Johnson, H., *Scenarios, Task Analysis and The Adept Design Environment*, in: Scenario-Based Design, Carroll, J. (Ed.), Addison-Wesley, 1995.

[142] Kaczmarek, T., Mark, W., Sondheimer, N., *The Consul/CUE Interface: An Integrated Interactive Environment*, Proceedings CHI'83, Human Factors in Computing Systems, Special Issue of the SIGHCHI Bulletin, December, 1983, pp. 98-102.

[143] Kamada, T., Kawai, S., *A General Framework for Visualizing Abstract Objects and Relations*, ACM Transactions on Graphics, Vol. 10, No. 1, January 1991, pp. 1-39.

[144] Kaplunova, A., *Hervorhebung als Mittel der visuellen Kommunikation*, Universtität Hamburg, Fachbereich Informatik, Studienarbeit, 1996.

[145] Karp, P.D., Lowrance, J.D., Strat, T.M., Wilkins, D.E., *The Grasper-CL Graph Management System*, in: Lisp and Symbolic Computation, Vol 7, No. 4, Kluwer Academic Publishers 1994, pp. 251-290.

[146] Keene, S.E., *Object-Oriented Programming in CLOS – A Programmer's Guide to CLOS*, Addison-Wesley, 1989.

[147] Kieras, D.E., Polson, P.G., *An approach to the formal analysis of user complexity*, Int. Journal of Man-Machine Studies, 22, pp. 365-394.

[148] Kim, W., Foley, F., *DON: User Interface Presentation Design Assistant*, in: Proceedings UIST'90, October 1990, pp. 10-20.

[149] Kitajima, M., Polson, P.G., *A Computational Model of Skilled Use of a Graphical User Interface*, Proceeding Computer-Human Interaction CHI'92, ACM 1992, pp. 241-249.

[150] Klement, K., *Präsentation mit STEP – Schnittstelle zwischen Computer-Graphik und CAD/CAM*, Springer-Verlag, 1992.

[151] Knospe, G, *Repräsentation von begrifflichem Wissen auf der Grundlage von kognitionspsychologischen Befunden*, Fachbereich Informatik, Universität Hamburg, FBI-HH-M-185/90, August 1990.

[152] Kobsa, A., *A Taxonomy of Beliefs and Goals for User Models in Dialog Systems*, SFB 314, Universität des Saarlandes, Bericht Nr. 28, December 1987.

[153] Kobsa, A., Wahlster, W (Hrsg.), *User Models in Dialog Systems*, Springer-Verlag, 1989.

[154] Kobsa, A., *Modeling the User's Conceptual Knowledge in BGP-MS, A User Modeling System*, SFB 314 (XTRA), Universität des Saarlandes, Bericht Nr. 71, September 1990 (revised version), also in: Computational Intelligence 6(4), 1990, pp. 193-208.

[155] Kochhar, S., Marks, J., Friedell, M., *Interaction Paradigms for Human-Computer Cooperation in Graphical Object Modeling*, in: Proceedings of Graphics Interface '91, Morgan Kaufmann Publishers, pp. 180-191.

[156] Köhler, J., *An Application of Terminological Logics to Case-Based Reasoning*, in: Proceedings KR'94 Principles of Knowledge Representation and Reasoning, Doyle, J., Sandewall, E., Torasso, P. (Hrsg.) Morgan-Kaufmann Publishers, Inc., 1994, pp. 351-362.

[157] Köhler, J., *Wiederverwendung von Plänen in deduktiven Planungssystemen*, Infix-Verlag, Reihe DISKI, Nr. 65, 1994.

[158] Kolb, R., *Die graphische Umsetzung statistischer Aussagen*, in German, TASSO-Report Nr. 39, Gesellschaft für Mathematik und Datenverarbeitung mbH, April 1992.

[159] Kolb, R., *Applying Non-Monotonic Reasoning to Graphics Design*, TASSO-Report Nr. 52, Gesellschaft für Mathematik und Datenverarbeitung mbH, August 1993.

[160] Kölln, M., *Diskursgenerierung - alte und neue Ansätze*, in German, University of Hamburg, Computer Science Department, 1991, FBI-M-192/91.

[161] Kölln, M., *Generating Natural Language from Discourse Goals*, in German, University of Hamburg, Computer Science Department, 1991, FBI-M-196/91.

[162] Kopisch, M., *Expertensystemgestützte Konfigurierung der Passagierkabine eines Verkehrsflugzeuges*, Diploma Thesis, in German, University of Hamburg, 1991.

[163] Kopisch, M., Günter, A., *Configuration of a Passenger Aircraft Cabin Based on a Conceptual Hierarchy, Constraints and Flexible Control*, in: Proceedings 5th International Conference IEA/AIE-92 Industrial and Engineering Application of Artificial Intelligence and Expert Systems, Belli, F., Radermacher, F.J. (Eds.), Springer Verlag, 1992, pp. 421-430.

[164] Kosslyn, S., *Elements Of Graph Design*, Freeman, New York, 1994.

[165] Krasner, G.E., Pope, S.T., *A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80*, ParcPlace Systems, Smalltalk Manual, Palo Alto.

[166] Kumar, B., Roberts, J.C., Hoffman, R., Ikeuchi, K., Kanade, T., *VANTAGE: A Frame-Based Geometric Modeling System, Programmer/User's Manual Version 2.0*, Technical Report, Carnegie Mellon University, The Robotics Institute, CMU-RI-TR-91-31, 1992.

