KfK 4933
November 1991

Integration des Altlastenexpertensystems XUMA in ein Umweltinformationssystem

M. Overlack
Institut für Datenverarbeitung in der Technik
Projekt Schadstoffbeherrschung in der Umwelt

Kernforschungszentrum Karlsruhe

KERNFORSCHUNGSZENTRUM KARLSRUHE

Institut für Datenverarbeitung in der Technik
Projekt Schadstoffbeherrschung in der Umwelt

KfK 4933

Integration des Altlastenexpertensystems XUMA in ein Umweltinformationssystem

Martin Overlack

Kernforschungszentrum Karlsruhe GmbH, Karlsruhe

Als Manuskript gedruckt

Kernforschungszentrum Karlsruhe GmbH
Postfach 3640, 7500 Karlsruhe 1

ISSN 0303-4003

Zusammenfassung

Integration des Altlastenexpertensystems XUMA in ein Umweltinformationssystem

Ausgehend von einer allgemeinen Einführung in die Integrationsproblematik werden Anforderungen aus unterschiedlichen Sichten an die die Integration betreffenden Systemkomponenten herausgearbeitet. In einem weiteren Schritt werden hieraus drei Integrationsansätze entwickelt, die zugleich auch eine Klassifizierung bzw. Abstufung der Integration darstellen. Von diesen Ansätzen wird einer - der anwendungsorientierte Ansatz - eingehend behandelt. Das hierfür entwickelte Basissystem wird vorgestellt, wobei der Schwerpunkt dieser Ausführungen auf einer Dialogbeschreibungssprache liegt, mit deren Hilfe Dialoge abstrakt definiert werden können. Die Probleme, die bei der Umsetzung des Basissystem-Konzeptes auf die speziellen Anforderungen von XUMA auftreten, werden beschrieben und abschließend wird dessen Durchführbarkeit mit Hilfe eines User Interface Management System - Werkzeugs prototypisch gezeigt.

Abstract

Integration of XUMA the Expert System on Risk Assessment of Contaminated Sites into an Environmental Information System

On the basis of a general introduction into the problem of integration, requirements to be met by the system components relevant for integration are determined from various points of view. In the following step, three integration approaches representing a classification or graduation of integration are developed. One of these approaches, namely, the application-oriented approach, is investigated in further detail. The respective basic system developed is presented with the emphasis lying on a dialog description language, by means of which dialogs can be defined in an abstract manner. The problems arising when transferring the basic system concept to the special requirements of XUMA are described. Finally, the practicability of the concept is shown by means of a prototype using a User Interface Management System tool.

Inhaltsverzeichnis

1. Einleitung

Der Einsatz informationstechnischer Werkzeuge in den Bereichen des aktiven und passiven Umweltschutzes gewinnt zunehmend an Bedeutung. Schon allein die Komplexität und der Umfang umweltrelevanter Informationen macht ihren Einsatz sinnvoll. Mit ihrer Hilfe können z.B. Gefährdungen und Bedrohungen der Umwelt erkannt, analysiert, überwacht und deren Beseitigung unterstützt werden. In gleichem Maße sind sie im aktiven Umweltschutz zu der Erforschung und Entwicklung umweltschonender (Produktions-) Verfahren einzusetzen, welche Belastungen der Umwelt von vorneherein vermeiden. Einen Überblick über die Bedeutung und die Einsatzgebiete der Umweltinformatik findet man in [Trauboth 87], [Weizsäcker 88], [Jaeschke 89] und [Page 90/1 und 2].)

Für die Entwicklung und den Einsatz informationstechnischer Methoden im Umweltbereich wurde in den letzten Jahren der Begriff der Umweltinformatik eingeführt. Er umschreibt kein eigenständiges Forschungsgebiet der Informatik, sondern wird einerseits durch den Anwendungsbereich und andererseits durch die in ihm angewandten Methoden charakterisiert. Typische Eigenschaften des Aufgabengebietes sind unter anderem :

- große, komplexe Datenmengen
- mehrdimensionale Objekte mit Raumbezug
- eine Vielzahl komplexer Objektbeziehungen
- unsicheres, vages Wissen

Die Aufgaben im Umweltbereich sind nur bewältigbar, wenn es gelingt, "...die verschiedenartigen Daten zu integrieren und daraus Informationen abzuleiten..." [Page 90/1].

Umweltinformationssysteme bilden hierfür eine geeignete Plattform. Unter einem Umweltinformationssystem (UIS)[1] versteht man ein komplexes Informationssystem für umweltrelevante Daten, welches in fachbezogene Komponenten (Fachinformationssysteme) unterteilt ist. Seine Aufgabe ist es, problemlösungs-

[1] Der Begriff Umweltinformationssystem ist in der Literatur nur unzureichend definiert. Er wird vielmehr durch bereits existierende Systeme geprägt (Kapitel 3 : Umweltinformationssysteme).

relevante, entscheidungsunterstützende Informationen zu beschaffen und sie in einer problembezogenen Darstellungsform zu visualisieren.

Das geringe Alter des Forschungsgebietes Umwelt(informatik) sowie die hohen Anforderungen, die bei der Modellierung von Umweltdaten einzuhalten sind, führten dazu, daß viele der bisher entwickelten informationstechnischen Systeme "Insellösungen" mit prototypischem bzw. Pilotcharakter sind. Sie sind für die Bearbeitung eines begrenzten Teilgebietes konzipiert und daher für die Lösung komplexer Umweltprobleme nur ungenügend geeignet. Jedoch können sie hierzu den auf ihr Anwendungsgebiet bezogenen Anteil beitragen, sofern sie untereinander gekoppelt bzw. in ein Verbundsystem (z.B. ein UIS) integriert werden.

Das Expertensystem zur Beurteilung der Umweltgefährlichkeit von Altlasten (XUMA) ist ein solches Umweltinformatik-System. Das Projekt wurde initiiert, um die mit der Erkundung, Beurteilung und Sanierung von Altlasten befaßten Sachbearbeiter der Wasserwirtschaftsverwaltung in Baden-Württemberg bei der Bewältigung ihrer Aufgaben mit den Methoden wissensbasierter Systeme zu unterstützen [Weidemann 89]. Ein Prototyp des Systems befindet sich seit Juni 1990 im Testeinsatz bei der Landesanstalt für Umweltschutz in Karlsruhe.

Diese vorliegende Arbeit geht von dem Ziel aus, XUMA in ein Umweltinformationssystem zu integrieren, um es damit einem größeren Anwenderkreis verfügbar zu machen.

Dazu sollen die mit der Integration von DV-Systemen einhergehenden Probleme beleuchtet und ein allgemeines Konzept entwickelt werden (Kapitel 4), welches dann auf XUMA übertragen wird (Kapitel 5). Mit Hilfe eines Prototypen soll die Durchführbarkeit des entsprechenden Konzeptes überprüft werden (Kapitel 6).

Bevor auf das Konzept eingegangen wird, soll zum besseren Verständnis der Problematik in den folgenden beiden Kapiteln das System XUMA kurz vorgestellt und auf Umweltinformationssysteme im allgemeinen eingegangen werden.

2. Das Altlastenexpertensystem XUMA

2.1 Motivation und Ziele

Die Altlastenproblematik hat in den letzten Jahren stark an Bedeutung gewonnen; Schätzungen gehen aus von derzeit ca. 80.000 Altlasten im Gebiet der Bundesrepublik. Die Zuständigkeit der Altlasten obliegt den Kommunen, die über Umfang, Reihenfolge und Art der Sanierung zu entscheiden haben und sich einem akuten Handlungsbedarf gegenüber sehen. Wesentlich für diese Entscheidung ist der Handlungsbedarf, der sich aus der vergleichenden Beurteilung der Gefährlichkeit der Altlasten ergibt [UM 88]. Zu diesem Zweck wurde XUMA (Expertensystem Umweltgefährlichkeit von Altlasten) im Rahmen eines Gemeinschaftsprojekts des Instituts für Datenverarbeitung in der Technik des Kernforschungszentrums Karlsruhe (KfK) und des Instituts für Altlastensanierung der Landesanstalt für Umweltschutz (LfU) des Landes Baden-Württemberg entwickelt [Geiger 89, 90, 91], [Weidemann 88, 89, 91], [Clausen 89], [Huber 88]. Es soll den Sacharbeitern das Wissen der wenigen Fachexperten auf diesem Gebiet leichter zugänglich machen und sicherstellen, daß die Erfahrungen aus den Sanierungen sowie andere neue Erkenntnisse unverzüglich in die Beurteilungen einfließen. Daneben soll das System zur landesweiten Vereinheitlichung des Vorgehens sowie der Beurteilungskriterien in Baden-Württemberg beitragen.

2.2 Überblick

Das Expertensystem XUMA wird in der Literatur ausführlich behandelt (s.o.). Die eigentliche Funktionalität des Systemes ist in dieser Arbeit von geringerer Bedeutung, wohingegen systemtechnische Rahmenbedingungen einen höheren Stellenwert einnehmen. Auf letztere wird in Kapitel 5 "Integration von XUMA in ein UIS" eingegangen. Dieses Kapitel soll einen Überblick über XUMA und das Anwendungsgebiet liefern.

Das allgemeine Vorgehen bei der Beurteilung einer Altlast ist der Abbildung 2.1 zu entnehmen.

Das System unterstützt die Altlasten-Sachbearbeiter in drei Phasen ihrer Tätigkeit:

- In der Phase der flächendeckenden Erfassung der Altlasten wird eine Kennziffer für die Umweltgefährlichkeit der Altlast ermittelt, welche zur Prioritätensetzug bei der weiteren Untersuchung der Altlasten dient.

- Ausgehend von konkreten Branchen- und Stoffhinweisen der jeweiligen Altlast wird die Zusammenstellung der Analysenparameter für die chemisch-analytische Untersuchung unterstützt.

- Nachdem die Proben der Altlast entsprechend dem Analysenplan laboranalytisch untersucht worden sind, können die Untersuchungsergebnisse erfaßt und beurteilt werden. Die Erstellung einer Stellungsnahme in Form eines Gutachtens wird unterstützt.

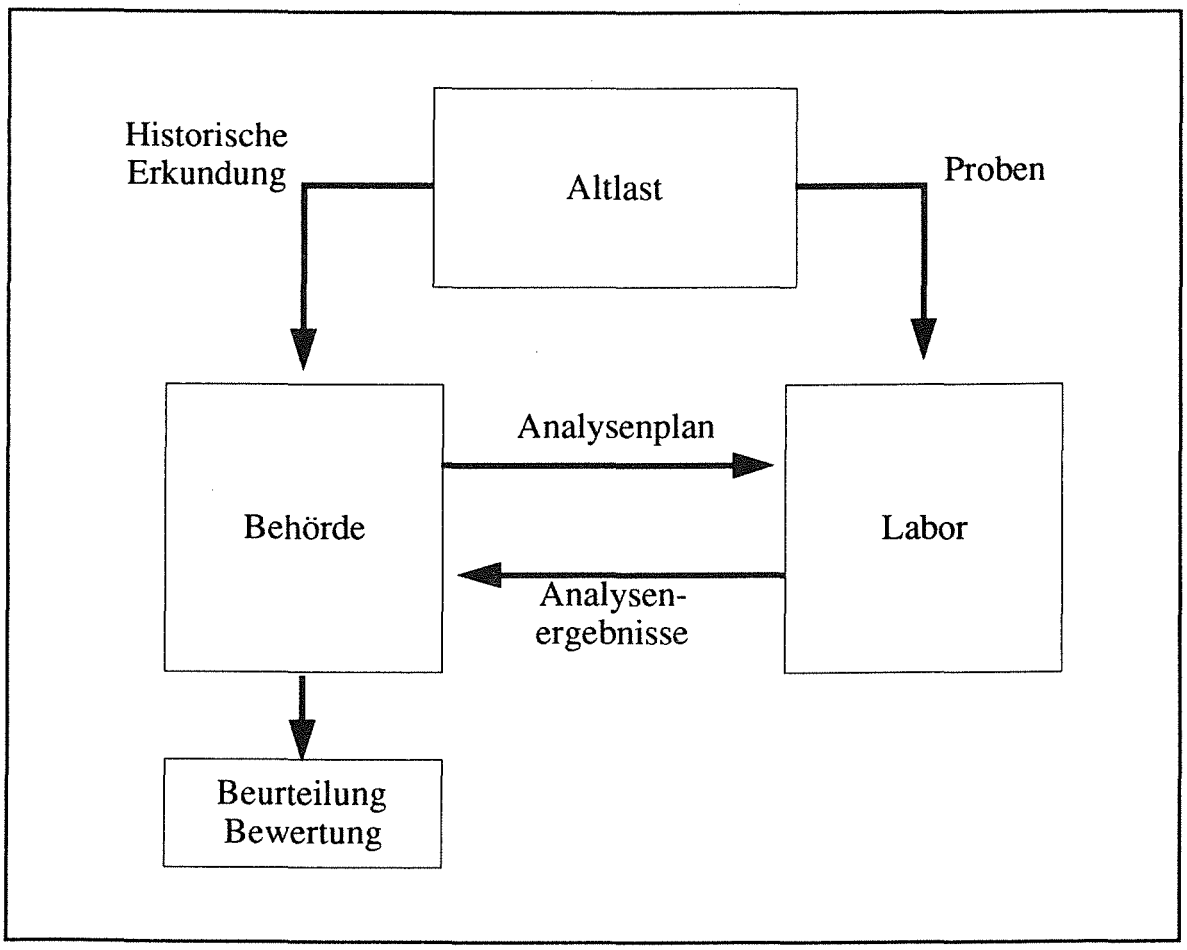


Abb. 2.1: Umfeld von XUMA

2.3 Funktionen von XUMA

In diesem Abschnitt sollen die Grundfunktionen von XUMA vorgestellt werden. Eine Übersicht über diese und deren Bezug zur Altlastenbewertung gibt die Abbildung 2.2 wieder.

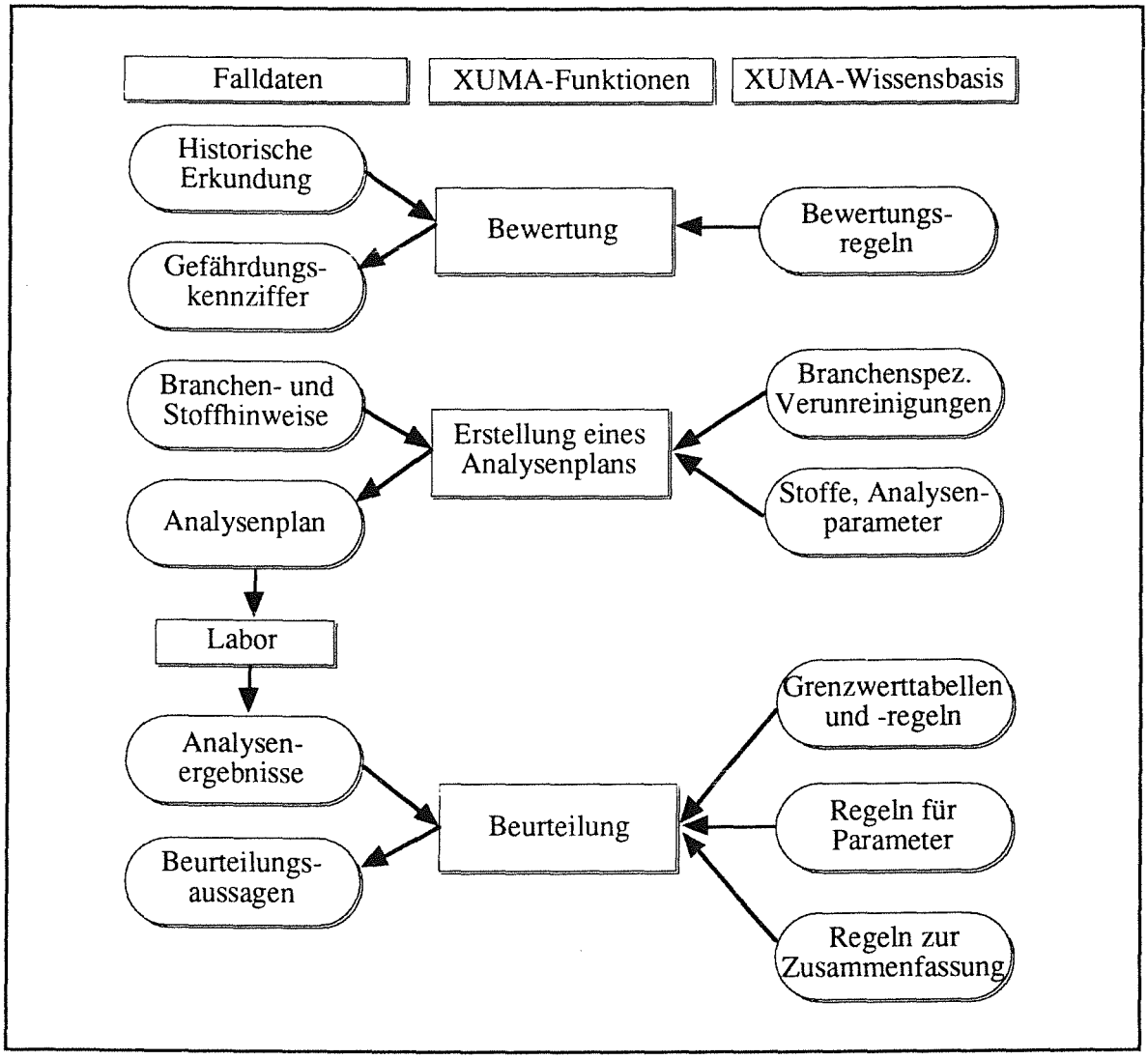


Abb. 2.2: Funktionen und Datenfluß in XUMA (ohne Wissensakquisition)

Erstellung des Analysenplans

Zur Ermittlung der von einer Altlast möglicherweise ausgehenden Gefährdung und gegebenfalls zur Vorbereitung einer Sanierung ist eine chemisch-analytische Erkundung des Standortes erforderlich. Aus Zeit- und Kostengründen ist es

praktisch nicht möglich, in einem dichten Raster die gesamte Palette möglicher Schadstoffe zu messen. Damit ergibt sich das Problem, welche Analysenparameter im jeweiligen Fall die geeignetsten sein könnten.

In praxi versucht man daher, vor Festlegung des Meßprogramms konkrete Hinweise über die möglichen Inhaltsstoffe der Altlast zu erhalten und daraus einen Analysenplan abzuleiten. Hierfür sind einerseits Hinweise wie z.B. der früheren Nutzung des Geländes als auch Wissen über branchenspezifische Abfälle sowie über Stoffe erforderlich.

Erfassen von Analysen

Mit Hilfe dieser Funktion werden die Ergebnisse der chemisch-analytischen Analyse in das System eingegeben. XUMA sieht unterschiedliche Analysenarten zu einem Altlastenfall vor :

- Brutto-Analysen von Abfall- oder Bodenproben,
- Analysen des wäßrigen Eluats einer Abfall- oder Bodenprobe,
- Wasser-Analysen von Grund-, Sicker- und Oberflächenwasserproben und
- Gas-Analysen von Deponiegas, Bodenluft bzw. -gas und atmosphärischer Luft.

Es werden folgende hierarchische Datenstrukturen verwendet:

Falldaten
Ein Fall wird durch Attribute wie z.B. Gemeinde, Flurstück und Betrieb identifiziert.

Probendaten
Charakeristisch für eine Probe sind Merkmale wie Art der Probe, Probennahmestelle und Datum der Probennahme

Analysendaten
Die Anlayse ist durch Attribute wie Untersuchungsobjekt (Orginalprobe oder Eluat), Labor-Nr. und Protokoll-Datum gekennzeichnet.

Beurteilung

In der Beurteilung werden auf der Basis der Analysenergebnisse Aussagen zur Einschätzung der Umweltgefährlichkeit der Altlast, Hinweise zum weiteren Untersuchungsbedarf sowie andere Aussagen (z.B. Hinweise auf Inkonsistenzen in den Analysendaten, Statistiken) abgeleitet und festgehalten, um so eine Stellungnahme in Art eines Gutachtens zu ermöglichen.

Bewertung

Bei der Bewertung wird entsprechend dem für Baden-Württemberg entwickelten Bewertungsverfahren [UM 88] in zwei Schritten vorgegangen. Im ersten Schritt wird die "Stoffgefährlichkeit in Vergleichslage" abgeschätzt, d.h. es wird angenommen, eine bestimmte Menge des zu bewertenden Boden- oder Abfallstoffes wäre unter festgelegten hypothetischen Lagebedingungen deponiert und es wird die Gefährlichkeit des Boden- oder Abfallstoffes bezogen auf diese Vergleichslage bewertet. Ziel dieses Schrittes ist es, einen "Risiko-Wert in Vergleichslage" zu ermitteln, der zum Vergleich der Stoffgefährlichkeit verschiedener Altlasten dient. Im zweiten Schritt, der "Berücksichtigung der örtlichen Verhältnisse", werden dann die tatsächlichen Lagebedingungen der Altlast und die Abfallmenge in die Beurteilung der Umweltgefährlichkeit einbezogen. In diesem Schritt soll ein Zahlenwert für das "maßgebliche Risiko" ermittelt werden, der zur Prioritätensetzung bei der Untersuchung und Sanierung von Altlasten herangezogen werden kann.

Wissensakquisition

Mit der Wissensakquisitionskomponente sollen die dazu authorisierten Experten der LfU in die Lage versetzt werden, innerhalb eines vorgegebenen Rahmens ohne Programmierkenntnisse neue Objekte und Regeln in die Wissensbasis einfügen sowie bestehende Objekte und Regeln ändern zu können. Solche Objekte sind insbesondere Branchen, Stoffe, Analysenparameter, Grenzwerttabellen, Regeln zu Grenzwerttabellen und Analysenparametern sowei Regeln zur Zusammenfassung der Bewertung auf Analysen-, Proben- und Fall-Ebene.