[167] Kurlander, D., *Graphical Editing by Example*, Ph.D. Dissertation, Dept. of Computer Science, Columbia University, 1993.

[168] Laird, J., Newell, A., Rosenbloom, P.S., *SOAR: An Architecture for General Intelligence*, Artificial Intelligence, 33, pp. 1-64.

[169] Landay, J.A., Myers, B.A., *Just Draw It! Programming by Sketching Storyboards*, Human-Computer Interaction Institute Technical Report, CMU-HCII-95-106,.

[170] Langen, M.H., *Evolutionäres Prototyping multimodaler, farbgraphischer Benutzerschnittstellen in der Medizin Entwurf, Implementierung und Ansätze zur Bewertung*, RWTH Aachen, Dissertation, 1991.

[171] Larkin, J.H., Simon, H.A., *Why a Diagram is (Sometimes) Worth Ten Thousand Words*, Cognitive Science, 11, 1987, pp. 65-99.

[172] Lemke, A.C., Fischer, G., *A Cooperative Problem Solving Szstem for User Interface Design*, Proc. AAAI'90, 8th National Conference on Artificial Intelligence, AAAI-Press/The MIT Press, 1990, pp. 479-484.

[173] Levelt, W.J., *Accessing Word in Speech Production: Stages, Processes and Representation*, Cognition, 42, pp. 1-22.

[174] Leyton, M., *Symmetry, Causality and Mind*, A Bradford Book, The MIT Press, 1992.

[175] Lieberman, H., *There's More th Menu System than Meets the Screen*, in: Proceedings ACM SigGraph, Vol. 19, No. 3, 1985, pp. 181-189.

[176] Lim, K.Y., Long, J., *Structured Notations to Support Human Factors Specification of Interactive Systems*, in: [53], pp. 313-326.

[177] Livingstone, M., *Kunst, Schein und Wagrnehmung*, Spektrum der Wissenschaft, März 1988.

[178] Loewgren, J., *The Ignatius Environment – Supporting the Design and Development of Expert-System User Interfaces*, IEEE Expert, Intelligent Systems and their Applications, August 1992, pp. 49-57.

[179] Lohse, J., *A Cognitive Model for the Perception and Understanding of Graphs*, Proceedings CHI'91, Conference on Computer-Human Interaction, ACM Press, 1991, pp. 137-144.

[180] Lohse, J., *A Cognitive Model for Understanding Graphical Perception*, Human-Computer Interaction, Vol. 8, Lawrence Erlbaum Associates, Inc., 1993, pp. 353-388.

[181] Lohse, G.L., Biolski, K., Walker, N., Rueter, H.H., *A Classfication of Visual Representations*, in: Communication of the ACM, Dezember 1994, Vol. 37, No. 12, pp. 36-49.

[182] Lowe, D.G., *Perceptual Organization and Visual Recognition*, Kluwer Academic Publishers, Boston, 1985.

[183] Luck, K. von, Owsnicki-Klewe, B., *KL-ONE: Eine Einführung*, in: Struß, P. (Hrsg.), Wissensrepräsentation, Olderbourg Verlag, München-Wien, 1991, pp. 103-121.

[184] Luo, P., *A Human-Computer Collaboration Paradigm for Bridging Design Conceptualization and Implementation*, in: [245], pp. 129-147.

[185] Maaß, S., Oberquelle, H., *Perspectives and Metaphors for Human-Computer Interaction*, in Floyd, Ch., Züllighoven, H., Budde, R., Keil-Slawik, R. (Eds.), *Software Development and Reality Construction*, Springer-Verlag, Berlin, 1992, pp. 223-251.

[186] MacDonald, L, *Smart Use of Color in Displays*, in: BYTE, Dezember 1991, pp. 84IS/35-47.

[187] MacIntyre, B., *A Constraint-Based Approach to Dynamic Color Management for Windowing Interfaces, Master Thesis, University of Waterloo*, Ontario, Canada, 1991.

[188] Mackinlay, J., Genesereth, M.R., *Expressiveness and Language Choice*, Department of Computer Science, Stanford University, Report No. STAN-CS-85-1037, January 1985.

[189] Mackinlay, J., *Automating the Design of Graphical Presentations*, Department of Computer Science, Stanford University, Report No. STAN-CS-86-1138, December 1986.

[190] Mackinlay, J., *Automating the Design of Graphical Presentations of Relational Information*, ACM Transactions on Graphics, Vol. 5, No. 2, April 1986.

[191] Mackinlay, J.D., Card, S.K., Robertson, G.G., *The Information Visualizer – An Information Workspace*, Proceedings CHI'91, Computer-Human Interaction 1991, ACM Press, pp. 181-183.

[192] Mann, W.C., Thompson, S.A., *Rhetorical Structure Theory: Description and Construction of Text Structure*, in: Natural Language Generation, Kempen, G. (Ed.), Martinus Nijhoff Publishers, Dordrecht, 1987, pp. 85-96.

[193] Mann, W.C., *Text Generation: The Problem of Text Structure*, in: Natural Language Generation Systems, McDonald, D.D., Bolc, L. (Eds), Springer-Verlag, 1988, pp. 47-67.

[194] Marcus, A., *Graphic Design for Electronic Documents and User Interfaces*, ACM Press.

[195] Mark, W., *Representation and Inference in the Consul System*, in : Proceedings IJCAI'81, American Association for Artificial Intelligence, August 1981, pp. 375-381.