2.4 Stand des Vorhabens

In einer ersten Ausbaustufe wurde ein Prototyp des Expertensystems entwickelt, das die Funktionen "Erstellen eines Analysenplanes", "Erfassung von Analysen" und "Beurteilung" sowie die Erklärungskomponente enthält und dessen Wissensbasis die Branche Kohleveredlungsbetriebe umfaßt. Dieser Prototyp befindet sich seit Juni 1990 im Testeinsatz in der LfU. Durch die enge Zusammenarbeit zwischen der LfU und dem KfK konnte das System in seiner Funktionalität und seiner Wissensbasis weiter ausgebaut werden. Derzeit wird intensiv an der "Bewertung" sowie an der Erweiterung des Einsatzgebietes (zusätzliche Branchen) gearbeitet. Neben dem funktionalen Ausbau von XUMA soll das System einem größeren Anwenderkreis zugänglich gemacht und für die Integration in Umweltinformationssysteme vorbereitet werden. Mit der zuletzt genannten Problematik beschäftigt sich die vorliegende Arbeit.

3. Umweltinformationssysteme (UIS)

3.1 Der Begriff UIS

Der Begriff des Umweltinformationssystems (UIS) wird in der Literatur nicht einheitlich festgelegt. Vielmehr ist er geprägt durch die bereits vorhandenen Systeme. Innerhalb des in Aufbau befindlichen Arbeitskreises "Umweltinformationssysteme" der Fachgruppe 4.6.1 "Informatik im Umweltschutz" der Gesellschaft für Informatik e.V. (GI) soll unter anderem eine solche Begriffsbildung vorgenommen werden. Aufgrund der in der Literatur genannten Definitionen und der Anforderungen an ein UIS soll versucht werden, die Komplexität eines Umweltinformationssystemes zu beschreiben.

Page, Jaeschke und Pillmann [Page 90/1] verwenden folgende Definition:

> *"Informatik-System zur Erfassung von Umweltdaten (Meßwerten, Fakten wie chemische Stoffeigenschaften oder Abfallarten, Dokumenten wie Umweltgesetze oder Umweltliteratur), deren Speicherung, Organisation, Integration und Wiedergabe in Form von Umweltinformation. Die inhaltliche Erschließung zielt meist auf die Auswahl und Zusammenstellung der Umweltdaten (z.B. in der Berichterstattung)."*

In dem selben Artikel wird in Bezug auf Non-Standard-Datenbankanwendungen ein weitere Definition angegeben :

> *"Aufgrund ihres konkreten Raumbezuges können Umweltinformationssysteme ... angesehen werden als erweiterte geographische Informationssysteme, die der Erfassung, Speicherung und Verarbeitung von raum-, zeit- und inhaltsbezogenen Daten zur Beschreibung des Zustandes und der Entwicklung der Umwelt hinsichtlich Belastungen und Gefährdungen dienen. "*

Diese Definitionen legen den Schwerpunkt auf die zu verwaltenden Umweltdaten und beinhalten noch keinerlei Hinweise auf eine Struktur eines UIS (was sicherlich auch nicht das Ziel der Autoren war).

Zunächst soll genauer auf die zu speichernden Umweltdaten eingegangen werden, da aus diesen die grundlegenden Anforderungen an ein UIS hervorgehen.

3.2 Umweltdaten

Unter dem Begriff Umweltdaten sollen alle umweltrelevanten Informationen verstanden werden, die zur Lösung von Umweltproblemen herangezogen werden können. Eine ausführliche Klassifizierung von Umweltdaten findet man in [Schimak 91], die an dieser Stelle auszugsweise wiedergegeben wird:

- *Topographische Daten*
  - digitale topograpische Karten
  - digitale Geländemodelle
  - digitalisierte und gescannte Luftbilder
  - Satellitendaten

- *Daten zur Raumstruktur*
  - -statistische Daten für Länder, Bezirke und Gemeinden über Bevölkerung, Wirtschaft, Siedlungsstruktur und Flächennutzung
  - Digitalkartierung der Siedlungs- und Infrastruktur, Flächennutzung, Verwaltungsgrenzen, Raumeinheiten der Raumordnung und des Umweltschutzes

- *Daten für Naturschutz, Artenschutz und Landschaftsschutz*
  - digitale Biotopkartierung und digitale Artenschutzkartierung
  - Digitalkartierung der Land- und Forstwirtschaft
  - digitale Waldschadenskartierung
  - rasterbezogene Sammlung ökologischer Daten

- *Daten zu Umweltbelastung und Umweltgefahren*, bezogen aus
  - Meß- und Informationssystemen zur Überwachung der Luftbelastung, Bodenbelastung und der Belastung von Grund- und Oberflächengewässern
  - EDV-gestützten Frühwarnsystemen für Lawinen, Hochwasser und Smog
  - Datenbanken umweltrelevanter Anlagen
  - Umweltchemikalien-Monitoring
  - Bioindikatormeßnetzen
  - Programmsystemen zur Unterstützung des Verwaltungsvollzuges in den Bereichen Abfall, Sondermüll, Lärmschutz und Strahlenschutz
  - Radioaktivitätsmeßsystemen

Diese Aufzählung läßt sich ergänzen durch [Page 90/1]:

- Umweltgesetze, die einerseits in dokumentarischer Form oder als einzuhaltende Grenzwerte abgelegt sein können.
- Umweltliteratur, mit deren Hilfe beispielsweise Methoden der Altlastensanierung ermittelt werden können.
- (chemische) Stoffeigenschaften

Mit Hilfe der genannten Umweltdatentypen kann nun eine Charakterisierung vorgenommen werden [Fuhr 90] :

- unsichere und unvollständige Daten. Oft existieren nur textuelle Beschreibungen der Sachverhalte.

- unterschiedliche Darstellungsformen
  - Text
  - Fakten
  - Elemente wissensbasierter Repräsentationsformalismen
  - räumliche Daten

- heterogene Datenstrukturen mit komplexen Objektbeziehungen
  - unterschiedliche Herkunft
  - spezielle Anwendungskontexte

Fast alle der oben genannten Daten beziehen sich auf geographische Basisinformationen. Diese stellen somit ein Bindeglied zwischen den unterschiedlichen Datenstrukturen dar und bilden das Fundament eines UIS.

3.3 Objekte, Sichten und Benutzerklassen

Umweltrelevante Informationen müssen in Abhängigkeit von dem Benutzer und der Problemstellung unterschiedlich aufbereitet sein. Je nach Benutzerklasse ergeben sich demnach verschiedene Sichten (views) auf die entsprechenden Daten. Das derzeit vom Umweltministerium des Landes Baden Württemberg entwickelte Umweltinformationssystem ist in verschiedene Schichten aufgeteilt, die ihren Benutzern jeweils die für sie notwendige Sicht auf die Informationen bietet, die sie zur Erfüllung ihres Aufgabengebietes benötigen ([Mayer-Föll 89], [Henning 89]). Dies wird durch Aggregation und Filterung der vorhanden Daten erreicht, so daß nur noch die entscheidungsorientierten Informationen dem Benutzer zukommen (Vermeidung einer Informationsflut). Die Klassifizierung wird in diesem Fall durch die administrative Bedeutung des Aufgabenge-

bietes des Benutzers vorgenommen. Zitat eines Beispieles aus [Henning 89] : "... Wasserwirtschaft : Während für die Führungsspitze im Umweltministerium in der Regel ein Überblick über die Wasserqualität im Lande ausreicht, benötigt die Verwaltungsebene der Wasserwirtschaft detaillierte Angaben über einzelne Belastungsstoffe und Flußabschnitte."

Denkbar, aber nicht ausreichend, ist eine andere Klassifizierung der Benutzer, die in der Abbildung 4.1 bereits angedeutet wird, in der drei Schichten des Gesamtsystems erkennbar sind. Es gäbe demnach drei Benutzerklassen:

- Eine, die für die Wartung des Wissens bzw. der Daten zuständig ist,
- eine, die sich durch die entsprechenden Einzelsysteme beschreiben läßt und
- die mit dem höchsten Aggregationsgrad, die der Benutzer des UIS.

Eine besondere Herausforderung stellen die multimedialen Informationen dar. Zum Beispiel umfaßt die historische Erkundung eines Altlastenstandortes Dokumente jeglicher Art, Filmmaterial, Tonbandaufnahmen, Gesprächsaufzeichnungen, usw. Die Bewertung des Stofftransports im Grundwasser in der Umgebung der Altlast schließt geographische Informationen, Simulationsberechnungen, usw. ein.

3.4 Das Umweltinformationssystem des Landes Baden-Württemberg

Viele der erwähnten umweltrelevanten Daten sind in einzelnen Fachinformationssystemen vorhanden. Ein praktizierter Ansatz zum Aufbau eines UIS ist die Integration der existierenden Informationssysteme zu einem Gesamtsystem. In praxi sind die derzeit eingesetzten Fachinformationssysteme aber aufgrund ihrer Architektur nach außen hin abgeschlossen und bieten keine Schnittstellen an, die die wesentliche Vorraussetzung für deren Integration ist.

Die Anwendergruppe von XUMA sind die Wasserwirtschaftsämter des Landes Baden-Württemberg. Diese setzen derzeit das Informationssystem KIWI ein (Arbeitsdatei für wasser- und abfallwirtschaftliche Objekte), welche in das ressortübergreifende Umweltinformationssystem des Landes Baden-Württemberg, kurz UIS, eingebettet ist. Aus diesem Grund wird das landesweite, integrierte UIS nun kurz vorgestellt, wobei der Schwerpunkt auf seiner Architektur liegt. Nähere Informationen zu diesem Thema entnehme man [Mayer-Föll 89] und [Henning 89].

Anmerkung:
In den letzten Abschnitten dieses Kapitels wird mit dem Begriff UIS das Umweltinformationssystem des Landes Baden-Württemberg bezeichnet.

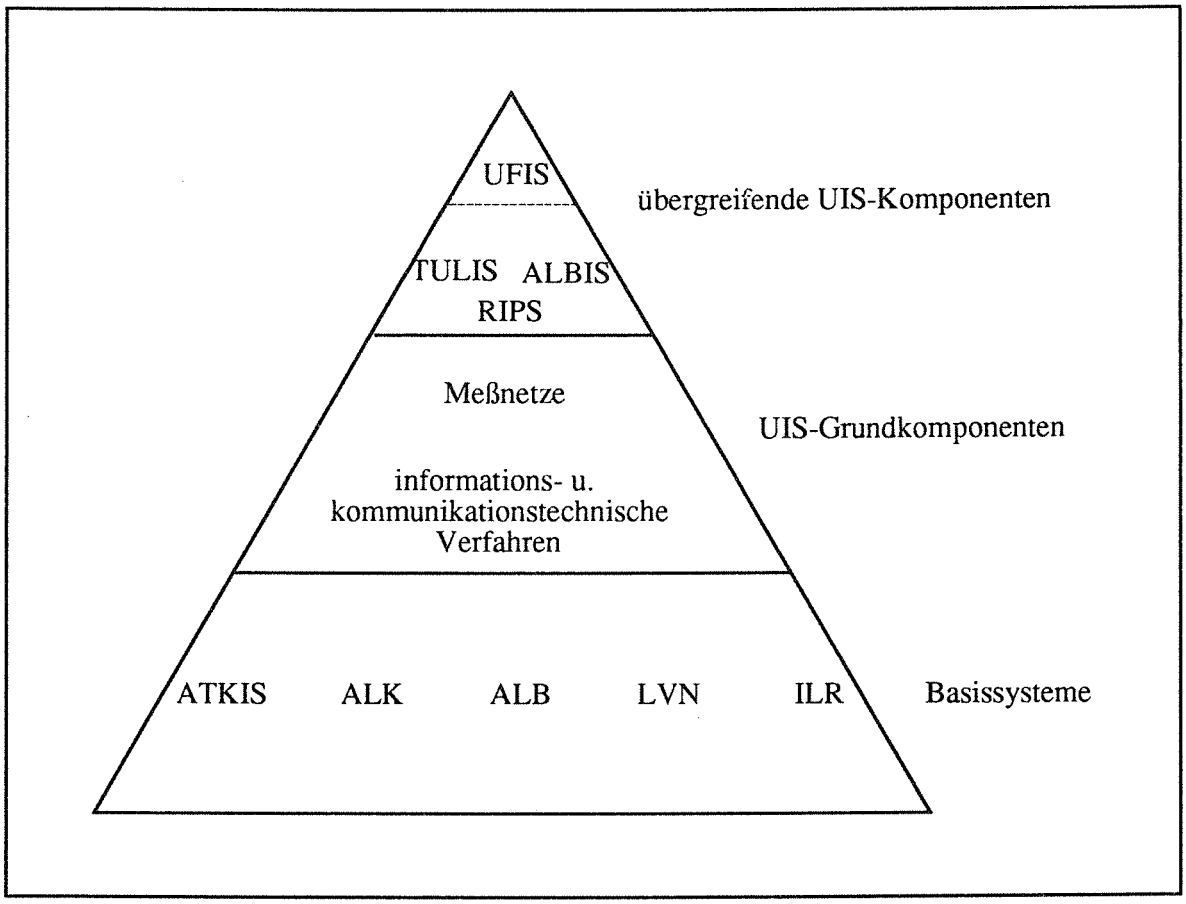


Abb. 3.1: Architektur des UIS (Baden-Württemberg)

Das UIS ist führungsorientiert aufgebaut. Die oberste Schicht (übergreifende Systemkomponenten) dient der Information der Führung der Ministerien (UFIS) sowie der Referats- und Dienststellenleiter des Umweltministeriums, des Ministeriums ländlicher Raum und der LfU sowie der Regierungspräsidien (TULIS, ALBIS, RIPS). Die UIS-Grundkomponenten sind Systeme zur Unterstützung der einzelnen Aufgaben mit Umweltbezug (Meßnetze für Boden, Wasser, Luft, Radioaktivität sowie informations- und kommunikationstechnischer Verfahren in Fachbereichen wie Wasser- und Abfallwirtschaft, Gewerbeaufsicht, usw.). Die unterste Ebene stellen die Basissysteme dar. Sie dienen der Erledigung allgemeiner Umweltaufgaben und sind die Infrastrukturvorraussetzung für das UIS (ALK, ATKIS, ALB, LVN und ILR).

Auf die einzelnen Komponenten soll nur kurz eingegangen werden:

Das Umwelt-Führungs-Informationssystem (**UFIS**) dient der Versorgung der Führung der Ministerien mit bedarfsgerecht aufbereiteten Informationen über den Zustand von Schutzgütern und die Wirkung von Maßnahmen in allen Umweltbereichen.

ALBIS (Arten-, Landschafts-, Biotop-Informationssystem)

TULIS (Technosphäre und Luft-Informationssystem)

RIPS (Räumliches Informations- und Planungssystem)
Dieses Informationssystem ist das Herzstück des UIS. Es verknüpft die räumlichen Informationen mit den Fachinformationen und enthält die Graphikanwendungen des UIS. Ihm zugrunde liegt das landeseinheitliche Graphikkonzept. Zielgruppe dieses Systemes sind Mitarbeiter aller Ressorts aller Ebenen. RIPS ist eine funktionale Erweiterung der Basiskomponenten ALK und ATKIS.

ATKIS (Amtliches Topographisch-Kartographisches Informationssystem)

ALK (Automatisierte Liegenschaftskarte)

ALB (Automatisiertes Liegenschaftsbuch)

LVN (Landesverwaltungsnetz)

ILR (Informationssystem ländlicher Raum)

4. Integration von einzelnen DV-Systemen in ein Umweltinformationssystem

In diesem Kapitel sollen zunächst Anforderungen erarbeitet werden, denen ein integriertes UIS genügen soll. Anschließend werden unterschiedliche Integrationsansätze vorgestellt, die diese Anforderungen umzusetzen versuchen. Es folgt ein Überblick über Realisierungsmöglichkeiten dieser Modelle und dabei auftretende Probleme. Am Ende werden zusammenfassend verschiedene Stufen zu einem integrierten System aufgezeigt. Die Formulierungen sind bewußt allgemein gehalten und sollen eine Einführung in die Integrationsproblematik darstellen.

Das Schema in Abbildung 4.1 soll die Komplexität der Integrationsproblematik verdeutlichen. Es geht bei diesem Vorhaben um die Verfügbarkeit von Diensten verschiedener Anwendungen auf einem beliebigen Zielsystem. Die entsprechenden Anwendungen können auf unterschiedlichen Plattformen lokalisiert sein.

Wissen bzw. Daten sind in der Realität verteilt und den entsprechenden Einzelsystemen zugeordnet. Jedoch wird durch deren Integration in ein UIS ein Zusammenhang zwischen den Daten verfügbar (siehe Abbildung 4.1).

Das UIS soll nach allen Seiten offen sein. Die Einzelsysteme können sowohl direkt von den Anwendern genutzt werden, für die diese ursprünglich konzipiert wurden, als auch von dem (in der Abbildung 4.1 dargestellten) Benutzer des UIS, vor dem verborgen bleiben kann, welches Einzelsystem die jeweilige Funktion des UIS tatsächlich realisiert.

Die Einzelsysteme bedürfen der ständigen Wartung zum Zwecke der Qualitätssicherung (sowohl Datenbestand als auch Anwendungsprogramm).

Ziel dieser Arbeit ist es, ein Konzept zu entwickeln, mit welchem Anwendungen in ein UIS integriert werden können. Dabei liegt der Schwerpunkt sowohl auf der Anwendung selbst als auch auf dem UIS, welches die notwendigen Voraussetzungen für eine Integration erfüllen muß.

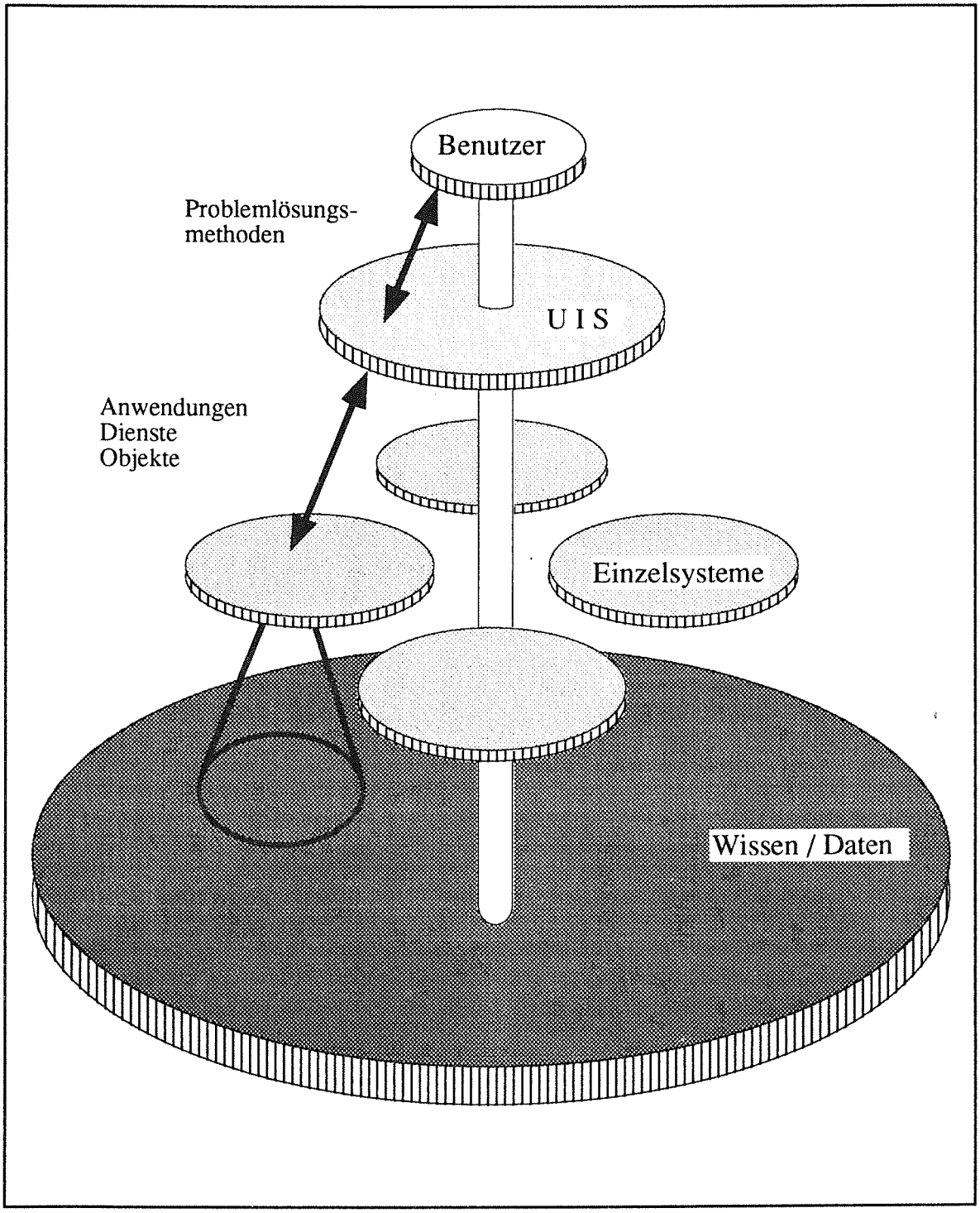


Abb. 4.1: Schematischer Aufbau eines UIS

4.1 Anforderungen

An ein integrationsfähiges UIS werden in Abhängigkeit des Betrachterstandpunktes unterschiedliche Anforderungen gestellt. In diesem Abschnitt werden diese den verschiedenen Sichtweisen entsprechend dargelegt. Dabei liegt der Schwerpunkt auf den qualitativen Merkmalen und weniger auf der Funktionalität der einzelnen Anwendungen. Diese werden unter anderem in [Fuhr 90] und [Mayer-Föll 89] behandelt. Eine Übersicht und Klassifikation von Qualitätsmerkmalen für Software-Produkte befindet sich in [Balzert 82].

Im folgenden werden die zentralen Anforderungen dargestellt, die in den weiteren einzelnen Abschnitten stärker detailliert werden.

Anforderungen ....

... aus der Sicht des Anwenders

Für den Anwender ist es nicht ersichtlich, daß es sich bei dem integrierten System um unterschiedliche Anwendungen handelt, die ihm nur ihre Dienste in einem heterogenen Verbundsystem zur Verfügung stellen. Seine Anforderungen richten sich also an das integrierende System.

Entscheidend für den Benutzer ist die Brauchbarkeit des Systems. Hierunter versteht [Balzert 82] Zuverlässigkeit, Effizienz und Benutzerfreundlichkeit. Während die beiden ersten Merkmale stark von den einzelnen Anwendungen abhängen, wird das Qualitätsmerkmal Benutzerfreundlichkeit dagegen entscheidend von der Dialogkomponente des UIS geprägt.

Der Benutzer soll eine **einheitliche** Benutzeroberfläche vorfinden, mit deren Hilfe er alle integrierten Anwendungen handhaben kann. Diese Oberfläche ihrerseits soll die Merkmale der ihm vertrauten Rechnerumgebung (Fenstersystem, Tastaturbelegung,...) einhalten. So soll sich das integrierte UIS den jeweiligen Gegebenheiten des Arbeitsplatzes bzw. des Benutzers anpassen. Beispielsweise soll ein Benutzer eines Arbeitsplatzes, der mit der OSF/Motif-Oberfläche ausgestattet ist, die Anwendungen in dem "look-and-feel" von OSF/Motif [OSF-SG 90] zur Verfügung haben, wohingegen der Benutzer einer OS/2 - Umgebung dieselbe Anwendung nach den Richtlinien des Style Guides des Presentation-Manager vorfinden soll. Das bedeutet, daß sich das integrierende UIS (und damit die einzelnen Anwendungen) den jeweiligen Richtlinien der aktuellen Benutzerumgebung anzupassen hat.

... an die zu integrierenden Anwendungen

Ob das Merkmal Benutzerfreundlichkeit für die einzelnen Anwendungen relevant ist, hängt davon ab, ob die Anwendung selbst oder das integrierte System für die Verfügbarkeit der Anwendung unter verschiedenen Benutzeroberflächen zuständig ist. Diese Unklarheit existiert bezüglich der Zuverlässigkeit und der Effizienz jedoch nicht. Die Einhaltung dieser Attribute ist zunächst Aufgabe der einzelnen Anwendung; eine Verschlechterung durch das integrierende System soll selbstverständlich vermieden werden.

Daß jede Anwendung und das integrierende System über entsprechende Schnittstellen verfügen muß, damit eine Integration überhaupt möglich, ist offensichtlich. Auf diese wird in diesem Kapitel noch gesondert eingegangen.

Der Nutzen einer Anwendung für das integrierende System ist abhängig von dem Umfang und der Art, wie die Anwendung ihre Dienste zur Verfügung stellt. Ist z.B. der Zugriff auf Informationen nur über die Dialogschnittstelle der Anwendung möglich, so sind die möglichen Vorteile der Integration für den Benutzer geringer, als wenn die Anwendung eine Datenbankschnittstelle zur Verfügung stellt, mit Hilfe derer eine Koppelung der Anwendungen in dem integrierenden System stattfinden kann, damit anwendungsübergreifende Aufgaben leichter lösbar werden.

Ein weitere Anforderung ist die Verfügbarkeit der Anwendung. In einem heterogenen Verbundsystem gibt es eine Vielzahl von Faktoren, die diesbezüglich negativen Einfluß nehmen können (z.B. starke Belastung des Netzwerkes, ausgeschöpfte Systemressourcen auf Anwendungsseite, usw.).

... der zu integrierenden DV-Systeme

Auch hier ergeben sich die Anforderungen an die Verfügbarkeit, nur dieses Mal aus umgekehrter Sicht. Eine Anwendung, die ihren Dienst anbietet, wird diesen einer möglichst großen Zahl von Benutzern anbieten wollen. Demnach darf ein Benutzer die Anwendung nicht unnötig blockieren bzw. belasten.

Die Mächtigkeit der angebotenen Dienste bringt auch Pflichten für den Nutzer mit sich. Um beim Beispiel einer Datenbankschnittstelle zu bleiben, ist der Benutzer des UIS (oder das UIS selbst) aufgefordert, bestimmte Integritätsbedingungen einzuhalten, wenn er Informationen ändert, löscht oder neue hinzufügt, um einen konsistenen Datenbestand zu gewährleisten, soweit dies nicht von der Anwendung selbst abgefangen wird.

... an das integrierende System

Das integrierte System steht im "Kreuzfeuer" der o.g. Anforderungen. Es muß sie alle beachten und versuchen jeder Seite gerecht zu werden. Besonders sei nochmals die Bedeutung der Benutzerschnittstelle hervorgehoben. Jede Anwendung verfügt in der Regel über ihre eigene Oberfläche, die geprägt ist von ihrem Anwendungsgebiet, der Ablaufumgebung und getroffener Design-Entscheidungen. Diese gilt es nun derart zu vereinheitlichen, daß der Benutzer des integrierten Systems, ohne sich jeweils neu einarbeiten zu müssen, alle Anwendungen nutzen kann, als ob es eine einzige wäre.

Berücksichtigung von Standards

Wie bereits aus den oben genannten Anforderungen ersichtlich ist, muß das integrierende UIS eine Vielzahl von Standards berücksichtigen, um eine einwandfreie Integration zu gewährleisten. Diese sind einerseits auf dem Kommunikationsbereich angesiedelt (SNA, DNA, ISO/OSI-BRM, ...), desweiteren auf dem Bereich der graphischen Benutzerschnittstellen[1] (OSF/Motif, Open Look, Macintosh, NeWS, Presentation Manager, NextStep, Open Desktop, ...) und auf dem Datenbankbereich (DB2, Oracle, Ingres, dBASE, 4th Dimension, RTMS, ...). Hinzu kommen plattform-spezifische Probleme durch die Verwendung unterschiedlicher Betriebssysteme (UNIX, VMS, VM, DOS, OS/2,...). Derzeit sind herstellerabhängige Integrationsvorhaben zu beobachten (New Wave (HP), SAA (IBM), X/Open). (Anmerkung: Die in Klammern angeführten Systeme/Techniken dienen nur als Beispiele).

Zusammenfassung der Anforderungen

Sowohl UIS als auch die Anwendungen sollen offene Systeme sein, die über klar definierte Schnittstellen verfügen. Es wird eine größtmögliche Unabhängigkeit von Betriebssystem, Hardware-Plattform, Benutzeroberfläche und Kommunikationswegen angestrebt. Die zu definierenden Schnittstellen müssen eine problemlose Integration weiterer DV-Komponenten ermöglichen. Die vorhandenen Datenbestände (behördliche, firmeninterne Informationssysteme) sind soweit möglich zu nutzen und in das UIS einzubeziehen. Das Gesamtsystem soll sich besonders durch eine benutzerfreundliche Dialogschnittstelle und Fehlertoleranz auszeichnen.

---

[1] engl.: Graphical User Interface, kurz GUI

4.2 Integrationsansätze

Anhand eines einführenden Beispieles soll der Umfang des Integrationsvorhabens verdeutlicht werden. Anschließend werden in einer deskriptiven Form verschiedene Integrationsmodelle beschrieben und eine erste Bewertung dieser vorgenommen.

Beispiel :

Suche nach einem geeigneten Standort für einen Kindergarten.

Ein solcher Standort läßt sich wie folgt charakterisieren :

- zentrale Lage innerhalb des Einzugsgebietes
- verkehrsberuhigte Zone
- Nähe von Grünflächen / Spielplätzen, die gut erreichbar sind
- keine derzeitige und zukünftige Belastung des Standortes durch gesundheitsschädliche Einflüsse
- kostengünstig (Nutzung von evntl. vorhandenen Gebäuden des Auftraggebers)
- Einhaltung gesetzlicher Vorschriften
- usw.

Alleine diese Kriterien machen die Komplexität der Auswahl deutlich und lassen Rückschlüsse auf die Vorgehensweise zu.
Einerseits ist topographisches und verkehrs- und städteplanerisches Wissen notwendig, genauso aber auch Kenntnisse über die Nutzung des Standortes und der Umgebung - und dies bezogen auf Vergangenheit (frühere Nutzung), Gegenwart (Grünflächen, zentrale Lage) und Zukunft (gepl. Nutzung der Umgebung).

Um einen geeigneten Standort zu finden, könnte unter Verwendung eines UIS wie folgt vorgegangen werden:
Zunächst wird das Objekt "Standort" modelliert und mit den notwendigen Merkmalen versehen, damit eine Prüfung auf die Eignung möglich ist. Anschließend werden an bestimmte Teilsysteme des UIS Anfragen gerichtet, die die Menge der in Frage kommenden Objekte bilden und die der Belegung dieser Merkmale dienen. Mit Hilfe eines Algorithmus wird nun aus dieser Konfliktmenge der geeignete Standort für den Kindergarten gewählt.

Für diese Aufgabe wären beispielsweise folgende Systeme notwendig, die in geeigneter Weise zu koppeln sind:

- topographisches, objektorientiertes System zur Verfügungstellung des raumbezogenen Wissens
- städteplanerisches Informationssystem über Verkehrsbelastung, Nutzung der Umgebung (Wohn-, Misch- oder Industriegebiet)
- Kenntnisse über vergangene Nutzung der in Frage kommenden Standorte und deren Bewertung (Altlastenbewertungssystem)
- Informationen über Lage und Art zur Verfügung stehender Gebäude, die eventuell genutzt werden könnten.
- juristisches Informationssystems, welches Auskunft gibt über Gesetze, Bestimmungen, Verordnungen, ...

Das raumbezogene Wissen spielt hierbei eine übergeordnete Rolle. Fast jede Modellierung von Umweltproblemen beinhaltet räumliches Wissen, das die örtlichen Beziehungen der einzelnen betrachteten Objekte zueinander beschreibt. Es dient des weiteren als Bindeglied für weitere Beziehungen zwischen Objekten (z.B. kann eine Beziehung zwischen Luftverschmutzungsgrad, repräsentiert durch Meßwerte, und Industriestandorten hergestellt werden). All die Informationen aus den o.g. Systemen sind für die Lösung des Problems miteinander zu verknüpfen und den einzelnen Objekten des topographischen Informationssystems zuzuordnen. Diese Zuordnung muß von dem UIS geleistet werden.

Die eigentliche Problemlösung beginnt mit der Attributierung des Objektes Standort und der Bestimmung von Anforderungen an die Attribute. Im nächsten Schritt bestimmt der Benutzer des UIS die geeigneten Anwendungssysteme, die die entsprechenden Attribute zur Verfügung stellen können, wobei er unter anderem folgende Fragestellungen zu klären hat :

- Welche Einzelsysteme können die benötigten Informationen liefern ?
- Wie sind die Informationen der einzelnen Systeme zu bewerten ?
- Wie aussagekräftig sind die gewonnen Informationen für die vorhandene Problemstellung ? Hierbei ist zu beachten :
  - Genauigkeit der Information
  - Berücksichtigung der unterschiedlichen Wissensrepräsentationen (Texte, Meßwerte (-reihen), Wahrscheinlichkeiten, Bereichsangaben, graphische Informationen, klassifizierende Informationen (hoch, mittel, niedrig), unsicheres Wissen)

Die eigentliche Suche nach einem geeigneten Standort besteht aus der Auswahl von in Frage kommenden Objekten und deren Bewertung nach den einzelnen Kriterien. Zunächst wird eine Zuordnung vorgenommen, die die zu erfüllenden Teilkriterien bestimmten Systemen zuordnet. So werden die juristischen Anfragen an ein entsprechendes Rechts-Informationssystem gerichtet werden (z.B. JURIS).

Mit Hilfe der Systeme werden systembezogene Standortmengen gebildet. Diese ergeben sich aufgrund der an die System gerichteten Anfragen. Jede einzelne dieser Mengen wird nun mit den anderen verglichen, wobei eine Gewichtung der Teilkriterien für die Auswahl beachtet wird. Bei diesem Selektionsvorgang kann es zu weiteren Anfragen an die Systeme kommen, die eventuell mit Neubildung der Standortmengen verbunden sein können.

Nehmen wir an, daß ein UIS vorhanden ist, das die Problemstellung unseres Beispieles lösen kann, jedoch nicht über eine entsprechende Komponente für die Bewertung von Altlasten verfügt. Das Kriterium "keine derzeitige und zukünftige Belastung des Standortes durch gesundheitsschädigende Einflüsse" kann also nicht herangezogen werden, um mögliche Standorte automatisch zu bewerten.

Der Nutzen der Integration einzelner Systeme in ein UIS für die Lösung dieser oder ähnlicher Aufgaben ist abhängig von

- der Schnittstellen, die die Anwendungen dem UIS zur Verfügung stellen,
- dem Leistungsumfang der Anwendungen und
- der Fähigkeit des UIS, die unterschiedlichen Anwendungen zu koppeln.

Im folgenden werden, ohne Anspruch auf Vollständigkeit, Ansätze vorgestellt, wie eine Integration vorgenommen werden kann.

4.2.1 Anwendungsorientierte Integration

Die anwendungsorientierte Integration ist die einfachste Form, verschiedene Systeme zu einem Verbundsystem zusammenzufügen. Dem Benutzer eines solchen Systemes werden die einzelnen Anwendungen auf seinem Rechnersystem verfügbar gemacht, indem die jeweilige Benutzerschnittstelle auf das UIS übertragen wird (siehe Abbildung 4.2). Zwischen den Anwendungen existiert keine Verbindung. Es besteht jedoch die Anforderung, daß sie sich dem Benutzer im Rahmen des UIS mit derselben Benutzerschnittstelle präsentieren (siehe 4.1 Anforderungen). Um dies realisieren zu können, muß sowohl das UIS als auch die Anwendung über mindestens zwei Schnittstellen verfügen :

Dialogschnittstelle und
Kommunikationsschnittstelle

Letztere dient dazu, eine Kommunikation zwischen UIS und Anwendung zu ermöglichen und erstere, um darauf aufbauend die Benutzerschnittstelle der Anwendung auf dem Zielsystem (dem UIS) zu realisieren. Des weiteren ist auf

Seiten des UIS eine Verwaltung der entsprechenden Anwendungen notwendig, um sie dem Benutzer zugänglich zu machen.

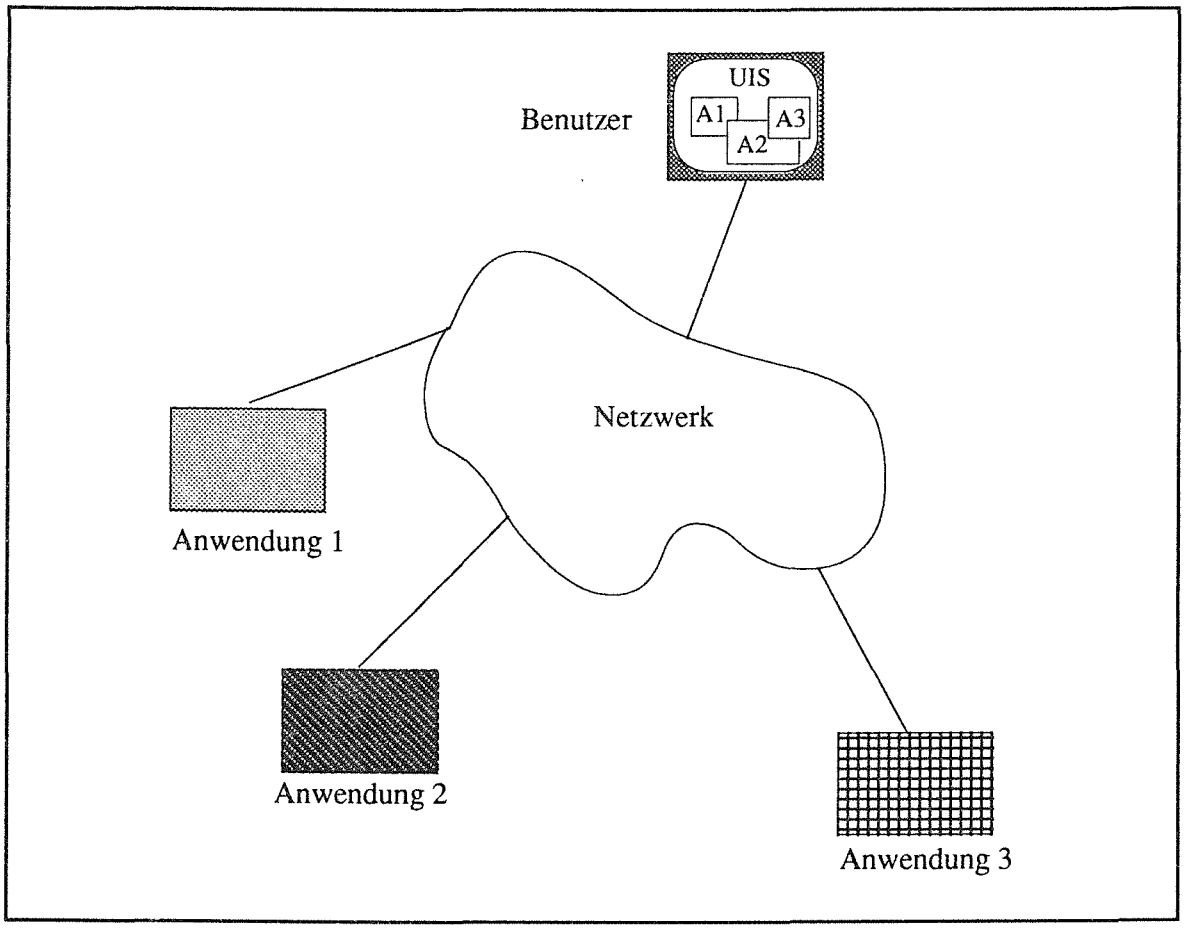


Abb. 4.2: Schema der Integration heterogener DV-Systeme in ein UIS

Diese Form der Integration läßt sich charakterisieren durch :

- Der Benutzer benötigt genaue Kenntnisse über die Mächtigkeit, den Aufbau und die Handhabung der einzelnen Anwendungen.

- Die Zuordnung von Teilproblem auf die Lösungsmethoden der einzelnen Systeme muß der Benutzer durchführen. Dies bedeutet, daß er über den Beitrag des Einzelsystems, den es zur Problemlösung beiträgt, genauestens informiert sein muß, um die Qualität der Lösung beurteilen zu können.

- Es besteht Datenredundanz. Jedes Objekt verfügt über bestimmte charakterisierende Merkmale, die in allen Systemen vorkommen, die Informationen zu diesem Objekt besitzen. Entsprechend groß ist der Aufwand bei anfallenden Änderungen. Diese müssen mehrfach durchgeführt werden,

wobei jeweils die Integritätsbedingungen der Einzelsysteme zu berücksichtigen sind. Dadurch kann eine hohe Fehleranfälligkeit entstehen.

- Redundanzen können ebenso auf Ebene der Funktionalität vorliegen. Ein Teilproblem kann von mehreren Systemen bearbeitet werden. Dabei hat der Benutzer zu entscheiden, welche Funktionen welcher Anwendung ihn bestmöglichst unterstützt.

- Positiv zu bewerten ist, daß der Realisierungsaufwand dieses Integrationsansatzes für das UIS als auch für die Anwendung relativ gering ist.

- Sowohl die Wartung als auch die Weiterentwicklung der Einzelkomponenten ist unabhängig von dem UIS. Andererseits muß der Benutzer sich über Änderungen informieren.

- Während der Problemlösung ist der Benutzer gezwungen, zwischen den einzelnen Anwendungen "hin- und herzuschalten", wobei er jedesmal eine Problemspezifikation vorzunehmen hat.

Das UIS kann den Anwender bei der Problemlösung unterstützen, indem es zu bestimmten Problemklassen Informationen anbietet, welche Teilprobleme mit welchem Einzelsystem zu lösen sind. Dieses Wissen basiert auf den Informationen, die von Seiten der Entwickler der Einzelsysteme vorliegen und den Erfahrungen des Anwenders.

Wenn man die Forderung nach einer benutzerangepaßten Dialogschnittstelle fallen läßt, so ist mittels des Dienstes "virtuelles Terminal" der Anwendungsschicht des Basis-Referenz-Modelles für die Kommunikation offener Systeme (ISO/OSI-BRMs [Tanenbaum 90]) eine noch einfachere Form der Integration möglich.
Auf diese Form der Integration wird hier nicht weiter eingegangen, da diese nur für alpha-numerische Benutzeroberflächen vorgesehen ist und der Anwender zu der funktionalen Handhabung jeder Anwendung nun auch deren spezifische Benutzeroberflächen beherrschen muß. (Alleine die unterschiedliche Belegung von Funktionstasten kann bei Anwendungen, die deren Benutzung erfordern, zu einem "Lotteriespiel" werden.)