[196] Mark, W., *Knowledge-Based Interface Design*, in: User Centered System Design – New Perspectives on Human-Computer Interaction, Norman, D.A., Draper, S.W. (Eds.), Lawrence Erlbaum Ass., 1986, pp. 219-238, also published in: Human-Computer Interaction, Vol. 1, 1985, pp. 339-357.

[197] Marks, J., *A Syntax and Semantics for Network Diagrams*, 1990 IEEE Workshop on Visual Languages, IEEE Computer Society Press, 1990, pp. 104-110.

[198] Marks, J., Reiter, E., *Avoiding Unwanted Conversational Implicatures in Text and Graphics*, Proceedings AAAI'90, Eighth National Conference on Artificial Intelligence, AAAI Press/The MIT Press, 1990, pp. 450-456.

[199] Marr, D., *Vision*, W.H. Freeman & Company, 1982.

[200] Matsuoka, S., Takahashi, S., Kamada, T., Yonezawa, A., *A General Framework for Bidirectional Translation between Abstract and Pictorial Data*, in: ACM Transactions on Information Systems, Vol. 10, No. 4, Oct. 1992.

[201] Maybury, M.T., *Planning Multimedia Explanations Using Communicative Acts*, Proceedings AAAI-91, Ninth National Conference on Artificial Intelligence, AAAI Press/The MIT Press, 1991, pp. 61-66.

[202] Maybury, M.T., *Communicative Act for Presentation Generation*, Int. J. Man-Machine Studies, 37, 1992, pp. 135-172.

[203] Maybury, M.T., *Planning Multimedia Explanations Using Communicative Acts*, in: [204] pp. 60-74.

[204] Maybury, M.T. (Ed.), *Intelligent Multimedia Interfaces*, AAAI Press/The MIT Press, 1993.

[205] McCleary, G.F., *An Effective Graphic "Vocabulary"*, IEEE Computer Graphics and Applications, Vol. 3, Number 2, March/April 1983.

[206] McKeown, K., *Text Generation*, Cambridge University Press, 1985.

[207] McTear, M.F. (Ed.), *Special Issue on User Modeling*, Artificial Intelligence Review, Vol. 7, Nos. 4-3, August 1993.

[208] Mittal, V.O., Paris, C.L., Patil, R., Swartout, B., *Organizing Plan Libraries in Subsumption Hierarchies: Specifity-Based Plan Selection*, University of Southern California, Information Sciences Institute, Report Nr. ISI/RR-93-305, Dezember 1991.

[209] Miyashita, K., Matsuoka, S., Takahashi, S., Yonezawa, A., Kamada, T., *Declarative Programming of Graphical Interfaces by Visual Examples*. UIST'92, 5th Symposium of User Interface Software and Technology, Nov. 15-18, 1992, ACM 1992.

[210] Möller, J.-U., *Leitfaden zur Modellierung von Fachdialogen zwischen Nutzern und System* (in German), University of Hamburg, Computer Science Department, Report FBI-HH-M-211/92, 1992.

[211] Möller, R., *Gestaltung und Implementierung einer graphischen Dialogschnittstelle nach dem Vorbild eines datenfluß- und objektorientierten Bildfolgenanalysesystems*, Ralf Möller, Universität Hamburg, Fachbereich Informatik, Mitteilung Nr. 163/88, November 1988.

[212] Möller, R., Haarslev, V., *Layoutspezifikationen für komplexe graphische Objekte*, in German, in: Proc. Graphik und KI, GI-Fachgespräch, Kansy, K., Wißkirchen, P. (eds.), Königswinter, April 1990, pp. 78-91.

[213] Möller, R., *Reasoning about Domain Knowledge and User Actions for Interactive Systems Development*, IFIP Working Groups 8.1/13.2 Conference on Domain Knowledge for Interactive System Design, Chapman & Hall, 1996.

[214] Möller, R., *Knowledge-Based Dialog Structuring for Graphics Interaction*, in: Proc. ECAI'96, Budapest, Hungary, August 1996.

[215] Möller, R., *A Functional Layer for Description Logics: Knowledge Representations Meets Object-Oriented Programming*, in: Proc. OOPSLA'96, Object-Oriented Programming: Systems, Languages and Applications, San Jose, 1996.

[216] Mooney, D.J., Carberry, M.S., McCoy, K.F., *The Identification of a Unifying Framework for the Organization of Extended, Interactive Explanations*, Technical Report 90-1, July 19, 1989, Department of Computer and Information Sciences, University of Delaware, Newark, Delaware 19716.

[217] Moore, J.D., *A Reactive Approach to Explanation in Expert and Advice-Giving Systems*, Doctorial Dissertation, University of California, Los Angeles, 1989.

[218] Müller, B.S., Sprenger, M., *Dialogabhängige Erklärungsstrategien für modellbasierte Expertensysteme - Das Projekt DIAMOD*, NRW-Forschungsverbund, Anwendungen der Künstlichen Intelligenz, in German, DIAMOD Report Nr. 2, Juli 1991.

[219] Mulken, S. van, Reasoning about the User's Decoding of Presentations in an Intelligent Multimedia Presentation System, in: Proc. 5th International Conf. on User Modeling, Carberry, S., Zukerman, I. (Eds.), pp. 67-74.

[220] Murch, G.A., *Physiological Principles for the Effective Use of Color*, in: IEEE Computer Graphics, Vol. 4, No. 11, 1984, pp. 49-54.

[221] Myers, B.A., *Creating User Interfaces By Demonstration*, Academic Press, Boston, 1988.

[222] Myers, B.A., *User Interface Tools: Introduction and Survey*, IEEE Software, January 1989, pp. 15-23.

[223] Myers, B.A., *Encapsulating Interactive Behaviors*, in: Proceedings Human Factors in Computing Systems, Bice, K., Lewis, C. (Eds.), Austin, Texas, 1989, pp. 319-324.

[224] Myers, B.A., Vander Zanden, B., Dannenberg, R.B., *Creating Graphical Interactive Application Objects by Demonstration*, in: Proceedings UIST'89, Nov. 1989, pp. 95-104.

[225] Myers, B.A. (Ed.), *The Garnet Toolkit Reference Manuals Support for Highly-Interactive, Graphical User Interfaces in Lisp*, Carnegie Mellon University, Computer Science Department, CMU-CS-90-117, March 1990.

[226] Myers, B.A., Giuse, D.A., Dannenberg, R.B., Vander Zanden, B., Kosbie, D.S., Pervin, E., Mickish, A., Marchal, P., *Garnet: Comprehensive Support for Graphical, Highly Interactive User Interfaces*, IEEE Computer, 23, 11, 1990, pp. 71-85.

[227] Myers, B.A. (Ed.), *Languages for Developing User Interfaces*, Jones and Bartlett Publishers, 1992.

[228] Nebel, B., *Reasoning and Revision in Hybrid Representation Systems*, Lecture Notes in Artificial Intelligence, Vol. 422, Springer-Verlag, 1990.

[229] Neches, R., Swartout, W.R., Moore, J.D., *Enhanced Maintenance and Explanation of Expert Systems Through Explicit Models of Their Development*, IEEE Transactions on Software Engineering, Vol. SE-11, No. 11, November 1985, pp. 1337-1351.

[230] Neches, R., Foley, J., Szekely, P., Sukaviriya, P., Luo, P., Kovacevic, S., Hudson, S., *Knowledgeable Development Environments Using Shared Design Models*, in: Proceedings ACM/AAAI International Workshop on Intelligent User Interfaces, Orlando, Fl. Jan., 1993.

[231] Neil, J.G., Shapiro, S.C., *Intelligent Multi-Media Interface Technology*, in: [299], pp. 11-43.

[232] Nielson, I., Lee, J., *Conversations with Graphics: Implications for the Design of Natural Language/ Graphics Interfaces*, Int. J. Human-Computer Studies, 1994, 40, pp. 509-541.

[233] Norman, D.A., Draper, S.W. (Hrsg.), *User Centered System Design*, Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1986.

[234] Norman, D.A., *Dinge des Alltags, The Psychology Of Everyday Things*, Campus Verl., Frankfurt, 1989.

[235] Norman, D.A., *Cognitive Artifacts, in: Designing Interaction*, Carroll, J.M. (Ed.), Cabridge University Press, 1991, pp. 17-38.

[236] OPEN LOOK, *Graphical User Interface Application Style Guidelines*, Sun Microsystems, Inc., AT&T, Addison-Wesley, 1990.

[237] Owsnicki-Klewe, B., v. Luck, K., Nebel, B., *Wissensrepräsentation und Logik – Eine Einführung*, in: Görz, G., (Hrsg.), Einführung in die Künstliche Intelligenz, 2. Auflage, Addison-Wesley, 1995, pp. 15-58.

[238] Padgham, L, Lambrix, P., *A Framework for Part-of Hierarchies in Terminological Logics*, in: KR'94 Principles of Knowledge Representation and Reasoning, Doyle, J., Sandewall, E., Torasso, P. (Hrsg.) Morgan-Kaufmann Publishers, Inc., 1994, pp. 484-496.

[239] Pasternak, B., Neumann, B., *Adaptable Drawing Interpretation Using Object-Oriented and Constraint-Based Graphic Specification*, in: Proceedings ICDAR. 1993. Tsukuba (Ed.), pp. 359 - 364.

[240] Pasternak, B., Neumann, B., *ADIK - An Adaptable Drawing Interpretation Kernel. in Scientific Converence on AI*, XPS, NL. 1993. Avignon: pp. 531 - 540.

[241] Pasternak, B., *Processing Imprecise and Structural Distorted Line Drawings by an Adaptable Drawing Interpretation Kernel*, in: Proceedings DAS, 1994, Kaiserslautern, Germany, DFKI, pp. 349 - 365.

[242] Patel-Schneider, P.F., *Small can be Beautiful in Knowledge Representation*, AI Technical Report 37, Schlumber Palo Alto Research Center, October 1984.

[243] Patel-Schneider, P.F., Brachman, R.J., Levesque, H.J., *ARGON: Knowledge Representation Meets Information Retrieval*, Fairchild Technical Report 654, Schlumberger Palo Alto Research Center, September 1984.

[244] Patel-Schneider, P.F., Swartout, B., *Description Logic Specification from the KRSS Effort*, Unveröffentlichter Entwurf, `ksl.stanford.edu:/pub/knowledge-sharing/papers/dl-spec.ps`.

[245] Paternó, F. (Hrsg.), *Interactive Systems: Design, Specification, and Verification*, 1st Eurographics Workshop, Bocca di Magra, Italy, June 1994, Springer-Verlag, 1995.

[246] Petre, M., Green, T.R.G., *Learning to Read Graphics: Some Evidence that 'Seeing' an Information Display is an Acquired Skill*, Journal of Visual Languages and Computing, Vol. 4, 1993, pp. 55-70.