4.2.2 Dienstorientierte Integration

In einem klassischen menügeführten DV-System stellen die einzelnen Menüpunkte der Anwendung Dienste dar, die der Benutzer in Anspruch nehmen kann. Unter einem Dienst soll in diesem Zusammenhang eine klar abzugrenzende Funktionalität einer Anwendung verstanden werden, die sich auf die Datenbasis der Anwendung bezieht und unabhängig von anderen Diensten der-

selben oder anderer Anwendungen ist. Solch ein Dienst kann z.B. die Bewertung einer Altlast oder eine Auskunft über Schadstoffbelastungen sein. Diese Dienste werden dem UIS im Verlauf der Integration des Einzelsystems bekannt gegeben und im UIS in einer Art Dienstverzeichnis abgelegt. Mit Hilfe dieses Verzeichnisses ist es möglich, eine Zuordnung von Problemstellung zu Problemlösung auf Seiten des UIS zur Verfügung zu stellen, worin der Hauptvorteil gegenüber der anwendungsorientierten Integration liegt. Der Benutzer kann sich so einen Überblick über die Mächtigkeit der einzelnen Systeme machen, ohne diese genau kennen zu müssen.

Fordert der Benutzer des UIS einen solchen Dienst an, wird diese Aufforderung vom UIS an die Anwendung weitergeleitet. Um den Dialog zwischen Benutzer und Anwendung (bzw. dem Dienst der Anwendung) zu gewährleisten, ist die Benutzerschnittstelle der Anwendung auf die des UIS abzubilden.

Dieses Integrationsmodell ist eine Spezialisierung des anwendungsorientierten Modells. Während bei dem anwendungsorientierten Ansatz das UIS nur eine Hülle für die einzelnen Anwendungen darstellt, kann es bei der dienstorientierten Integration um eigene Funktionalität erweitert werden. So können mittels des Dienstverzeichnisses Algorithmen zur Lösung von Standardproblemen realisiert werden, die sich aus den einzelnen Diensten unterschiedlicher Anwendungen zusammensetzen.

Zu den genannten Schnittstellen für Dialog und Kommunikation ist eine weitere notwendig, eine Art semantische Schnittstelle, die dem UIS mitteilt, welche Dienste die Anwendung erbringt. Diese Dienstschnittstelle kann deskriptiver Art sein; die zur Verfügung gestellten Funktionen werden textuell beschrieben und auf Seiten des UIS in dem Dienstverzeichnis zusammengeführt.

Erweitert man die Dienstschnittstelle zu einer Funktionsschnittstelle, die normierte Parameterübergaben an die Anwendungen erlaubt, so lassen sich mit diesem Modell auch verteilte Anwendungen einfacher Art realisieren.

Die Dienste sind auf den Daten bzw. den Objekten der einzelnen Anwendung definiert. Bei Aufruf einer Anwendungsfunktion bedarf es zunächst der Angabe der Objekte (Objektmenge), auf die die Funktion angewendet werden soll. Dieser Vorgang wiederholt sich bei diesem Ansatz ständig. Eine Vereinfachung ergibt sich, wenn die Anwendung einen Mechanismus zur Verfügung stellt, mit dem der Benutzer Objekte bzw. Objektmengen speichern kann, um zu einem anderen Zeitpunkt auf diese weitere Dienste anzuwenden.

Die meisten der für die anwendungsorientierte Integration aufgeführten Charakteristika lassen sich auf die dienstorientierte übertragen. Zwar hat der Benutzer nun einen Überblick über die Mächtigkeit der einzelnen Anwendun-

gen, bewegt sich aber weiterhin nach Auswahl eines Dienstes in dieser Anwendung.

Die Abhängigkeiten zwischen den Anwendungen, die sich aus der Problemstellung des Benutzers ergeben, hat dieser selbständig zu realisieren. Ändert sich die Lösungsmenge in einer Anwendung und betreffen diese Änderungen auch andere Anwendungen, so hat er diese Änderungen bei den entsprechenden Anwendungen nachzuvollziehen.

Eine wesentliche Effizienzsteigerung würde entstehen, wenn diese Objektmengen transparent für andere Anwendungen wären. Damit könnte die o.g. Dienstschnittstelle um Übergabeparameter erweitert werden, um zu spezifizieren, worauf der Dienst angewendet werden soll. Hierbei geht es nicht nur um eine einheitliche Darstellung von Objekten (Normung der Datenformate) sondern vor allem um die Bedeutung der Daten. Weiterhin ungeklärt ist, wie unterschiedliche Wissensrepräsentationsformen aufeinander abgebildet werden können.

4.2.3 Objektorientierte Integration

Das letzte vorgestellte Modell ist auch das mächtigste, bedarf allerdings eines hohen Realisierungsaufwandes auf Seiten des UIS und der einzelnen Anwendungen. Ihm zugrunde liegt das Paradigma der objektorientierten Programmierung [Cox 86]. Die einzelnen Anwendungen bieten Objektklassen und in bzw. auf ihnen definierte Methoden in Form einer Objektschnittstelle an. Diese Objekte sind in einer noch zu definierenden Syntax und Semantik abstrakt beschrieben . Das zu lösende Problem wird vom Benutzer oder dem Entwickler des UIS objektorientiert modelliert. Dabei werden die benötigten Objektklassen soweit zerlegt, daß eine Abbildung auf die Objektklassen möglich ist, die von den einzelnen Anwendungen zur Verfügung gestellt werden. Es entsteht ein Netzwerk von voneinander abhängigen Objekten.

Benötigt der Anwender eine Information von einem Objekt und ist dieses Objekt von einem anderen abhängig, so leitet es die Aufforderung an letzteres weiter. Dieser Mechanismus wird ausgeführt, bis eine Instanz einer Anwendungsobjekteklasse referenziert wird, welches die entsprechenden Daten beschaffen kann. Denselben Weg gelangt die nun ermittelte Information zurück. Dabei müssen die Methoden der einzelnen Objekte nicht auf demselben System wie das Objekt selber realisiert sein, sie können auch auf anderen Plattformen zur Verfügung gestellt werden.

In diesem Zusammenhang spielen die Informationsbeschaffung, also die Realisierung der Objektbeziehungen in einem verteilten System, als auch die Visualisierung der Informationen beim Endanwender eine große Rolle. Es müssen

demnach Schnittstellen definiert werden, die den Nachrichtenaustausch zwischen den Objekten regeln, als auch eine Beschreibungssprache entworfen werden, mit Hilfe der der Inhalt der Nachrichten eindeutig beschrieben wird.

Die Merkmale des objektorientierten Ansatzes sind :

- Flexibilität des System bezüglich Erweiterbarkeit und Integration weiterer Applikationen
- Eine einheitliche Benutzeroberfläche und Benutzerführung
- Anpassbarkeit auf die Bedürfnisse des Benutzers
- Aufwendige Modellierung für den Benutzer oder den hierfür zuständigen Systementwickler
- Alle Anwendungen müssen über eine Objektschnittstelle verfügen
- Problematische Normierung der Objektschnittstelle (s.u.)

Welches der oben vorgestellten Integrationsmodelle nun das geeignete ist, hängt im wesentlichen von den Anwendungen ab, die zu integrieren sind. Die meisten vorhandenen Applikationen im Bereich Umweltinformatik sind Insellösungen, die auf ein bestimmtes Problem hin entwickelt worden sind. Oft handelt es sich auch nur um Prototypen. Dies liegt nicht zuletzt an dem noch sehr jungen Alter dieses Forschungsgebietes. Häufig anzutreffen sind klassische DV-Systeme, die sich durch eine hierarchische Benutzeroberfläche auszeichnen. In diesem Falle ist eine Integration maximal nach dem dienstorientierten Modell möglich. Ist weiterhin eine Trennung von Oberflächen- und Anwendungsfunktionalität nicht möglich, so ist auch die anwendungsorientierte Integration schwerlich realisierbar.

4.2.4 Schnittstellenproblematik

Unabhängig von dem Integrationsansatz ist die Schnittstellenproblematik: Wie kann gewährleistet werden, daß unterschiedliche Systeme (bzw. Objekte dieser) miteinander kommunizieren können. Prinzipiell kann man zwei Vorgehensweisen unterscheiden. 1. Man entwirft für je zwei Systeme, die miteinander zu koppeln sind eine Schnittstelle. 2. Es wird eine einzige Schnittstelle entworfen, die von allen einzuhalten ist. Diese beiden Vorgehensweisen werden kurz ausgeführt :

1. Zwischen je zwei miteinander verbundenen Objekten unterschiedlicher Systeme wird ein Translator-Objekt geschaltet, was die Kommunikation zwischen diesen beiden regelt. Es hat dabei die Aufgabe, sowohl die Datenformate aufeinander anzupassen als auch die Übertragung der Daten zwischen den Systemen zu regeln. Die maximale Anzahl solcher Übersetzer-Objekte ergibt sich aus der Anzahl der unterschiedlichen Systeme und entspricht den möglichen Kombinationen je zweier Systeme. Sie beträgt demnach

$\sum_{i=1}^{n-1} i = \frac{n^2-n}{2}$, wobei n die Anzahl der zu integrierenden Systeme ist.

Der maximale Aufwand läßt sich demnach mit $O(n^2)$ abschätzen. Diese Aussage gilt nur unter der Vorraussetzung, daß sich alle Objekte einer Anwendung zu dem entsprechenden Translator-Objekt kompatibel verhalten und jedes Objekt / System mit allen anderen Objekten / Systemen direkt kommuniziert.

2. Es wird eine allgemeine Schnittstelle definiert, die alle Systeme beachten müssen. Der Aufwand reduziert sich nun auf die Anzahl der Systeme (O(n)). Die Definition einer solchen Schnittstelle ist aber ausgesprochen problematisch, da sie alle möglichen Anforderungen abdecken muß. Sie wird entsprechend komplex ausfallen. Dabei liegt der Schwerpunkt sicherlich auf der Beschreibung der Informationen. Alphanumerische Daten lassen sich relativ einfach normieren. Hierfür gibt es bereits Ansätze, so z.B. die ASN.1 - Normierung des ISO/OSI-Basis-Referenz-Modells in der Darstellungsschicht [Tanenbaum 90].
   Bei graphischen Informationen ist dies nicht mehr der Fall. Die vorhandenen Formate sind stark von dem Anwendungsgebiet, der Anwendung und nicht zuletzt von dem verwendeten Graphiksubsystem abhängig. Entsprechend groß ist die Anzahl der existierenden "Standard"-Graphikformate. Normierungsbemühungen werden derzeit von unterschiedlichen Gremien und Firmen unternommen.

Natürlich ist auch eine Kombination beider Formen möglich. So könnten beispielsweise alphanumerische Daten mittels der ASN.1 kodiert werden. Für die graphischen Informationen müßten entsprechende Konverter verwendet werden. Eine Aufwandsabschätzung ist in diesem Falle nicht möglich, da die Anzahl der verwendeten Kodierungsverfahren von graphischen Daten sehr groß ist.

4.2.5 Integrationsstufen

Ein reales integrierendes UIS wird alle der vorgestellten Integrationsformen zu realisieren haben. Systeme, die derzeit nach einem objektorientierten Ansatz entwickelt werden, als auch klassische Informationssysteme sind zu kombinieren. Dementsprechend sei hier eine Vorgehensweise vorgeschlagen, nach der ein integrierendes UIS zu entwickeln ist.
Zur Realisierung aller oben vorgestellten Integrationsansätze ist ein Basissystem notwendig, welches die Kommunikation zwischen den einzelnen Systemen sowie die Darstellung von Informationen regelt. Die entsprechenden Grundkomponenten gewährleisten zunächst die anwendungsorientierte Integration.

Dieses Basissystem wird sukzessive ausgebaut um weitere Komponenten, die die Verwaltung von Diensten und Objekten ermöglichen.

In dieser Arbeit soll nun ein Vorschlag über Umfang und Gestalt eines solchen Basissystemes vorgestellt werden.

4.3 Basissystem - Entwurf

Allen in dem Abschnitt Integrationsansätze vorgestellten Modellen erfordern ein Basissystem, das allein ausreicht, den anwendungsorientierten Ansatz zu realisieren. Es kann durch Erweiterung um zusätzliche Komponenten zur Umsetzung der mächtigeren Integrationsmodelle verwendet werden.

Das Basissystem ist fester Bestandteil des UIS und der Anwendung selbst. Es gewährleistet die Abbildung der Benutzeroberfläche der Anwendung auf die des UIS. Hierzu gehört ebenfalls die Realisierung der Kommunikation zwischen UIS und der Anwendung.

Die Anwendung bzw. das System, auf dem sie realisiert ist, dient dem UIS und seinen Benutzern als Erbringer von Dienstleistungen. Im folgenden soll daher der Begriff des Anwendungsservers (kurz Server) verwendet werden. Der Benutzer bzw. das UIS sind die Dienstleistungsempfänger und werden als Clients bezeichnet. Diese Festlegungen enthalten eine begriffliche Problematik, auf die an dieser Stelle kurz eingegangen wird.

Die etwas verwirrende Gleichsetzung von Client und UIS ergibt sich aus unterschiedlichen Betrachtungsweisen. Auf der einen Seite steht das UIS als System, welches seinem Benutzer eigene Dienste und die Integrationsleistung anbietet und andererseits nutzt es die Dienste der Anwendungen, um die ihm gestellten Aufgaben zu lösen. Desweiteren dient es der Anwendung durch das Bereitstellen von entsprechenden Ressourcen, um mit dem Benutzer zu kommunizieren.

Für die Realisierung eines integrierten Systems ist das Verhältnis von UIS und Anwendung maßgebend. Es wird deshalb im folgenden von dem UIS als Client und der Anwendung als Server gesprochen.

Die Anwendung wird als eine logische Einheit verstanden. Tatsächlich kann sie auf mehrere Systeme verteilt sein.

Für die Realisierung des anwendungsorientierten Integrationsansatzes werden lediglich zwei Schnittstellen benötigt. Eine, die die notwendige Kommunikation zwischen Client und Server regelt (Kommunikationsschnittstelle) und eine weitere, die für die Umsetzung der Benutzerschnittstelle zuständig ist (Dialogschnittstelle). Eine weitere Komponente regelt das korrekte Zusammenspiel zwischen den zu den einzelnen Schnittstellen gehörenden Schichten (Kontrollschicht). Alle weiteren Einheiten, die in späteren Entwicklungsstadien hinzukommen (Object-Management-System, Database-Interface, ...) werden an oder in die Kontrollschicht angeschlossen bzw. integriert. In der Abbildung 4.3 sind diese Einheiten bereits angedeutet.

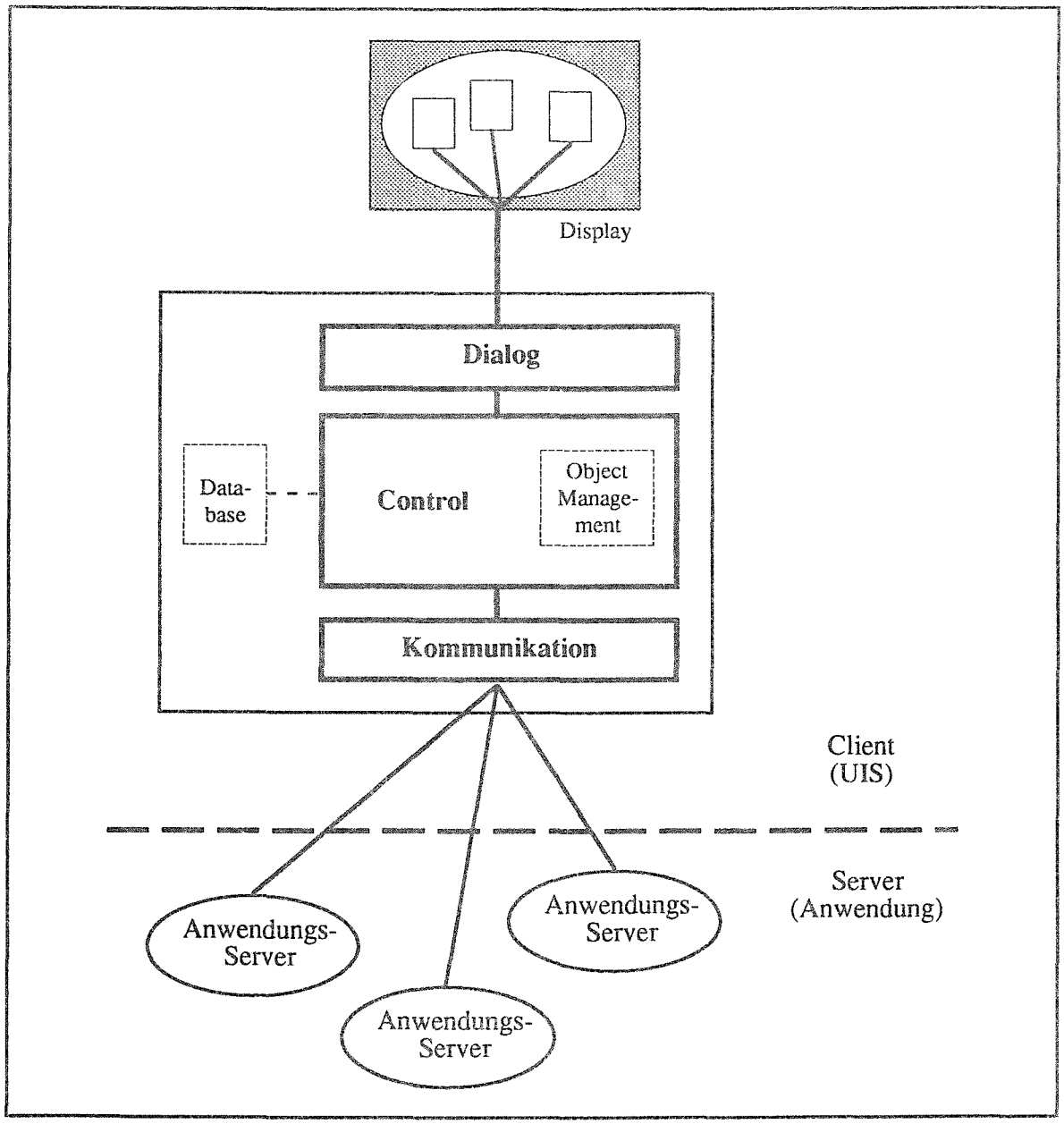


Abb. 4.3: Komponenten des Basissystems und mögliche Erweiterungen

An die Architektur der einzelnen Anwendungen werden keine Anforderungen gerichtet. Das betreffende System hat lediglich die entsprechenden Schnittstellen zur Verfügung zu stellen. Dies bietet den Vorteil, daß bereits vorhandene Systeme ohne große interne Modifikationen in ein UIS zu integrieren sind.

In den nun folgenden Abschnitten werden die einzelnen Komponenten des Basissystems vorgestellt, welches als Grundlage für die Realisierung des anwendungsorientierten Integrationsansatzes benötigt wird.

4.4 Kommunikationsschnittstelle

Die Kommunikationsschnittstelle ermöglicht die Kommunikation zwischen dem Anwendungsserver und der Kontrollschicht auf der Seite des Clients. Ihre Aufgabe ist es, die fehlertolerante, asynchrone Nachrichtenübermittlung mit Vorrangdaten und Synchronisationsmechanismen zwischen beliebigen Partnern zu gewährleisten. Auf diese Anforderungen und deren Realisierung wird im folgenden eingegangen.

Bei dem Entwurf der Schnittstelle wurde großen Wert auf die Unabhängigkeit von herstellerspezifischen Netzwerkarchitekturen gelegt; vielmehr ist eine Orientierung an dem ISO/OSI-Basis-Referenz-Modell (BRM) vorhanden. Da dieses Modell und seine Protokolle von den meisten Netzwerksystemen unterstützt wird, ist eine problemlose Umsetzung der Kommunikationsschnittstelle auf solchen Systemen möglich.

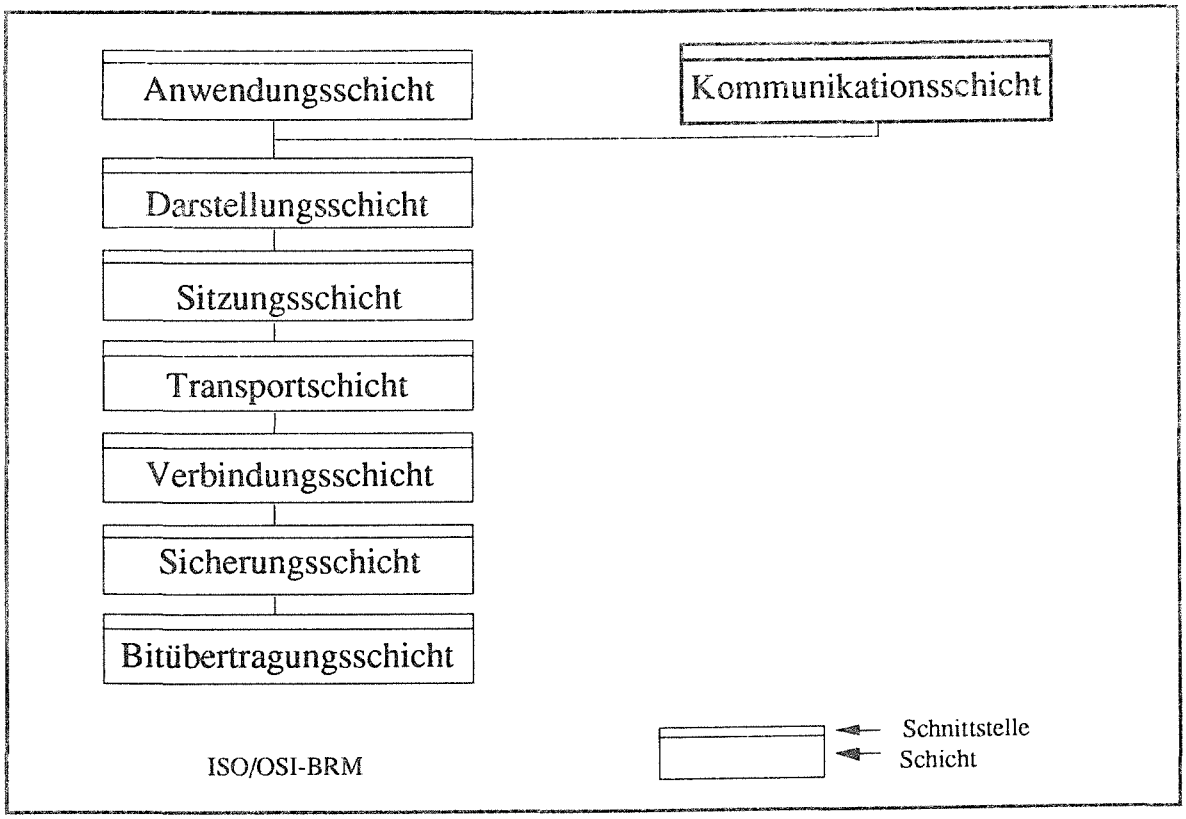


Abb. 4.4: Zuordnung der Kommunikationsschicht zum ISO/OSI-BRM

Die Kommunikationsschicht setzt, wie aus der Abbildung 4.4 ersichtlich ist, auf der Darstellungsschicht des ISO/OSI-BRM auf, auch wenn nicht der gesamte Funktionalitätsumfang dieser Schicht benötigt wird.