[247] Pfaff, G.E. (Ed.), *User Interface Management Systems*, Eurographics Seminar Series, Springer Verlag, 1985.

[248] Philips, M.D., Bashinski, H.S., Ammerman, H.L., Fligg, C.M., *A Task Analytic Approach to Dialog Design*, in: [124], pp. 835-857.

[249] Pinker, S., *Visual Cognition*, A Bradford Book, The MIT Press, 1985.

[250] Polson, P.G., Lewis, C.L., *Theory-Based Design for Easily Learned Interfaces*, Human-Computer Interaction, Vol. 5, 1990. pp. 191-220.

[251] Pribbenow, S., *Modeling Physical Objects: Reasoning about (Different Kinds of) Parts*, in: Proceedings of the TSM'95 Workshop (Time, Space, Movement), Bonas, France, 1995.

[252] Puerta, A.R., Eriksson, H., Gennari, J.H., Musen, M., *Beyond Data Models for Automated User Interface Generation*, in: [53], pp. 353-366.

[253] Rambow, O., *Domain Communication Knowledge*, in: Proceedings of the ACL-93 workshop on Intentionality and Structure in Relations, Columbus, Ohio, June 1993, pp. 87-93.

[254] Rasmussen, J., *Information Processing and Human-Machine Interaction*, North-Holland, 1986.

[255] Reiterer, H., *User Interface Evaluation and Design - Research Results of the Projects Evaluation of Dialog Systems (EVADIS) and User Interface Design Assistance (IDA)*, GMD-Bericht Nr. 237, Gesellschaft für Mathematik und Datenverarbeitung, R. Oldenbourg Verlag, München/Wien 1994.

[256] Reithinger, N., *Eine parallele Architektur zur inkrementellen Generierung mulitmodaler Dialogbeiträge*, in German, Infix, DISKI1: Dissertationen zur Künstlichen Intelligenz Band 1, 1991.

[257] Reitman Olsen, J., Olson, G.M., *The Growth of Cognitive Modeling in Human-Computer Interaction*, Human-Computer Interaction, Vol. 5, 1990, pp. 221-265.

[258] Resnick, L.A., Borgida, A., Brachman, R.J., McGuiness, D.L., Patel-Schneider, P.F., Zalondek, K.C., *CLASSIC Description and Reference Manual For the Common Lisp Implementation*, Version 2.2.

[259] Retz-Schmidt, G., *Die Interpretation des Verhaltens mehrerer Akteure in Szenenfolgen* (in German), Informatik Fachberichte 308, Springer-Verlag, 1992.

[260] Retz-Schmidt, G., *Recognizing Intention, Interactions, and Causes of Plan Failures*, in: User Modeling and User-Adapted Interaction, Vol 1, Kluwer Academic Publishers 1991, pp. 173-202.

[261] Rist, T., André, E., *Incorporating Graphics Design and Realization into the Multimodal Presentation System* WIP, DFKI Research Report RR-92-44, September 1992.

[262] Robertson, G.G., Mackinlay, J.D., Card, S.K., *Cone Trees: Animated 3D Visualizations of Hierarchical Information*, Proceedings CHI'91, Computer-Human Interaction, ACM Press, 1991, pp. 189-194.

[263] Robertson, P.K., Visualizing Color Gamuts: A User Interface for the Effective Use of Perceptual Color Spaces in Data Displays, in: IEEE Computer Graphics & Applications, September 1988, pp 50-64.

[264] Rome, E., *Wahrnehmungspsychologie, Bilderkennung und der Graphikdesigner*, TASSO-Report Nr. 36, Gesellschaft für Mathematik und Datenverarbeitung, February 1992.

[265] Rome, E., *Grafiksuche und Gestalterkennung*, in: [71], pp. 209-226.

[266] Roth, S.F., Mattis, J., Mesnard, X., *Graphics and Natural Language as Components of Automatic Explanation*, in: [299], pp. 207-239.

[267] Roth, S.F., Mattis, J., *Data Characterization in Intelligent Graphics Presentation*, in: Proceedings CHI'90 Human Factors in Computing Systems, ACM Press, 1990, pp. 193-200.

[268] Roth, S.F., Mattis, J., *Automating the Presentation of Information*, in: Proceedings Seventh Conference on Artificial Intelligence Applications, IEEE, 1991, pp. 90-97.

[269] Roth, S.F., Hefley, W.E., *Intelligent Multimedia Presentation Systems: Research and Principles*, in [204], pp. 13-58.

[270] Roth, S.F., Kolojejchick, J., Mattis, J., Goldstein, J., *Interactive Graphic Design Using Automatic Presentation Knowledge*, in: Proceedings CHI '94, Human Factors in Computing Systems, pp. 112-117.

[271] Rutledge, G.W., *Dynamic Selection of Models*, Dissertation, Stanford University, Department of Computer Science, Report No. STAN-CS-TR-95-1549, März 1995.

[272] Sattler, U., A Concept Language for an Engineering Application With Part-Whole Relations, in: Proc. of the International Workshop on Description Logics, Borgida, A., Nardi, D. (Hrsg.), Rome, 1995, pp. 117-123.

[273] Schick, M., Kockskämper, S., Neumann, B., *Konstellationserkennung auf Basis einer hybriden Raumrepräsentation*, Behavior Memo 03-93, Universität Hamburg, 1993.

[274] Schmid, C., *Statistical Graphics: Design Principles And Practices*, Wiley, New York, 1983.