4.4.1 Anforderungen

Verbindungen

Die Kommunikationsschnittstelle stellt die Basis für die Integration mehrerer Anwendungen auf einem System dar, über die unter Umständen verteilte Anwendungen miteinander kommunizieren können. Es wird dabei davon ausgegangen, daß die zugrunde liegende Netzwerkstruktur heterogen ist.

Da gleichzeitig mit mehreren Anwendungen kommuniziert werden soll, ist die Verwaltung von mehreren Verbindungen zu gewährleisten. Diese Verbindungen dienen nicht nur der Kommunikation zweier örtlich getrennter Prozesse, sondern sollen auch zur Interprozeßkommunikation innerhalb eines Systems genutzt werden können. Für den Benutzer der Schnittstelle ist es nicht von Interesse, wo sich die Anwendung, mit der er kommuniziert, befindet; sie kann sich also auch auf derselben Plattform befinden. Dabei entkoppelt die Kommunikationsschicht den Benutzer von der Problematik der Protokolle, Verbindungsauf- und -abbau usw. Deren Realisierung liegt alleine im Zuständigkeitsbereich der Kommunikationsschicht und ist nach außen nicht sichtbar.

Unabhängigkeit zwischen Kommunikations- und Kontrollschicht

Da das Kommunikations- als auch die Kontrollschicht mögliche Engpässe innerhalb des Basissystemes darstellen, stark voneinander abhängen und zur gleichen Zeit unterschiedlich stark in Anspruch genommen sein können, sollen beide Komponenten so unabhängig wie möglich voneinander operieren können.

Systemunabhängigkeit

Des weiteren ist darauf zu achten, daß die Kommunikationsschnittstelle möglichst frei von Systemabhängigkeiten ist, um so einen hohen Grad an Flexibilität zu erreichen. Dies beinhaltet auch die problemlose Portierung dieser Schnittstelle auf andere Hardwareplattformen.
In diesen Sachverhalt fallen auch netzabhängige Einschränkungen und Festlegungen wie maximale Paket-/Nachrichtenlänge, Gebühren, usw.. Der Benutzer der Kommunikationsschnittstelle sollte hiervon losgelöst sein.

Fehlertoleranz

Bei auftretenden Fehlern sollte die Schnittstelle in der Lage sein, diese zu erkennen und automatisch bestimmte Mechanismen zu deren Beseitigung zu starten. Haben diese keinen Erfolg, so ist der Fehler dem Kontrollschicht zu melden.

Berechtigungsüberprüfung

Zur Überprüfung, ob ein Prozess ausreichende Berechtigung für eine Verbindung hat, müssen die entsprechenden Mechanismen zur Verfügung gestellt werden.

Transaktionsverwaltung

Mehrere logisch zusammengehörige Nachrichten sollen als solche kennzeichenbar sein. Hierzu soll das aus der Datenbanktechnik bekannte Verfahren der Transaktionen verwendet werden, wobei auf geschachtelte Transaktionen verzichtet werden kann. Somit soll ermöglicht werden, umfangreiche Nachrichten zu bündeln und Wiederaufsetzpunkte zu setzen. Abbrechen von Transaktionen ist erlaubt, um beispielsweise die Übertragung umfangreicher Datenmengen, die fälschlicherweise gestartet wurde, unterbrechen zu können, obwohl der Auftrag bereits umgesetzt wird. Dieses Prinzip läßt sich mit der Aktivitätsverwaltung der Sitzungsschicht des ISO/OSI-BRMs vergleichen [Tanenbaum 90].

Bevorzugte Nachrichten

Es sollen zwei Nachrichtentypen unterstützt werden: normale und bevorzugte Nachrichten. Bevorzugte werden so schnell wie möglich übermittelt, sie können bereits vor ihnen gesendete normale Nachrichten "überholen". Sie werden zur Übermittlung besonders wichtiger Mitteilungen an den Benutzer bzw. dessen System verwendet.

Verbindungen zwischen heterogenen Systemen

Die zu integrierenden Anwendungen sind in der Regel auf unterschiedlichen Hardware-Plattformen mit verschiedenen Zugängen zu den öffentlichen Datennetzen realisiert. Daher muß die Kommunikationsschicht entsprechende Mechanismen zur Anpassung an die entsprechenden Systeme bieten, so daß eine einwandfreie Kommunikation möglich ist.

Aus diesen Anforderungen wurde folgendes Konzept entwickelt:

4.4.2 Konzepte und Realisierung

Asynchrone, gepufferte Datenübertragung

Zwischen Anwendung und Kontrollschicht wird ein vollduplex-fähiger gepufferter Kanal, bestehend aus zwei Warteschlangen aufgebaut, der für die asynchrone Übertragung von Nachrichten zuständig ist (Abbildung 4.5).

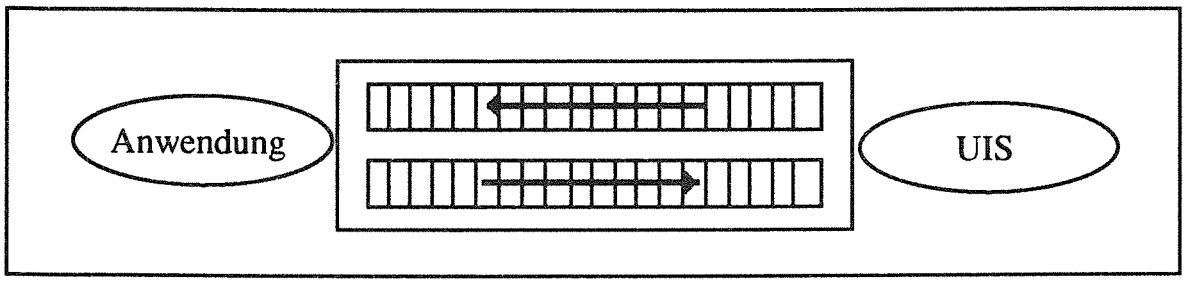


Abb. 4.5: Verbindung zwischen Anwendung und UIS

Das Kommunikationsmodul gewährleistet eine Pufferung in zweifacher Hinsicht. Einerseits wird eine interne und andererseits eine externe Pufferung vorgenommen. Die interne sorgt für eine zeitunkritische Kommunikation zwischen der Kontrollschicht und der Schnittstelle und die externe für die sichere Übertragung zwischen der "externen" Anwendung und dem Modul (siehe Abbildung 4.6). Hierdurch wird ein verzögerungsfreier Betrieb ermöglicht. Siehe hierzu auch "Nachrichtenhaltung und Weiterleitung".

Fehlertoleranz

Fehlgeschlagenes Senden von Nachrichten soll durch Wiederholen des Sendens behoben werden. Ebenso soll versucht werden, ungewollt unterbrochene Verbindungen wieder aufzubauen. Schlagen diese Sicherungsmechanismen fehl, so ist die übergeordnete Schicht von dem Abbruch der Verbindung zu unterrichten. Hierzu eignet sich das in der Sitzungsschicht des ISO/OSI-BRMs vorgeschlagene Prinzip der Synchronisation mittels sogenannter Synchronisationspunkte [Tanenbaum 90].

Zugangsberechtigung / Verbindungsaufbau

Eine Verbindung soll nicht ohne Zustimmung der Kontrollschicht des Empfängers zustande kommen. Daher wird der Verbindungsaufbau mit einem Quittungsverfahren realisiert (siehe Abbildung 4.7).

Hat sich die Kommunikationsschicht im Netzwerk angemeldet, so können beliebige Teilnehmer Verbindung zu dieser aufnehmen, sofern die System- bzw. Prozeßressourcen ausreichen. Dies ist aber nur zum Teil erwünscht. So wird durch die Kommunikationsschicht eine Benutzeridentifikation vorgesehen. Nach Verbindungsaufbau durch einen anderen Prozess, wird dies der Kontrollschicht gemeldet mit einem entsprechenden Ereignis (Request_Open). Wird von dort die Verbindung nicht gestattet, so ist sie mit einer entsprechenden Meldung abzubrechen. Andernfalls wird der erfolgreiche Verbindungsaufbau bis zur Kontrollschicht des Initiators bestätigt. Vor dieser positiven Quittung ist kein Senden von Nachrichten möglich.

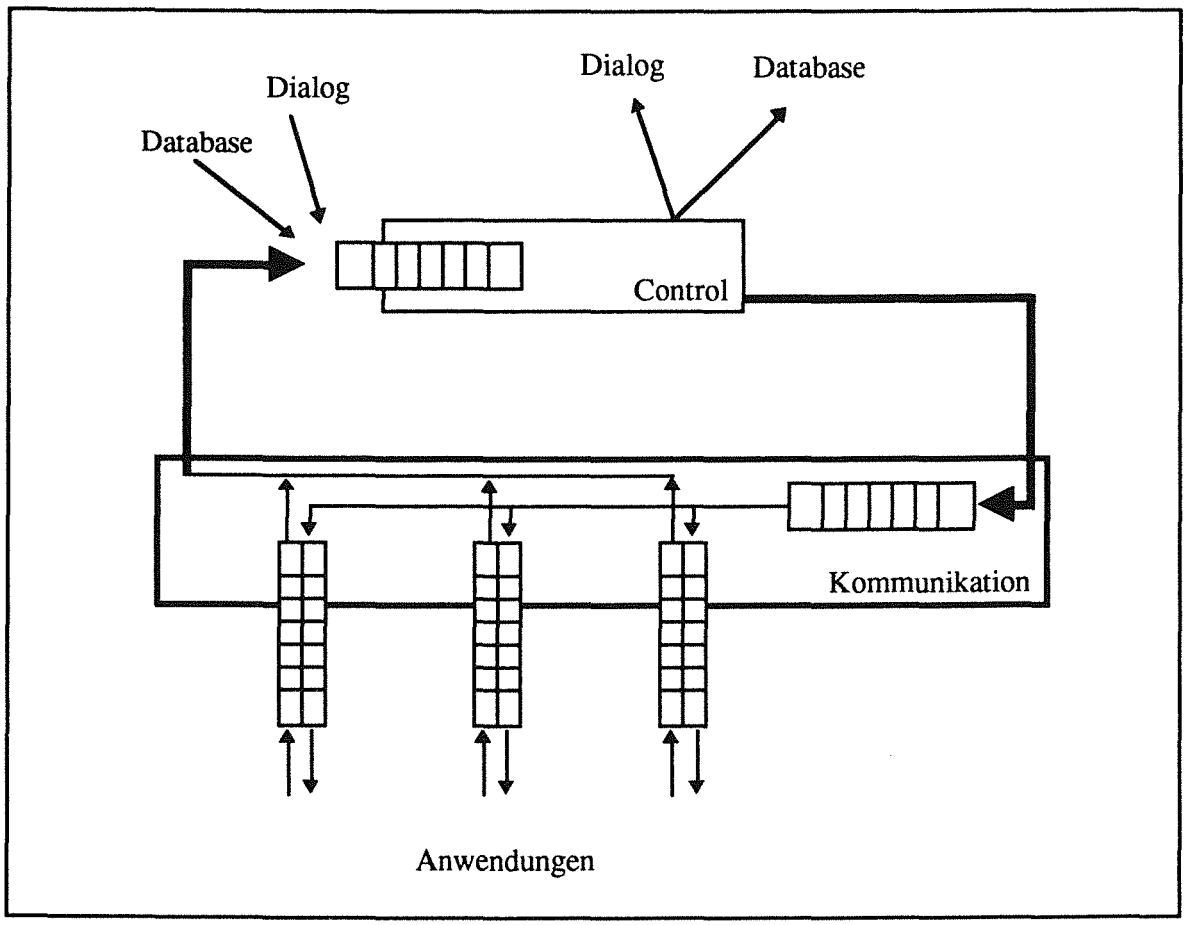


Abb. 4.6: Zusammenspiel zwischen Kommunikations- und Kontrollschicht

Mit dem Befehl zum Aufbau einer Verbindung (*Open_Connection*) wird die Verbindung zum Server angefordert und damit der Verbindungswunsch an die Kommunikationsschicht des Servers weitergeleitet. Diese teilt die Aufforderung der Kontrollschicht (Server) mittels eines Ereignisses mit (*Open_Request*). Nach deren Prüfung entscheidet die Kontrollschicht, ob die Verbindung zugelassen wird oder nicht. Die Zustimmung wird mittels *Acknowledge_Request* an die Kommunikationsschicht gegeben. Diese leitet diese Bestätigung an den die Verbindung wünschenden Partner weiter. Hiermit ist die Verbindung aufgebaut. Andernfalls wird die Ablehnung der Verbindung (*Reject_Request*) dem Partner übermittelt und danach die Verbindung abgebrochen.

Ein zurückgewiesener Request ist mit einem abrupten Abbruch der Verbindung zu vergleichen, wie er bei Netzzusammenbrüchen entstehen kann. Er unterscheidet sich jedoch in der Art, wie die Kommunikationsschicht reagieren kann. Bei einem ungewollten Abbruch kann ein Synchronisationsprozeß gestartet werden, um nach Finden eines gemeinsamen Aufsetzpunktes zwischen den

beiden Netzwerkinstanzen die Verbindung fortzusetzen. Dahingegen wird bei einer Ablehnung der Verbindung (*Request_rejected*) keine Synchronisation vorgenommen.

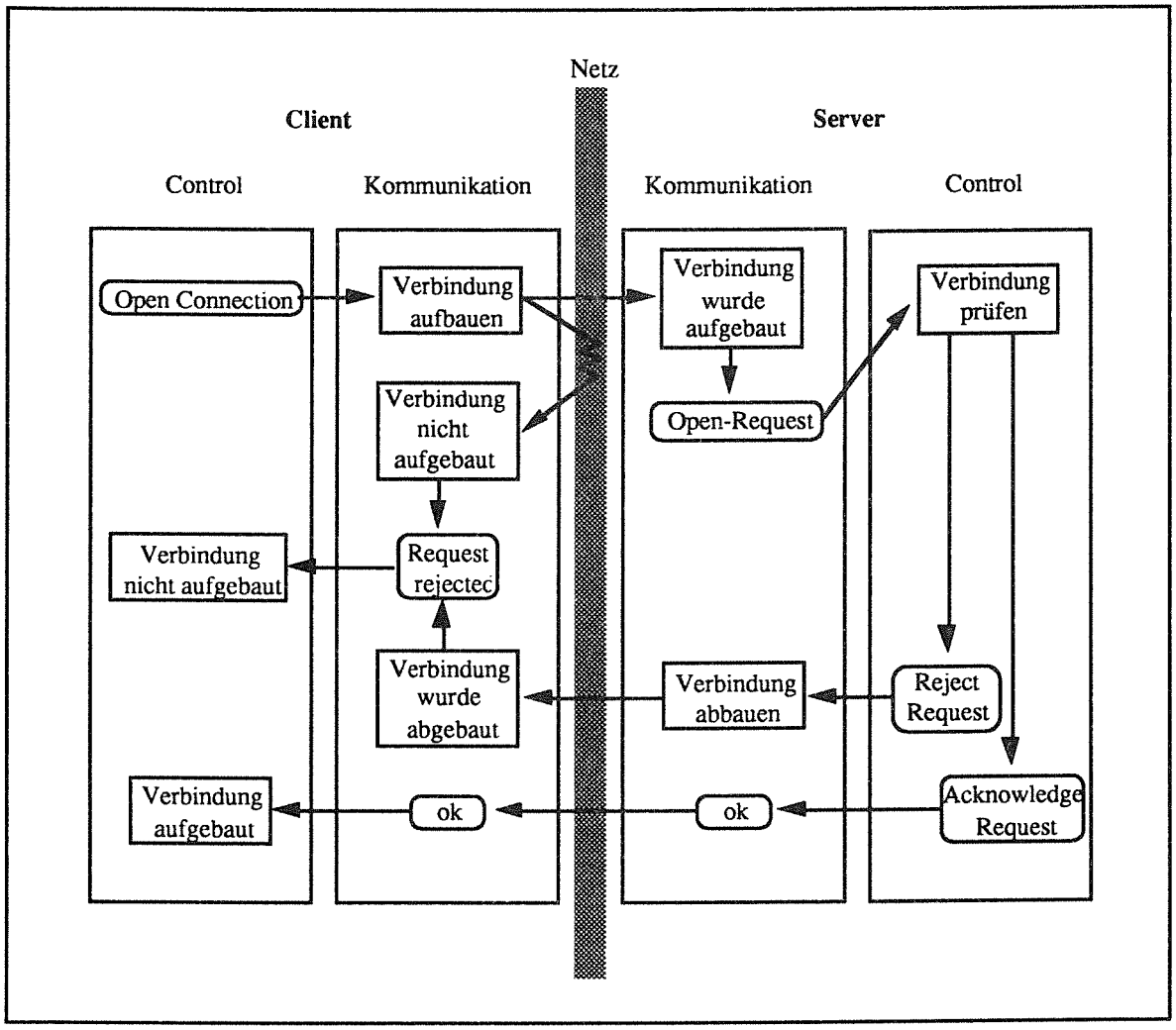


Abb. 4.7: Verbindungsaufbau

Verbindungsabbau / -rücksetzung

Das Zurücksetzen einer Verbindung kann nach einem vorhergehenden Zusammenbruch notwendig werden, um die beiden Partner neu zu synchronisieren. Der hierfür notwendige Verbindungsaufbau wird von dem Partner eingeleitet, der ursprünglich die Verbindung angefordert hatte.

Das Zurücksetzen wie auch der Abbau einer Verbindung wird von der Kontrollschicht gesteuert, da ein Verbindungsabbau in der Regel das Ende einer An-

wendung bedeutet und mit weiteren Aktionen verbunden ist, die nicht im Zuständigkeitsbereich der Kommunikationsschicht liegen.

In Abbildung 4.8 ist das Quittungsverfahren deutlich zu erkennen. Erst wenn die Gegenseite mit den Vorbereitungen zum Abbau fertig ist, eventuell noch ausstehende Nachrichten, die noch nicht an die Kommunikationsschicht übergeben wurden, verschickt hat, kann die Verbindung terminiert werden. Sind hingegen noch Unklarheiten vorhanden, so kann die Gegenseite den Wunsch zum Verbindungsabbau zurückweisen und den Dialog fortsetzen. Muß nun - aus welchen Gründen auch immer - die Verbindung trotzdem beendet werden, so kann dies mittels des Befehls zum bedingungslosen Verbindungsabbau (*Abort_Connection*) erfolgen. Das Zurückweisen einer Verbindungsabbauanforderung kann angewendet werden, wenn beispielsweise der Benutzer die Sitzung beenden will, aber ein zuvor eingeleiteter Datentransfer noch nicht beendet ist.

Transaktionen

Eine Transaktion beschränkt sich aus der Sicht der Kommunikationsschicht im Gegensatz zur Datenbankterminologie nur auf die Bündelung von Nachrichten und nicht auch auf Befehle. Letztere sind als Nachrichten codiert und der Schicht nicht zugänglich. Die eigentliche Transaktionsverwaltung obliegt den übergeordneten Schichten. So ist gewährleistet, daß einerseits komplexe Nachrichten als solche gekennzeichnet werden und sich andererseits eine anspruchsvolle Transaktionsverwaltung realisieren läßt. Hierfür besitzt die Kommunikationsschnittstelle die Befehle zum Setzen, Beenden und Abbrechen von Transaktionen. Ebenso wird die Kontrollschicht über Transaktionen mit speziellen Ereignissen in Kenntnis gesetzt (s.u.).

Ein Problem stellt lediglich die maximale Länge von Transaktionen dar. Rein theoretisch kann diese beliebig lang sein. Praktisch sind jedoch Grenzen vorhanden durch die begrenzte Kapazität für die notwendige Zwischenspeicherung der Transaktion. Diese wiederum ist abhängig von der Konfiguration der einzelnen Systeme, die die Verbindung unterhalten. Es ist also eine Synchronisation notwendig, die die maximale Transaktionslänge beim Verbindungsaufbau auf das Minimum beider Partner begrenzt.

Dies kann aber nicht nur auf die Kommunikationsschicht beschränkt bleiben, da das Setzen von Transaktionsbeginn und -ende von der übergeordneten Schicht vorgenommen wird. Das Ergebnis der Absprache über die maximale Transaktionslänge ist also dieser Schicht zugänglich zu machen.
Diese Grenze setzt sich zusammen aus den Längen der eigentlichen Nachrichten zuzüglich des Verwaltungsteiles, der jeder Nachricht von der Kommunikationsschicht angehängt wird. Beim Verbindungsaufbau werden diese beiden Werte zwischen den Partnern abgesprochen und an die jeweilige übergeordnete Schicht weitergegeben.

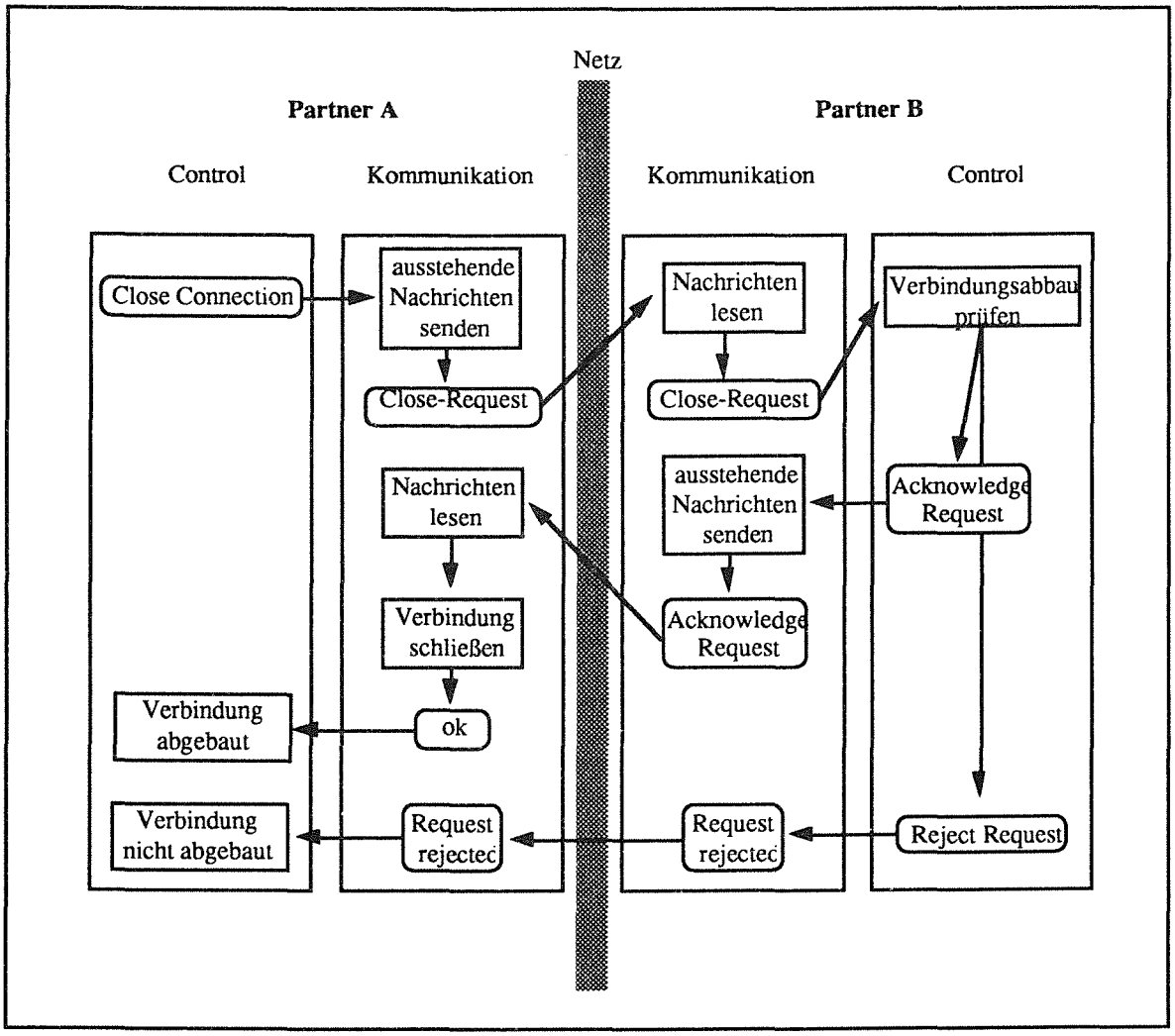


Abb. 4.8: Verbindungsabbau

Nachrichtenhaltung und Weiterleitung

Trifft eine Nachricht in der Kommunikationsschicht ein, so wird zunächst mittels eines Ereignisses deren Ankunft der Kontrollschicht angezeigt. Diese muß die Nachricht nicht sofort "abholen", d.h., daß die Nachrichten zunächst in der Kommunikationsschicht zwischengespeichert werden müssen.
Andererseits können von der Kontrollschicht mehr Nachrichten zu senden sein, als dies momentan möglich ist. Auch hier erfolgt zunächst eine Pufferung der Daten, bis diese gesendet werden können. Die Kommunikationsschicht ist also für die Speicherung aller Nachrichten verantwortlich. Die Nachrichten werden dabei in der Reihenfolge ihres Eintreffens abgelegt und in dieser auch weitergeleitet (FIFO).

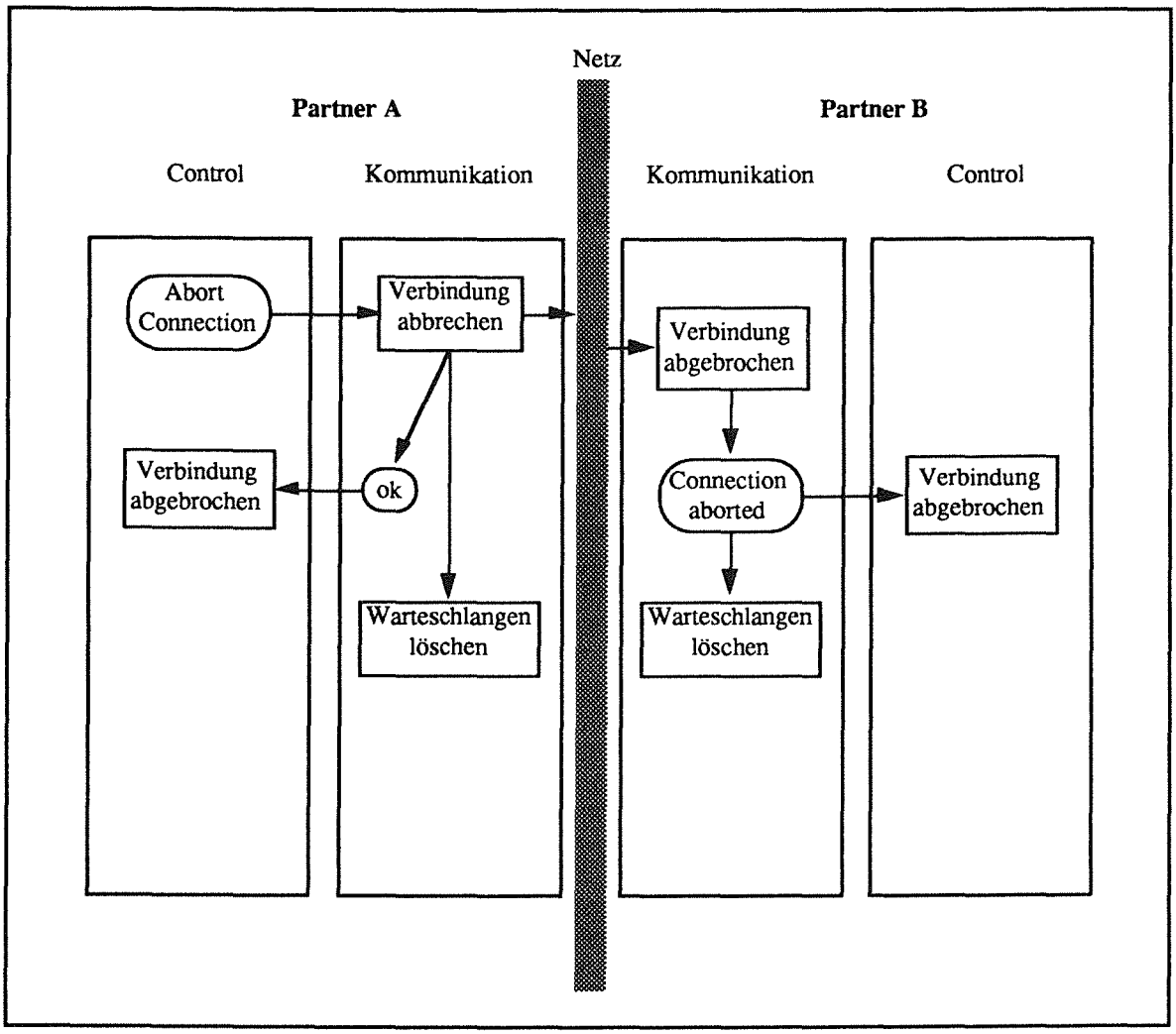


Abb. 4.9: Verbindungsabbruch

Dieses Konzept ist recht flexibel. Eine Auslagerungsstrategie läßt sich integrieren, ohne daß der Benutzer der Kommunikationsschnittstelle davon abhängig ist. Kritisch ist hier wiederum die maximale Länge einer Nachricht. Allerdings entsteht dieses Problem bereits innerhalb der Schicht beim Erhalt der Nachricht über das Netz.

Sende- und Empfangsmechanismus im Detail :

Die sendende Kontrollschicht übergibt synchron die Nachricht an die Kommunikationsschicht (KS), die dies nach Speicherung quittiert. Die Nachrichten werden innerhalb der KS in einer Warteschlange gehalten, die nach dem Prinzip FIFO (first in first out) organisiert ist. Diese Warteschlange wird Nachricht für Nachricht abgearbeitet. Kann eine Nachricht nicht gesendet werden, so wird

zunächst versucht, die Fehlerursache zu ermitteln und gegebenfalls den Sendeversuch zu wiederholen. Schlägt dies aber fehl, so wird die Verbindung abgebrochen und dies an die Kontrollschicht gemeldet. In der Abbildung 4.10 ist dies nur für den Partner A eingezeichnet. Das gleiche Verfahren findet beim Partner B Anwendung, welches hier aus Übersichtlichkeitsgründen ausgelassen wurde.

Trifft nun eine Nachricht beim Partner B ein, so speichert er diese zunächst in der Warteschlange und meldet danach seiner Kontrollschicht, daß eine Nachricht vom Partner A eingetroffen ist. Dieses wiederum wertet zunächst das Ereignis aus und ruft dann seinerseits die nun synchrone Read-Routine auf, um die Nachricht zu lesen. Dabei liest er immer die älteste, noch nicht gelesene Nachricht.

Die Ereignisübergabe erfolgt dabei mittels einer Ereigniswarteschlange, die sich zwischen Kontrollschicht und Kommunikationsschicht befindet. Diese ist in der Abbildung 4.10 nicht enthalten, jedoch aus Abbildung 4.6 ersichtlich.

Die Aufgaben der Kontrollschicht werden für wichtiger eingestuft als die der Kommunikationsschicht, die den anderen Schichten nur eine Dienstleistung zur Verfügung stellt. Aus diesem Grunde sind Anforderungen der Kontrollschicht synchron realisiert, wohingegen der umgekehrte Weg asynchron geregelt ist. Der synchrone Übergabemechanismus benötigt relativ wenig Zeit, da es sich hier lediglich um die Zwischenspeicherung bzw. Übergabe von Nachrichten handelt und dies entkoppelt von dem eigentlichen Sende- und Empfangsmechanismus ist. Anders verhält es sich mit den vorgestellten Mechanismen für den Verbindungsauf- und abbau. Der hierfür notwendige Zeitaufwand ist nicht vorherbestimmbar, weshalb diese Mechanismen asynchron realisiert werden.

Im Zusammenhang mit der Nachrichtenübermittlung wird an dieser Stelle nochmals auf die Transaktionsverwaltung eingegangen.

Wie das Transaktionsprinzip realisiert wird, ist offen gehalten. Einige Netzwerkarchitekturen (z.B. ISO/OSI-BRM, Sitzungsschicht) bieten hierfür bereits entsprechende Mechanismen an. Deshalb ist die Transaktionsverwaltung nicht Bestandteil des Protokolles. Jedoch sind einige Einschränkungen zu beachten. Transaktionen sind wie normale Nachrichten zu behandeln. Sie müssen mit diesen in der korrekten zeitlichen Reihenfolge ihres Sendens auch der Kontrollschicht des Partners angezeigt werden. Dies erfolgt über Ereignisse. Nur so ist eine eindeutige Zuordnung von Nachricht zu Transaktion möglich.

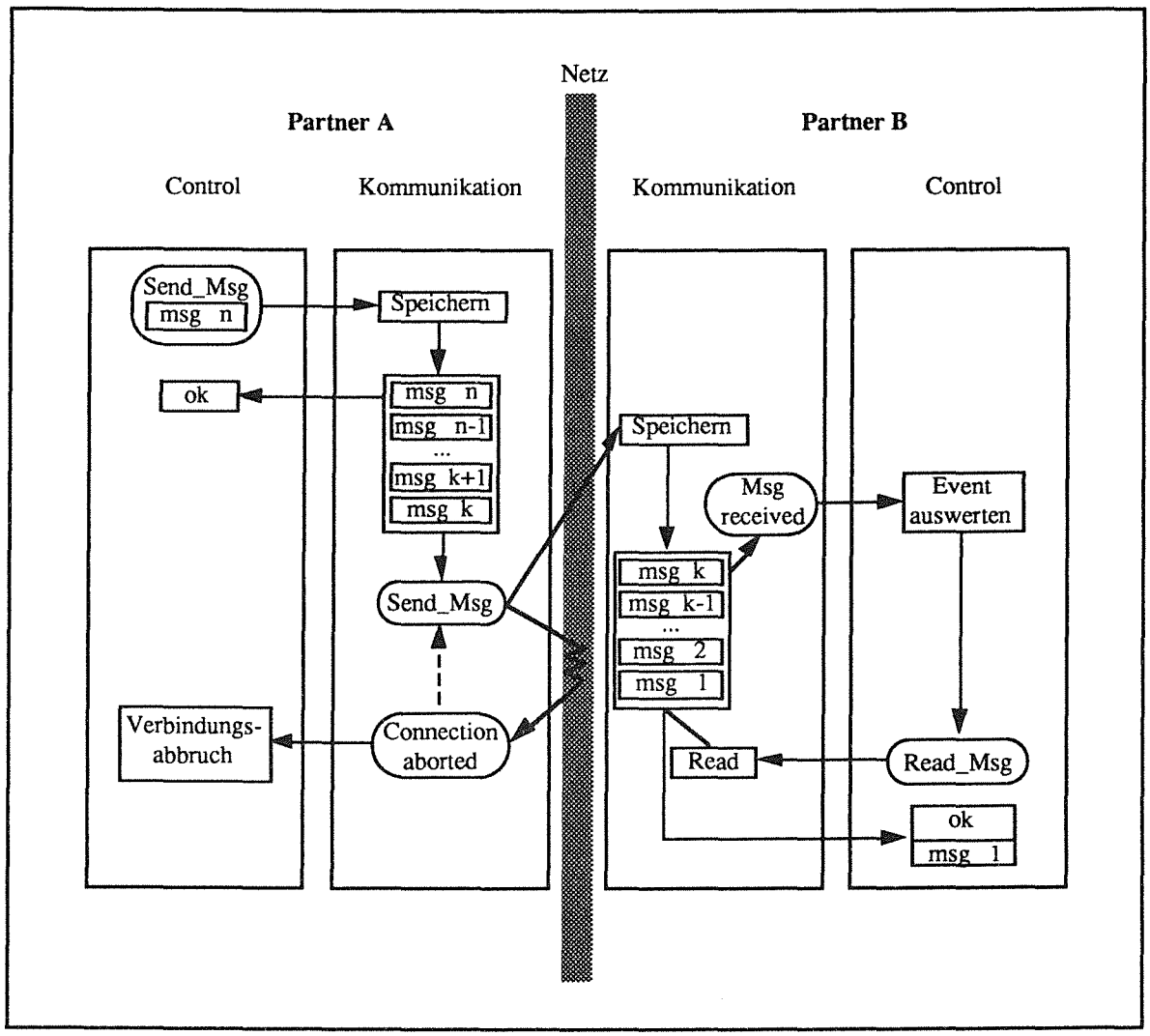


Abb. 4.10: Nachrichtenübermittlung

Bevorzugte Behandlung

Wie bereits im Abschnitt Anforderungen erwähnt, werden zwei Nachrichtentypen unterstützt. Die normalen und die bevorzugten Nachrichten. Letztere dienen vor allem der Übermittlung von dringenden Inhalten, die in besonderen Situationen notwendig werden, z.B. für die Ankündigung des Abschaltens eines Rechners während der Sitzung.
Hierfür wurden zwei Prioritäten eingeführt, die einer Nachricht zugeordnet werden. Die bevorzugten Nachrichten werden sofort an den jeweiligen Anfang einer Warteschlange eingereiht und können somit andere überholen. Trifft nun eine solche Nachricht ein, ist die normale Ankündigung über den Nachrichterhalt ausreichend (*Msg_Received*), um sie dem Empfänger als erste zugänglich

zu machen. Ein besonderes Ereignis ist also nicht notwendig. In der Abbildung 4.10 würde eine solche bevorzugte Nachricht beim Partner A zwischen Nachricht k+1 und k eingereiht werden. Nach dem Senden würde sie auf Seiten des Partners B zwischen Nachricht 2 und 1 positioniert. Hierbei wird davon ausgegangen, daß sich die Nachrichten k und 1 in Bearbeitung befinden.

Übertragungsgeschwindigkeit

Die Zeit, die bis zum Eintreffen einer Nachricht vergeht, ist durch zwei Faktoren bestimmt: Erstens die Übertragungsrate des Transfermediums und zweitens die Nachrichtenlänge. Einflüsse durch Belastung des Netzes bzw. der entsprechenden Systeme seien hier nicht betrachtet. Zur Optimierung der Übertragungsgeschwindigkeit gibt es folgende Möglichkeiten :

- Wahl eines schnellen Transfermediums
- Verwendung mehrerer virtueller Verbindungen, um eine logische Verbindung zwischen den Kontrollschichten zu realisieren (siehe Transportschicht des ISO/OSI-BRM [Tanenbaum 90]). Diese Methode muß nicht zwangsläufig zu einer Erhöhung der Übertragungsgeschwindigkeit führen, da sie mit höherem Verwaltungsaufwand verbunden ist.
- Komprimierung der zu übertragenden Daten (s.u.)

Datenkompression, -verschlüsselung und -kodierung

Auf Datenkompressionsverfahren soll in der Kommunikationsschicht verzichtet werden. Es ist Aufgabe des Netzwerkes für eine zeitunkritische und abgesicherte Übertragung zu sorgen. Dies beinhaltet auch eine eventuell notwendige Kompression und Kodierung der zu übertragenden Daten. So werden beispielsweise in der Darstellungsschicht des ISO/OSI-BRMs verschiedene Verfahren hierzu angeboten. Ebenso verhält es sich mit der Darstellung und Kodierung von Daten. Für die alpha-numerischen Informationen ließe sich die ASN.1 verwenden. Diese Notation ist aber für große komplexe Datenmengen, wie sie in der graphischen Bildverarbeitung vorkommen, nur bedingt einsetzbar.

Synchronisation / Quittungsverfahren / Fehlertoleranz

Ist einmal eine Verbindung zustande gekommen, wird von einer fehlerfreien Übertragung ausgegangen. Die einzelnen Nachrichten werden nicht quittiert. Kommt es jedoch zu Störungen (Abbruch der Verbindung), versucht die Kommunikationsschicht diese zu beheben (Wiederaufbau der Verbindung). Schlägt auch dies fehl, wird die übergeordnete Schicht von dem Verbindungsabbruch informiert.

Eine detaillierte Aufstellung aller Befehle und Ereignisse ist Kapitel 6.3.4 zu entnehmen.

4.5 Dialogschicht und -schnittstelle

In diesem Abschnitt wird die Komponente des Basissystemes vorgestellt, welche für die Benutzeroberfläche auf Seiten des Benutzers (Clients) zuständig ist: die Dialogschicht. Bevor auf den Aufbau dieser Schicht eingegangen wird, werden die Anforderungen an ihre Schnittstelle beschrieben. Hieraus werden erste Schlüsse für die Konzeption gezogen und durch bekannte Ansätze ergänzt. Darauf folgt eine abstrakte Beschreibung von Dialogen. Abschließend wird die Architektur der Schicht selber vorgestellt. Schwerpunkt bei diesen Ausführungen ist die anwendungsorientierte Integration.

4.5.1 Anforderungen

Die Dialogschnittstelle regelt die gesamte dialogorientierte Kommunikation zwischen dem Anwendungsserver und dem Client. Sie ist eine allgemeine Schnittstelle, die unabhängig von Graphik- bzw. Fenstersystem zu sein hat.

Anpassung an die vorhandene Umgebung des Clients

Die Hauptaufgabe der Dialogschnittstelle ist die Abbildung der Benutzeroberfläche der Anwendung auf die des Clients, wobei das look-and-feel des letzteren zu berücksichtigen ist (look : visuelle Ausprägung, feel : Bedienung). Da die Anwendung von verschiedenen Clients mit unterschiedlicher Systemkonfiguration genutzt werden kann, ist dies in der Abbildung entsprechend zu berücksichtigen.

Die Verbreitung von graphischen Terminals mit entsprechenden Oberflächensystemen ist verglichen mit der von alpha-numerischen relativ gering. Das bedeutet, daß ein Großteil der Anwender des UIS bzw. der Dienste der Anwendungen sich nicht auf die Möglichkeiten graphischer Visualisierungsmethoden stützen kann. Dieser Aspekt muß besonders berücksichtigt werden. Auch wenn die Benutzeroberfläche der Anwendung auf einem graphischen Fenstersystem basiert, so ist zu gewährleisten, daß diese Funktionalität dem Benutzer eines alpha-numerischen Terminals zur Verfügung steht. Hierbei muß man selbstverständlich von der Visualisierung ausschließlich graphischer Informationen absehen.