[275] Schmidt-Schauss, M., *Subsumption in KL-ONE is Undecidable*, in: Principles of Knowledge Representation and Reasoning: Proc. of the First Int. Conf. (KR'89), Brachman, R., Levesque, H.J., Reiter, R. (Hrsg.), Morgan Kaufmann Publishers 1989.

[276] Schreiber, G., Wielinga, B., Breuker, J. (Eds.), *KADS – A Principled Approach To Knowledge-Based System Development*, Academic Press, 1993.

[277] Schröder, C., Möller, R., Lutz, C., *Eine partielle logische Rekonstruktion von PLAKON/KONWERK*, in: Proc. WRKP-96, Workshop on Knowledge Representation and Configuration, DFKI-Dokument D-96-04, 1996, pp. 55-64.

[278] Schwarz, A., Berry, D.M., Shaviv, E., *Representing and Solving the Automated Building Design Problem*, Computer-Aided Design, Vol. 26, Number 9, September 1994.

[279] Seligmann, D.D., Feiner, S., *Automated Generation of Intent-Based 3D Illustrations*, Computer Graphics, Volume 25, Number 4, July 1991, pp. 123-132.

[280] Seligmann, D.D., Feiner, S., *Supporting Interactivity in Automated 3D Illustrations*, Proceedings of the 1993 International Workshop on Intelligent User Interfaces, Orlando Florida, ACM Press, Gray, W.D., Heffey, W.E., Murray, D. (Eds.), 1993, pp. 37-44.

[281] Selker, T., Koved, L., *Elements of Visual Language*, Research Report, IBM Research Division, T.J. Watson Research Center, RC 13929 (#62616), August 1988.

[282] Senay, H., Ignatius, E., *A Knowledge-Based System for Visualization Design*, IEEE Computer Graphics and Applications, Vol. 14, No. 6, November 1994.

[283] Serra, L., Chua, T.-S., Teh, W.-S., *A Model for Integrating Multimedia Information Around 3D Graphics Hierarchies*, in: The Visual Computer, Vol. 7, Springer-Verlag 1991, 1991, pp. 326-343.

[284] Shapiro, S.C. (Editor-in-Chief), *Encyclopedia of Artificial Intelligence*, John Wiley & Sons, 1987.

[285] Shneiderman, B., *Direct Manipulation: A Step Beyond Computing*, IEEE Computer, August 1983.

[286] Shneiderman, B., Panel: Anthropomorphism: From Eliza to Terminator 2, CHI'92, Massachusetts, Addison-Wesley, 1992, pp. 67-70.

[287] Sitter, S., Stein, A., *Modelling the interlocutionary aspects of information-seeking dialogues*, Arbeitspapiere der GMD 515, Gesellschaft für Mathematik und Datenverarbeitung, February 1991.

[288] Slack, J., Conati, C., *Effective Graphics: Accessing Spatial Relations*, Proceedings of Human-Computer Interaction HCI'92, York, 1992.

[289] Song, J., *Modeling Timed User-Interactions in Multimedia Documents*, IBM research Division, T.J. Watson Research Center, Research Report, RC 19933 (88543) 1995.

[290] Sowa, J. (Hrsg.), *Principles of Semantic Networks*, Morgan Kaufmann Publishers, Inc., 1990.

[291] Speel, P.-H., Patel-Schneider, P.F., *CLASSIC extended with physical whole-part relations*, in: Working Notes of the 1994 Description Logic Workshop, Juni 1994, Bonn, pp. 44-49.

[292] Spenke, M., Beilken, M., *An Overview of GINA - the Generic Interactive Application*, Esprit/ Eurographics Workshop on User Interface Management Systems and Environments, Lisbon, 4-6 June 90.

[293] Spenke, M., Berlage, T., *GINA - Ein objektorientiertes Integrationsmodell für die Benutzerschnittstelle des Assistenz-Computers*, in German, GMD-Jahresbericht 1989, Gesellschaft für Mathematik und Datenverarbeitung mbH, Institut für angewandete Informationstechnik.

[294] Steele, G.L., *Common Lisp The Language - Second Edition* , Digital Press, 1990.

[295] Stein, A., Thiel, U., *A Conversational Model of Multimodal Interaction*, Arbeitspapiere der GMD 741, Gesellschaft für Mathematik und Datenverarbeitung, March 1993.

[296] Struss, P., *What's in SD? Towards a Theory of Modeling for Diagnosis*, aus: Readings in Model-Based Diagnosis, Hauscher, W., Console, L., de Kleer, J. (Hrsg.), Morgan Kaufmann Publishers, Inc., 1992.

[297] Sugiyama, K., Shjiro, T., Toda, M., *Methodes for Visual Understanding of Hierarchical System Structures*, IEEE Transactions on Systems, Man and Cybernetics, Feb. 1981.

[298] Sukaviriya, P.N., *A Model-Based User Interface Architecture: Enhancing a Runtime Environment with Declarative Knowledge*, in: [245], pp. 181-197.

[299] Sullivan, J.W., Tyler, S.W. (Eds.), *Intelligent User Interfaces*, ACM Press,1991.

[300] Sundström, G., *User Modelling for Graphical Design in Complex Dynamic Environments: Concepts and Prototype Implementations*, Int. Journal Man-Machine Studies, 38, 1993, pp. 567-586.

[301] Suthers, D., Woolf, B., Cornell, M., *Steps from Explanation Planning to Model Construction Dialogues*, Proceedings AAAI'92, ACM Press, 1992, pp. 24-30.