Trotz der gerade erwähnten Problematik ist besonderen Wert auf die Qualität der Benutzerschnittstelle des Clients zu legen, um den Nutzen der Anwendung für den Benutzer zu maximieren. Eine Diskussion zu diesem Thema findet man in [Bullinger 90].

Erweiterbarkeit, Wartbarkeit, Flexibilität

Es soll gewährleistet werden, daß die abzubildenden Dialoge erweiterbar sind. So können Änderungen der Anwendung unabhängig vom Client geschehen, wodurch die Wartbarkeit und die Flexibilität des Gesamtsystems gewährleistet werden soll.

Des weiteren ist die Änderung von Dialogen zur Laufzeit zu ermöglichen. Diese Anforderung ist von besonderer Bedeutung. Gerade bei Expertensystemen kommt es vor, daß sich Dialoge in Abhängigkeit vom aktuellen Zustand der Wissensbasis ändern.

Benutzerklassen

Unabhängig davon gilt es, verschiedenen Gruppen von Benutzern gerecht zu werden, die eine unterschiedliche Informationsaufbereitung benötigen oder wünschen.

Unterstützung mehrerer Anwendungen

Durch die Verfügbarkeit mehrerer Anwendungen in einem integrierenden UIS zur gleichen Zeit wird der Nutzen des Gesamtsystems deutlich erhöht. Die Dialogschnittstelle soll dies entsprechend unterstützen. Sie ist nicht für die Kopplung der Anwendungen zuständig. Dies ist Aufgabe der Kontrollschicht. Die Dialogschicht befaßt sich nur mit der Visualisierung von (Dialog-) Objekten und der Dialogsteuerung.

Offen für Erweiterung des Basissystemes

Die Dialogschnittstelle und die Architektur der Dialogschicht soll von vorne herein objektorientiert sein. Damit soll eine problemlose Erweiterung für neue Anforderungen, die durch den Ausbau des Basissystems entstehen, ermöglicht werden.

Minimierung des Kommunikationsaufwandes

In klassischen DV-Systemen ist die Dialogführung eng verzahnt mit der Anwendung selber. Aufgrund von Benutzerinteraktionen werden Anwendungsfunktionen ausgeführt, die entweder der Benutzerführung dienen oder aber anwendungsbezogene Dienste darstellen. Die Kommunikation zwischen Anwendung und Benutzer kann demnach in einen dialogbezogenen und anwendungsbezogenen Teil aufgegliedert werden. Betrachtet man den Integrationsaspekt, ergibt sich ein großer Bedarf an Netzwerk- und Rechnerkapazität. Diese Vorraussetzungen sind aus systemtechnischen und Kostengründen in der Regel nicht gegeben. Die Dialogschnittstelle und die zugehörige Dialogschicht sollen entsprechende Mechanismen zur Verfügung stellen, um diese Kommunikation

zu minimieren. Hierbei ist es das Ziel, die dialogbezogenen Aktionen auf der Seite des Clients zu realisieren und so den Kommunikationsaufwand auf die Anwendungsfunktionalität zu beschränken.

Ein Teil der anwendungsbezogenen Dialogkommunikation ist gekennzeichnet durch die Wertebereichsüberprüfung von Benutzereingaben. Auch hierfür sollen Verfahren auf Seiten des Clients angeboten werden, um eine weitere Minderung des Kommunikationsaufwandes zu bewirken.

Anbindung an die Anwendung

Aus der soeben genannten Anforderung ergibt sich der Bedarf, die Benutzerschnittstelle von deren Anwendung weitgehend abzutrennen. Die Schnittstelle soll sowohl hierfür geeignete Methoden zur Verfügung stellen, als auch für die notwendig gewordene Kopplung zwischen Dialog und Anwendung.
Die Benutzerfreundlichkeit erfordert hierfür Verfahren, die dies ohne Benutzerinteraktion realisieren. Es ist daher notwendig, daß aus dem Dialog heraus Anwendungsfunktionen angefordert werden können, ohne diesen dabei verlassen zu müssen. Andererseits soll die Anbindung auf Seiten der Anwendung keinen großen zusätzlichen Aufwand für die Server-Seite bedingen.

Einschränkung auf textuelle Informationen

In der ersten Stufe der Realisierung der Dialogschnittstelle sollen ausschließlich alpha-numerische Informationen verarbeitet werden können. Der Ausbau um graphische Visualisierungsmethoden ist aber unabdingbar, wenn die Vorteile der Integration unterschiedlicher Anwendungen in ein UIS ausgeschöpft werden sollen. Die Mächtigkeit eines Umweltinformationssystems wird nicht zuletzt an der Informationsvielfalt, der Art, wie unterschiedliche Wissensstrukturen miteinander verknüpft und deren Beziehungen dargestellt werden können, gemessen werden. Die Kopplung von geographischen Grundwissen mit textuellen Informationen spielt dabei eine große Rolle. Weiterhin läßt sich strukturiertes Wissen symbolisch verständlicher darstellen, verglichen mit der reinen textuellen Repräsentationsform. Dies gilt im speziellen für die Visualisierung von Beziehungen zwischen Objekten. Auch kann der Wissenserwerb in Expertensystemen mittels graphischer Hilfsmittel wesentlich verbessert werden bis hin zu dem Wegfall des bisher meist noch notwendigen Wissensingenieurs. Eine derartige Wissenserwerbskomponente wird z.B. in [Gappa 88,89], [Puppe 88,90] vorgestellt (CLASSIKA).

4.5.2 Konzepte

Aufgrund dieser Anforderungen gilt es nun, Konzepte für die Realisierung der Dialogschnittstelle zu entwickeln.

Aus historischen und systemtechnischen Gründen wird zwischen rechnerinitiiertem und benutzerinitiiertem Dialogablauf unterschieden. Viele der herkömmlichen Systeme sind durch eine rechnerinitiierte Dialogsteuerung gekennzeichnet. Das Anwendungsprogramm "diktiert" dem Benutzer den einzuschlagenen Ablauf oder bietet ihm eine feste Anzahl von Auswahlmöglichkeiten an. Der Vorteil dieses Konzeptes ergibt sich aus dem geringen Realisierungsaufwand auf der Anwendungsseite. Andererseits bedingen Änderungen der Anwendungsfunktionalität solche der Benutzerschnittstelle und umgekehrt.

Graphisch-interaktive Systeme dahingegen basieren auf dem benutzerinitiiertem Dialogmodell. Der Benutzer bestimmt, welche Aktion er als nächstes ausführt. Die Anwendung reagiert auf die Ereignisse, die durch eine Benutzerinteraktion ausgelöst wurden. Sie wertet die eintreffenden Ereignisse aus und führt den entsprechenden Programmteil aus. Dies hat auf den grundsätzlichen Aufbau von Programmen wesentliche Auswirkungen. Die Anwendung hat einen geringeren Einfluß auf ihre Steuerung verglichen mit den Applikationen, die durch rechnerinitiierte Dialogführung charakterisierbar sind.

Der Entwurf der Dialogschnittstelle muß beide Arten der Dialogsteuerung berücksichtigen.

1. Räumliche Trennung von Anwendung und ihrer Benutzerschnittstelle

Aufgrund der räumlichen Verteilung von Client und Server ergibt sich die Notwendigkeit, die Benutzeroberfläche von ihrer Anwendung zu trennen. Um dies zu realisieren, existieren unterschiedliche Ansätze, von denen die für die vorhandene Problemstellung relevanten vorgestellt werden.

Virtuelles Terminal

Der Dienst des „virtuellen Terminals“ der Anwendungsschicht des ISO/OSI-BRMs (siehe Kapitel Integrationsansätze) führt die Trennung von Anwendung und Dialog nur bedingt durch. Dieser Dienstes wird durch die Übertragung aller Terminalsteuerungs- und Benutzerzeichen zwischen Terminal und Anwendung über ein Netzwerk realisiert. Die Trennung wird durch das Netzwerkprotokoll ermöglicht. Der Kommunikationsaufwand bleibt gleich bzw. wird durch den Overhead des Netzwerkprotokolles erhöht. Gerade diesen aber gilt es zu minimieren. Aus diesen und den bereits genannten Gründen wird dieser Lösungsansatz den Anforderungen nicht gerecht.

X Windows und das OSF Technology Framework

Zu den Fenstersystemen, die die oben genannte benutzerinitiierte Dialogsteuerung unterstützen, gehört X-Windows (MIT). Es realisiert ein verteiltes netzwerkorientiertes Fenstersystem in einem heterogenen Netzwerk. Ihm zugrunde liegt das sogenannte Client-Server-Konzept.

Die Verwendung der Begriffe Client und Server ist dabei auf das Display bezogen. Das Display bezeichnet dabei eine Einheit aus Tastatur, Zeigegerät (Maus) und einem oder mehreren Bildschirmen. Der Server verwaltet das Display des Benutzers (dient als Server des Displays) und die Anwendung (Client), die sich innerhalb des heterogenen Netzwerkes befindet, nutzt den Server, um seine Benutzeroberfläche dem Anwender zur Verfügung zu stellen. Hierfür wird ein Window-Manager benötigt. Dieser ist ebenfalls ein Client, der eine Anwendung darstellt, die den Server verwendet (zur Realisierung des Fensterverwaltungssystems). Dieses Konzept wird in der Abbildung 4.12 dargestellt.

Führt der Benutzer eine Interaktion (Mausbewegung, Selektion, fensterbezogene Operation, Tastatureingabe) durch, so meldet dies der Server an den Client. Dieser wertet das Ereignis aus und reagiert entsprechend. Dadurch entsteht ein großer Bedarf an Netzwerkressourcen für die Kommunikation zwischen Client und Server. Weitere Informationen zu X-Windows sind unter anderem in [Scheifler 86], [Muth 91], [XWin 90] zu finden.

Ähnliche Prinzipien finden sich beispielsweise in MS-Windows, Presentation-Manager, Open Desktop, ..., nur um einige weitere zu erwähnen.

Ein auf X-Windows begrenztes Konzept der Open Software Foundation (OSF), bildet die Basis für eine Trennung von Anwendungs- und Oberflächenfunktionalität [Atlas 89] (siehe Abbildung 4.11).

Es soll an dieser Stelle nur kurz auf die einzelnen Schichten dieses Rahmenwerkes eingegangen werden.

Data Stream Encoding Layer (X-Protocol)

Realisiert die Kommunikation zwischen Server und Client mit elementaren Befehlen. Der Austausch von Nachrichten erfolgt asynchron und ohne explizite Quittierung.

Base Window System Layer (Xlib)

Diese Funktionsbibliothek stellt eine Sprachanbindung für die Programmiersprache C an das X-Protocol dar. Die Ausgabe- und Verwaltungsfunktionen ent-

sprechen im wesentlichen eins-zu-eins der durch das X-Protocol angebotenen Funktionalität. Zusätzlich werden Optimierungsverfahren und Ereigniswarteschlangen zur Verfügung gestellt.

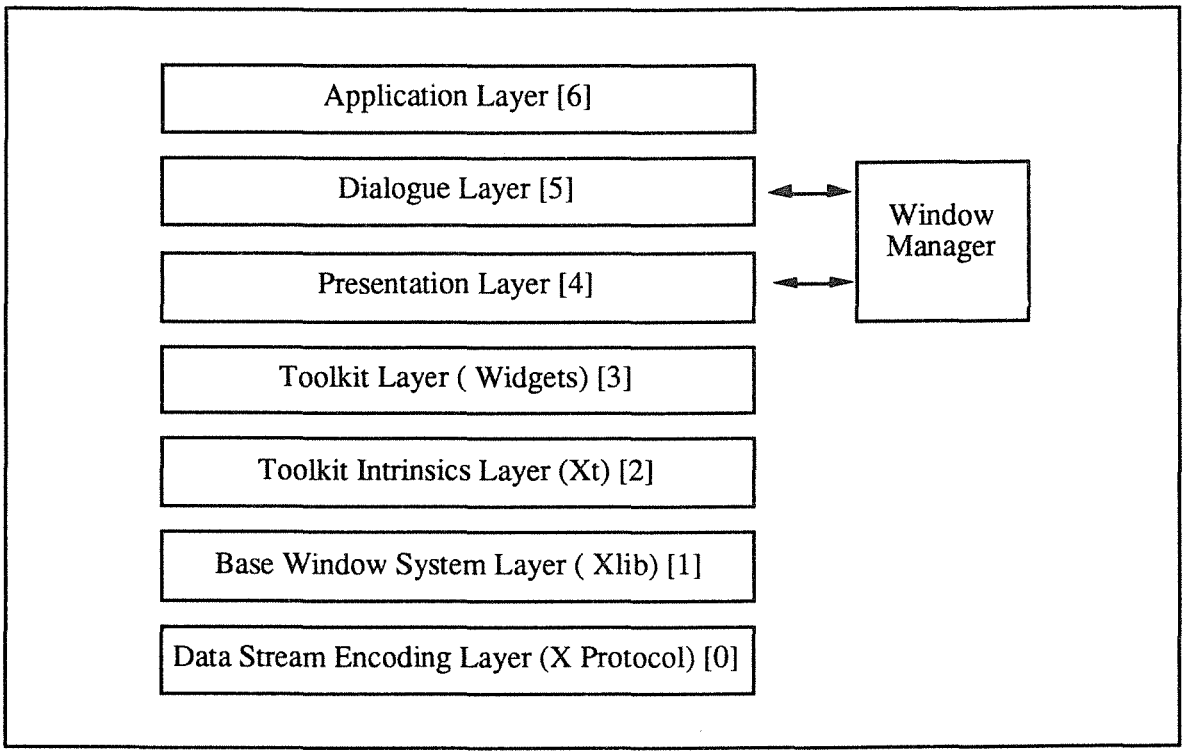


Abb. 4.11: OSF Technology Framework

Toolkit Intrinsics Layer (Xt, Xtool)

Die Xt-Bibliothek dient der vereinfachten Entwicklung von graphisch-interaktiven Anwendungen. Sie stellt die graphischen Basisobjekte (basic widgets) zur Verfügung. Widgets bestimmen das Bedienverhalten und durch ihre graphische Ausprägung auch den optischen Eindruck einer Benutzeroberfläche.

Toolkit Layer (Widgets)

Auf dieser Ebene beginnt OSF/Motif. Sie enthält eine Menge von Oberflächenobjekten, die es erlauben ein API (Application Programming Interface) zu definieren.

Presentation Layer

Diese Ebene beinhaltet die Initialisierung und Änderung von Darstellungsaspekten der Oberfläche, wie Ort, Größe, Farbe und Stil der Objekte auf dem Bildschirm, die die Oberfläche ausmachen. Das Presentation Layer stützt sich mei-

stens auf "description files", die interpretiert werden und Einzelheiten einer Oberfläche festlegen. Diese Funktionalität wird auch durch eine Funktionsschnittstelle gewährleistet. Die description files müssen jedoch zunächst compiliert werden, bevor sie mit dem Anwendungsprogramm gelinkt werden können. Aus diesem Grunde eignen sie sich nur bedingt für die Realisierung einer dynamischen Dialogsteuerung.

Dialoque Layer

Diese Schicht nimmt der Anwendung lästige Detailarbeit ab. Die Schnittstelle zwischen Application- und Dialogue-Layer ist üblicherweise auf einem abstrakten semantischen Niveau und unabhängig von Spezifika der Oberflächengestaltung.

Application Layer

Hier befindet sich die eigentliche Applikation.

Die Ebenen 0 bis 2 repräsentieren den strukturellen Aufbau von X-Windows. Das eigentliche Rahmenwerk der OSF setzt auf diesen auf und legt die Schichten 3 bis 6 fest. Nicht in der Abbildung 4.11 enthalten ist das Modul zur Benutzerschnittstellenbeschreibung (User Interface Language, UIL). Dieses ist wie der Window Manager zwischen den Schichten 4 und 5 anzusiedeln.

Das Problem des hohen Kommunikationsaufwandes zwischen Applikation (Client) und ihrer Benutzerschnittstelle (Server) bleibt mit dem OSF Technology Framework jedoch bestehen.

Ein Überblick über Werkzeuge, Konzepte und User Interface Design Systems auf der Basis von X-Windows findet man in [Rumpf 90].

2. Trennung von Anwendungs- und Oberflächenfunktionalität

In klassischen DV-Systemen ist häufig eine sehr enge Bindung zwischen Funktionalität und Benutzerschnittstelle vorhanden. Zur Visualisierung eines Sachverhaltes werden Aufrufe von (graphischen) Kernfunktionen verwendet. Eine spätere Änderung der Anwendungsoberfläche ist aufwendig, auch wenn diese keine funktionale Änderung der Anwendung selbst erfordert. Nachträgliche Modifikationen des Anwendungskerns fallen entsprechend komplexer aus und können ein Redesign von Teilen der Oberfläche notwendig machen.

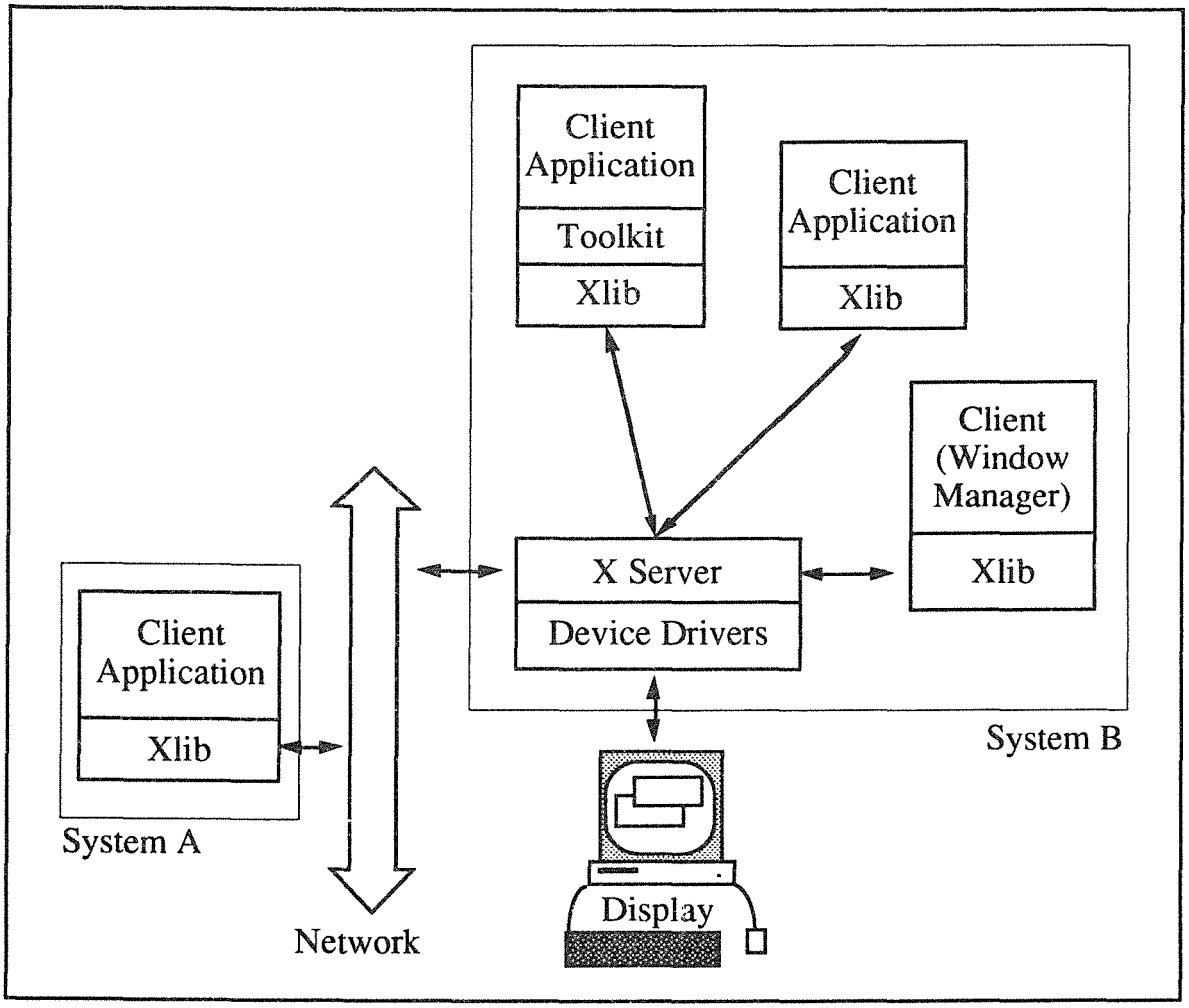


Abb. 4.12: Verteilte Applikationen unter X-Windows

User Interface Management System

Ein weiterführender Ansatz ist der der User Interface Management Systeme (UIMS). Sie realisieren die Loslösung von Anwendungs- und Oberflächenfunktionalität. Ursprünglich wurden sie zur Realisierung portabler Oberflächen und getrennter Entwicklung von Anwendung und Dialogteil entworfen. (Seeheimer Modell, [Pfaff 85], speziell : [Tanner 85], als auch [Burgstaller 89], [Cote-Munoz 89], [Green 85/1], [Green 85/2], [Grollmann 90], [Heimlich 89], [Köhler 90], [Muth 89], [Müller 89], [Raether 90], [Rumpf 90], [Shan 90], [Sibert 85], [Simões 88], [Thomas 85], [Weng-Beckmann 88])

In einer typischen interaktiven graphischen Anwendung lassen sich drei Arten von Programmcode unterscheiden :

1. Code, der die internen Datenstrukturen und Funktionen des Anwendungskerns betrifft. Dieser enhält keine Visualisierungsoperationen.