[302] Suthers, D., *An Analysis of Explanation and its Implications for the Design of Explanation Planners*, Dissertation, University of Massachusetts, 1993.

[303] Swartout, W., *Producing Explanations and Justifications of Expert Consulting Programs*, Massachusetts Institute of Technology, Report No. MIT/LCS/TR-251, 1981.

[304] Swartout, W., XPLAIN - *A System for Creating and Explaining Expert Consulting Systems*, Artificial Intelligence, Vol. 21, No. 3, 1983, pp. 285-325.

[305] Swartout, W., Moore, J.D., *Explanation in Second Generation Expert Systems*, in: [63], pp. 543-585.

[306] Symbolics, *Programming the User Interface Vol A*, Symbolics Inc., September 1986.

[307] Symbolics, *Common Lisp Interface Manage (CLIM): Release 2.0*, Symbolics Inc., Jan. 1994.

[308] Szekely, P., *Structuring Programs to Support Intelligent Interfaces,* in: [299], pp. 445-464.

[309] Szekely, P., Luo, P., Neches, R., *Beyond Interface Builders: Model-Based Interface Tools*, in: Proceedings of INTERCHI'93, April 1993, pp. 383-390.

[310] Szekely, P., Luo, P., Neches, R., *Facilitating the Exploration of Interface Design Alternatives: The HUMANOID Model of Interface Design*. in: Proceedings of CHI'92, The National Conference on Computer-Human Interaction, May 1992, pp. 507-515.

[311] Szekely, P., Sukaviriya, P., Castells, P., Muthukumarasamy, J., Salcher, E., *Declarative Interface Models for User Interface Construction Tools: The MASTERMIND Approach*, in: Proc. EHCI'95, Working Conference on Engineering for Human-Computer Interaction, August 1995.

[312] Takahashi, S., Matsuoka, S., Yonezawa, A., *A General Framework for Bi-Directional Translation between Abstract and Pictorial Data*, Proceedings 4th Annual Symposium on User-Interface Software and Technology, UIST'91, November 11-13, 1991, ACM Press, pp. 165-174.

[313] Tamassia, R., Di Batista, G., Batini, C., *Automatic Drawing and Readability of Diagrams*, IEEE Transactions on Systems, Man and Cybernetics, Feb. 1988.

[314] ten Hagen, P.J.W., *Critique of the Seeheim Model*, in: User Interface Management and Design, Duce, D.A., Gomes, M.R., Hopgood, F.R.A., Lee, J.R. (Eds.), Eurographics Seminars, Springer Verlag, 1991, pp. 3-6.

[315] Tetzlaff, L., Schwartz, D.R., *The Use of Guidelines in Interface Design*, Proceedings CHI'91, ACM Press, 1991, 329-333.

[316] The Penman Natural Language Group, *The Penman Primer*, Information Sciences Institute of the University of Southern California, November, 1989.

[317] Thiel, U., *Konversationale Modellierung interaktiver Systeme: Ein Beispiel*, Proceedings GWAI'90, 14th German Workshop on Artificial Intelligence, Marburger, H. (Hrsg.), Informatik-Fachberichte 251, Springer-Verlag, 1990, pp. 65-74.

[318] Thies, M., *Planbasierte Hilfeverfahren für direktmanipulative Systeme: Erkennung, Vervollständigung und Visualisierung von Interaktionsplaenen*, in German, Dissertation, Stuttgart University, March 1994.

[319] Top, J., Akkermans, H., *Tasks and Ontologies in Engineering Modelling*, Int. J. Human-Computer Studies, Vol. 41, 1994, pp. 585-617.

[320] Treisman, A., *Properties, Parts, and Objects*, in: Handbook of Perception and Human Performance, Volume II, Cognitive Processes and Performance, John Wiley and Sons, 1986.

[321] Tufte, E., *The Visual Display Of Quantitative Information*, Graphics Press, Cheshire, Co, 1984.

[322] Tufte, E., *Envisioning Information*, Graphics Press, Cheshire, Ct, 1990.

[323] Tufte, E., *Visual Design Of The User Interface*, IBM New York.

[324] Ulich, E., *Arbeitspsychologie*, vdf-Verlag, 2. Auflage, Zürich, 1992.

[325] Ullman, S., *Visual Routines*, in: [249], pp. 97-159.

[326] Uschold, M., King, M., *Towards a Methodology for Building Ontologies*, Workshop on Basis Ontological Issues in Knowledge Sharing, IJCAI-95, 1995.

[327] Uschold, M., King., M, Moralee, S., Zorgios, Y., *The Enterprise Ontology, Version 1.0*, 1995.

[328] Wahlster, W., *User and Discourse Models for Multimodal Communication*, in: [299], pp. 45-67.

[329] Wahlster, W., *One Word Says More Than a Thousand Pictures. On the Automatic Verbalization of the Results of Image Seqeunce Analysis Systems*, SFB 314 (VITRA), Bericht Nr. 25, Februar 1988.

[330] Wahlster, W., André, E., Bandyopadhyay, S., Graf, W., Rist, T., *WIP: The Coordinated Generation of Multimodal Presentations from a Common Representation*, DFKI Research Report RR-91-08, February 1991.

[331] Wahlster, W., André, E., Finkler, W., Profitlich, H.-J., Rist, T., *Plan-Based Integration of Natural Language and Graphics Generation*. Artificial Intelligence, 63, 1993, pp. 387-427.

[332] Wanger, L.R., Ferwerda, J.A., Greenberg, D.P., *Perceiving Spatial Relationshiops in Computer-Generated Images*, IEEE Computer Graphics & Applications, May 1992, pp. 44-58.

[333] Warren, Y., Hägglund, S., Löwgren, J., Rankin, I., Sokolnicki, T., *Communication Knowledge for Knowledge Communication*, Int. J. Human-Computer Studies, 1991, 37, pp. 215-239.

[334] Weida, R., Litman, D., *Terminological Reasoning with Constraint Networks and an Application to Plan Recognition*, in: Proceedings KR'92 Principles of Knowledge Representation and Reasoning, Nebel, B., Rich, C., Swartout, W. (Eds.) Morgan Kaufmann Publishers, Inc., 1992, pp. 282-269.

[335] Weida, R., Litman, D., *Subsumption and Recognition of Heterogeneous Constraint Networks*, aus: Proceedings 10th Conference on Artificial Intelligence for Applications, IEEE Computer Society Press, 1994, pp. 381-388.

[336] Weir, G.R., Alty, J.L. (Hrsg.), *Human-Computer Interaction and Complex Systems*, Academic Press, 1991.

[337] Wenger, E., *Artificial Intelligence and Tutoring Systems – Computational and Cognitive Approaches to the Communication of Knowledge*, Morgan Kaufmann Publishers, Inc., 1987.

[338] Wiecha, C. Bennett, W., Boies, S., Gould, J., Greene, S., *ITS: A Tool for Rapidly Developing Interactive Applications*, in: ACM Transactions on Information Systems, Vol. 8, No. 3, July 1990, pp. 204-236.

[339] Wielinga, B., Schreiber, G., Breuker, J., *Modelling Expertise*, in: [276], pp. 21-46.

[340] Wiese, D., *Überlegungen zum Einsatz von Graphgeneratoren bei der Erzeugung von Visualisierungen: domänenabhängige und -unabhängige Formatierungen*, in German, Studienarbeit, University of Hamburg, Computer Science Department, 1992.

[341] Winston, M.E., Chaffin, R., Herrman, D., *A Taxonomy of Part-Whole Relations*, Cognitive Science, 11, 1987, pp. 417-444.

[342] Wittenburg, K., Weitzman, L., *Visual Grammars and Incremental Parsing for Interface Languages*, 1990 IEEE Workshop on Visual Languages, IEEE Computer Society Press, 1990, pp. 111-118.

[343] Winizki, A., *Erklärung von Expertenlösungen in einem wissensbasierten Beratungsdialog*, Diploma Thesis, in German, University of Hamburg, Computer Science Department, FBI-HH-M-175/90, 1990.

[344] Wolf, S., Setzer, R., *Wissensverarbeitung mit KEE - Einführung in die Erstellung von Expertensystemen*, in German, Oldenbourg, Muenchen, 1991.

[345] Woods, W.A., Schmolze, J.G., *The KL-ONE Family*, in: Semantic Networks in Artificial Intelligence, Lehmann, F. (Ed.), Pergamon Press, 1992, pp. 133-177.

[346] Young, M., Taylor, R.N., Troup, D.B., *Software Environment Architectures and User Interface Facilities*, IEEE Transactions on Software Engineering, 14, 6, June 1988, pp. 697-707.

[347] Zdybel, F., Greenfeld, N., Yonke, M, Gibbons, J., *An Information Presentation System*, in: Proceedings of IJCAI 81, IJCAI, Vancouver, Canada, August 1981, pp. 978-984.

[348] Zukerman, I., *Content Planning based on a Model of a User's Beliefs and Inferences*, in: Proceedings UM'92, 3rd International Workshop on User Modeling, André, E., Cohen, R., Graf, W., Kass, B., Paris, C., Wahlster,W. (Eds.), Dokument D-92 17, Deutsches Forschungszentrum für Künstliche Intelligenz, August 1992, pp. 162-173.

[349] Zukerman, I., McConachy, R., *Consulting a User Model to Address a User's Inferences during Content Planning*, in: User Modeling and User-Adapted Interaction, 3, 1993, pp. 155-185.

Lebenslauf

Persönliche Daten:	6.6.64	geboren in Freiburg/NE
Schulbesuch:	1970-1974	Besuch der Grundschule in Wischhafen
	1974-1976	Besuch der Orientierungsstufe in Freiburg/NE
	1976-1983	Besuch der Gymnasiums in Hemmoor
	1983	Abitur
Wehrdienst:	1983-1984	Grundwehrdienst
Studium:	1984-1990	Studium der Informatik an der Universität Hamburg mit Nebenfach Medizin, Abschluß als Diplom-Informatiker
Berufstätigkeit:	3/90-3/91	Wiss. Mitarb. am Labor für Künstliche Intelligenz (LKI) an der Universität Hamburg im Projekt KISP (KI-Software-Praktikum) und im Projekt FIS (Fahrplan-InformationsSysteme)
	4/91-4/96	Wiss. Mitarb. am Arbeitsbereich KOGS des Fachbereichs Informatik der Universität Hamburg
	seit 4/96	Wiss. Mitarb. am LKI im Projekt INDIA (INtelligente Diagnose In der Anwendung)

Hamburg, den 5.7.96

Eidesstattliche Erklärung

Ich versichere hiermit an Eides statt, daß ich die Arbeit selbständig und nur mit den angegebenen Hilfsmittel angefertigt habe.

Hamburg, den 5.7.96