2. Code, der sich direkt mit der graphischem Input und Output beschäftigt.

3. Code, der die Verbindung von Anwendungsfunktionalität (1) und Visualisierung (2) realisiert.

Ziel beim Einsatz von UIMS ist die möglichst vollständige Trennung von interaktionsspezifischen und interaktionsunabhängigen, d.h. funktionalen Teilen einer Applikation.

In Abbildung 4.13 wird das allgemeine Vorgehen beim Entwurf einer Benutzerschnittstelle beschrieben. Das User Interface wird zunächst formal spezifiziert und in einer User Interface Definition festgehalten. Diese wird dann zur Laufzeit mittels des Modules Run-Time-Support mit der Anwendung verbunden. Dieses Prinzip ist vergleichbar mit dem der UIL in dem vorgestellten OSF Technology Framework.

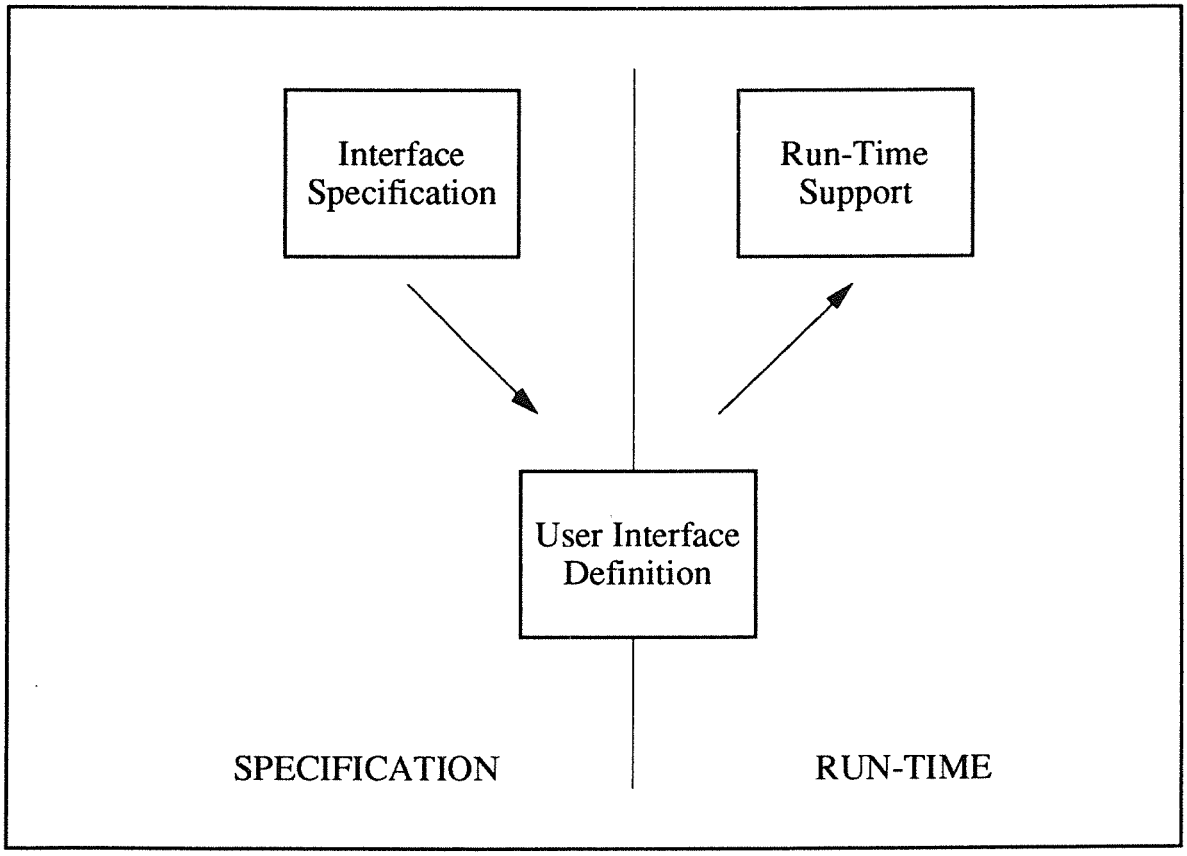


Abb. 4.13: Generelles Modell eines UIMS

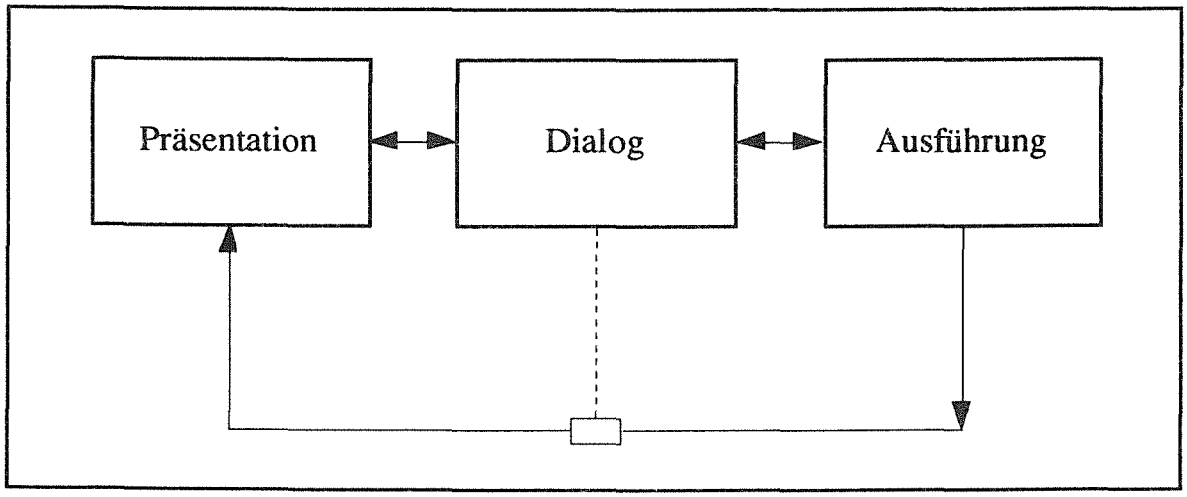


Abb. 4.14: Funktionales Modell eines UIMS

Der funktionale Ablauf geht aus den Abbildungen 4.14 und 4.15 hervor.

Die **Präsentationsschicht** ist verantwortlich für die externe Präsentation der Benutzerschnittstelle. Diese Komponente erzeugt die Dialogobjekte auf dem Bildschirm und wandelt die vom Display kommenden Benutzerinteraktionen in die interne Repräsentationsform der **Dialog-Kontroll**-Komponente (in der Abbildung kurz Dialog genannt) um. Letztere definiert und verwaltet die Strukturen, die zur Kopplung von der Anwendungsschicht (Ausführung) und Präsentationsschicht notwendig sind. Sie erhält lineare Sequenzen von Input-Merkmalen der Präsentationsschicht bzw. Output-Merkmalen der Applikation. In Abhängigkeit dieser Sequenzen bestimmt die Dialog-Kontroll-Komponente die Struktur der Interaktionen und leitet sie ihrem Ziel zu. Sie kann als Vermittler zwischen Benutzer und Anwendungsprogramm verstanden werden und steuert den eigentlichen Dialog anhand der ihr zur Verfügung stehenden abstrakten Dialogbeschreibung. Die **Applikationsschnittstelle** repräsentiert die Anwendung aus der Sicht der Benutzerschnittstelle und dient der Kopplung des UIMS mit der Anwendung.

Die darzustellenden Informationen sind für die Dialogsteuerung irrelevant und werden durch Umgehung der Dialog-Kontroll-Komponente direkt von der Anwendung an die Präsentationsschicht übergeben. Jedoch kontrolliert sie diesen Datenfluß, wandelt ihn in das entsprechende Datenformat um und stellt die notwendigen Übertragungsmechanismen zur Verfügung. Dieser Ansatz ist besonders bei der Übertragung von großen Datenmengen von der Anwendung zum Display effektiv.

In Abhängigkeit von der Ablaufsteuerung der Anwendung entstehen unterschiedliche Kontrollflüsse zwischen den einzelnen Komponenten. In Abbildung

4.14 sind diese ersichtlich. Die Ablaufsteuerung kann entweder innerhalb der Anwendung selbst oder außerhalb im UIMS erfolgen. Eine dritte Möglichkeit ergibt sich durch die Kombination der bereits genannten. In diesem Fall hat der Anwendungsentwickler situationsbedingt zu entscheiden, welches der beiden Modelle er verwendet.

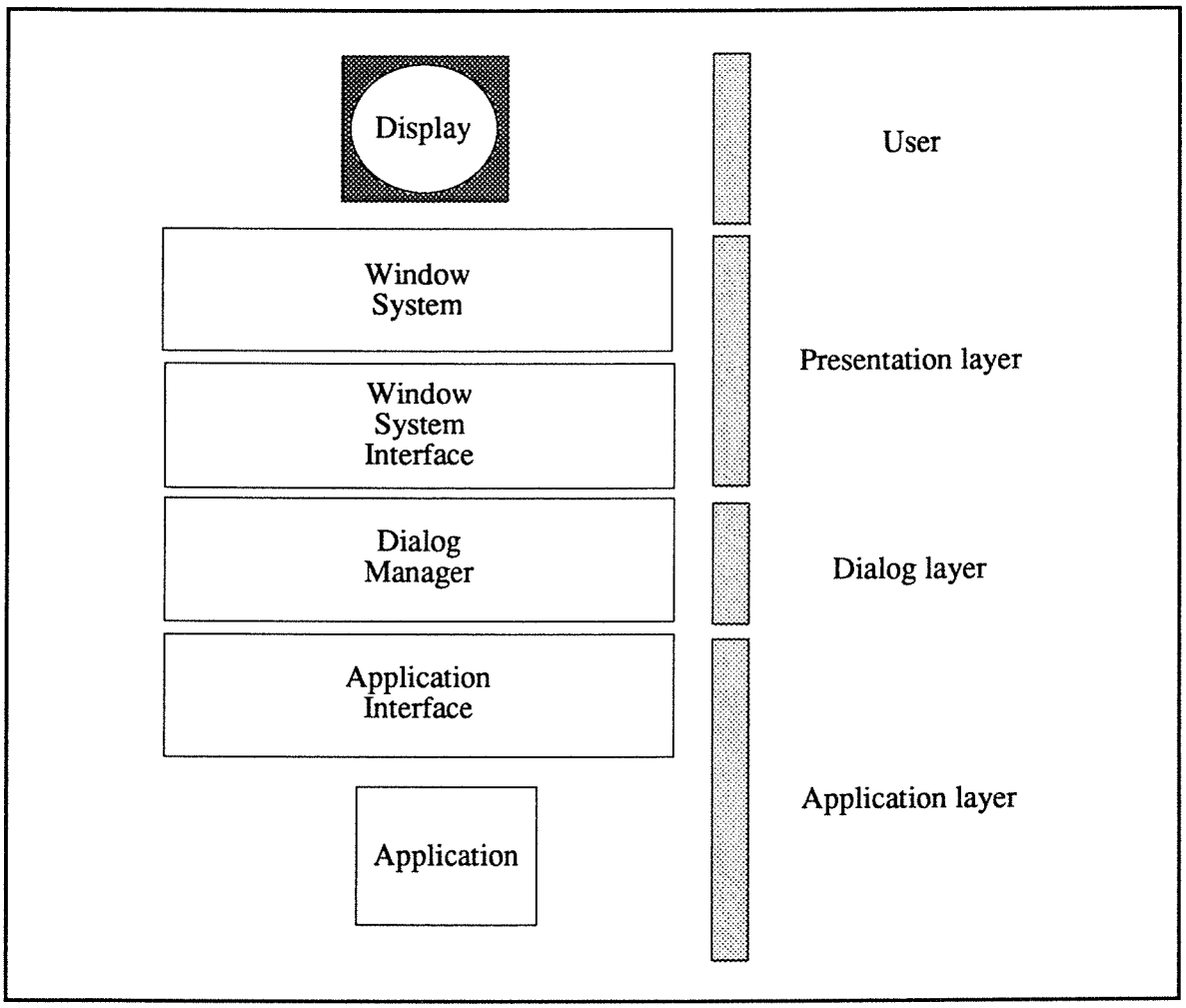


Abb. 4.15: Architektur eines User Interface Management Systems

Die Trennung zwischen Anwendungs- und Benutzeroberflächenfunktionalität ist in diesem Modell umgesetzt. In Abhängigkeit von den Fähigkeiten des Dialogmanagers kann nun auch eine Trennung der dialogorientierten Steuerung von der eigentlichen Anwendungssteuerung vollzogen werden.

Das Model-View-Controller Modell

(The Model-View-Controller Triad, [Wisskirchen 90])

Dem von Wisskirchen vorgestellten Modell eines UIMS liegt ein objektorientierter Ansatz zur Trennung der Benutzerschnittstelle von der Anwendung zugrunde, mit dessen Hilfe ...

- Anwendungskomponenten und Display-Komponenten mittels zweier getrennter Objektstrukturen realisiert werden und
- die Anwendung ohne jeden Bezug zu deren Visualisierungskomponente entwickelt werden kann.

Diese Trennung wird in einem objektorientierten System wie folgt umgesetzt:

- Die Kommunikation zwischen Applikation und Display erfolgt mittels message-passing.
- Es erfolgt eine externe Behandlung von (graphischem) Input und Output. Alle Ein- und Ausgaben, die die Anwendung nicht direkt betreffen, werden außerhalb (extern) von ihr vollzogen.

Die grundsätzliche Idee, auf der dieses Konzept basiert, ist, daß die Anwendungsobjekte nicht direkt Nachrichten bzw. Befehle an das graphische Kernsystem senden, sondern stattdessen Display-Objekte mittels message-passing dazu auffordern. Letztere setzen diese mittels der ihnen inhärenten Methoden um (Output). Nach demselben Prinzip erfolgt der Input. Das Display-Objekt, welches durch eine Benutzerinteraktion referenziert wird, sendet die Benutzereingabe mittels einer Nachricht an das Anwendungsobjekt.

Dieses Konzept weicht von der klassischen Programmiermethodik ab und impliziert eine benutzerinitiierten Dialogsteuerung (s.o.).

Umgesetzt wird der Model-View-Controller Ansatz (MVC) durch die Konstruktion eines Objektes mit Hilfe eines Dreiecks, bestehend aus

- einem Anwendungsobjekt, genannt *Model*
- einem Repräsentationsobjekt, genannt *View* und
- einem Objekt, das für den Input zuständig ist, genannt *Controller*.

Das **Model**-Objekt enthält die für die Anwendung relevanten Datenstrukturen und auf ihnen definierte Methoden. Es enthält keinen direkten Bezug zur Benutzerschnittstelle.

Für die Dialogsteuerung ist das **Controller**-Objekt zuständig. Es wertet die Benutzerinteraktionen aus und sendet sie entsprechend ihrer Semantik an das Model-Objekt oder das View-Objekt.
Die benutzerorientierte Darstellung von Informationen obliegt dem **View**-Objekt. Es hat in dem MVC eine besondere Stellung. Einerseits ist es für die Erzeugung (Instanzenbildung) des "Dreieckes" Model-View-Controller zuständig. Andererseits stellt es die optische Schnittstelle zum Benutzer dar und ist somit Ausgangspunkt aller Benutzerinteraktionen.

Zwischen Model-, View- und Controller-Objekten bestehen jeweils zwei Kommunikationspfade (Abbildung 4.16) :

- vom Model zum View (1) : Zustandsänderungen des Model bedingen eine Änderung seiner Repräsentation (View).

- vom View zum Model (2) : Das View-Objekt benötigt Informationen, die den Aufbau des Display betreffen.

- vom Controller zum View (3) : Aufgrund einer Benutzereingabe bedarf es einer Zustandsänderung des View, die nicht das Model betreffen (Scrolling innerhalb eines Fensters).

- vom View zum Controller (4) : Das View-Objekt kann eine Benutzereingabe anfordern.

- vom Controller zum Model (5) : Aufgrund einer Benutzereingabe ist die Änderung des Model notwendig.

- vom Model zum Controller (6) : Durch eine Model-Änderung kann die Struktur und das Verhalten des Controller-Objektes beeinflußt werden. (Veränderung eines Menübalkens)

Dieses Konzept unterscheidet sich in einem wesentlichen Punkt vom klassischen UIMS. Die dialogorientierte Ablaufsteuerung wird nicht an die Anwendung weitergereicht. Es werden nur die Nachrichten ausgetauscht, die die eigentliche Funktionalität betreffen.

Wendet man das MVC-Modell auf die Integrationsproblematik an, so sind die Model-Objekte dem Anwendungsserver und die View-Objekte als auch die Controller-Objekte dem Client zuzuordnen. Die Trennung von Anwendungs- und Oberflächenfunktionalität ist deutlich zu erkennen. Der entstehende Kommunikationsaufwand ist geringer im Vergleich zum klassischen UIMS (der dialogorientierte Anteil entfällt). Da aber nicht alle Anwendungen über eine benutzerinitiierte Dialogsteuerung verfügen, wie sie für das MVC-Modell not-

wendig ist, kann das message-passing der View-Objekte an ihre Model-Objekte nicht oder nur unzureichend realisiert werden.

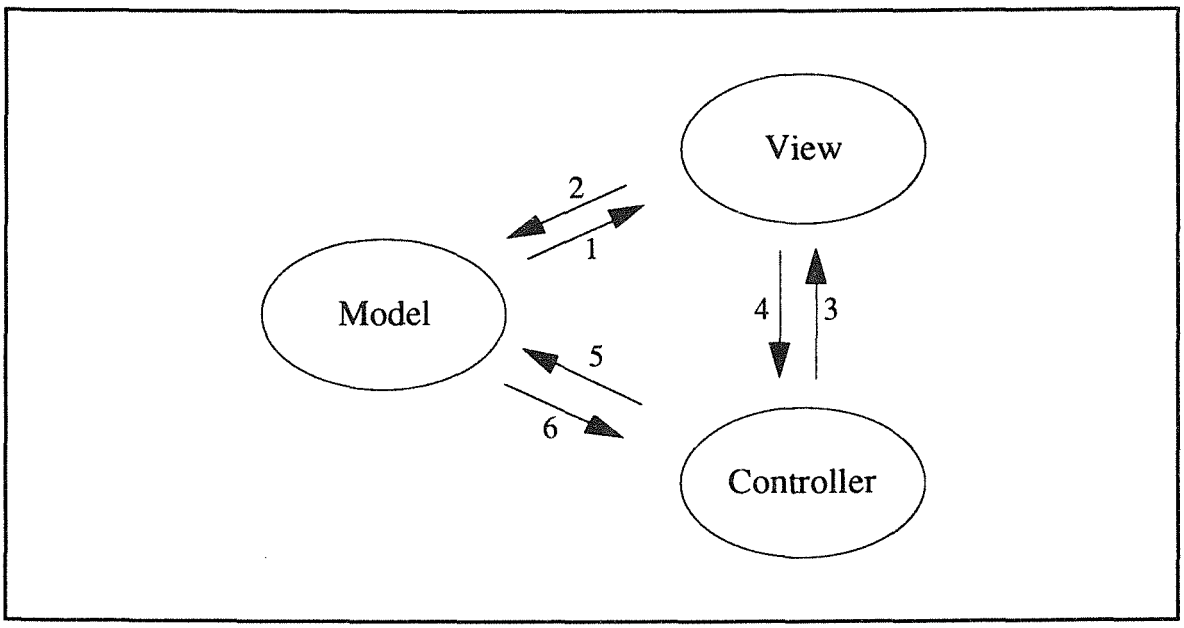


Abb. 4.16: Kommunikationspfade im MVC

Für die Konzeption der Dialogschnittstelle des Basissystems wird deshalb eine Kombination aus klassischem UIMS und dem Model-View-Controller Modell vorgeschlagen, auf das in den folgenden Abschnitten eingegangen wird.

Somit verbleibt von den Anforderungen nur noch eine : Wie können dynamisch verändernde Dialoge behandelt werden ?

3. Verwendung von dynamischen oder statischen Dialogen

In vielen Anwendungen trifft man die Situation an, daß die Dialoge als sogenannte Ressourcen des Programmes getrennt entwickelt und abgelegt werden. Die Vorteile sind offensichtlich. Mühsame Anpassungen entfallen und die Wartbarkeit wird deutlich erhöht. Diese vordefinierten Dialoge sind statisch, d.h. nur in geringem Maße zur Laufzeit manipulierbar. Die Verwendung solcher Ressourcen hat einen weiteren Nachteil : Sie sind gebunden an das zugrundeliegende Fenstersystem.

Übertragen auf die Integrationsproblematik hieße dies, daß diese Ressourcen bereits beim Client vorhanden sein müßten bzw. diesem bei Bedarf zukommen zu lassen sind. Diese Ressourcen wiederum müssen in einem normierten Daten-

format vorliegen, um sie auf beliebigen Client-Plattformen verwenden zu können.
Mittels der User Interface Language (UIL), wie sie in OSF/Motif (siehe OSF Technology Framework) Anwendung findet, wird eine Lösungsmöglichkeit angeboten. Die UIL ist an die Verwendung von X-Windows gebunden und ist, wie bereits gesagt, eher dem statischen Ansatz zuzuordnen.

Statische Dialoge sind allerdings in dem betrachteten Anwendungsgebiet in der Regel nicht ausreichend. Viele der vorhandenen Umwelt-DV-Systeme befinden sich noch im Entwicklungsstadium (Prototypentwicklung, Pilotanwendungen), was Änderungen der Anwendung beinhalten kann. Daher ist die Anforderung nach Flexibilität von großer Bedeutung.

Aus diesen Gründen sollen hauptsächlich dynamische Dialoge verwendet werden, die zunächst beim Anwendungsserver erzeugt werden und daraufhin dem Client bzw. seinem Basissystem übermittelt werden, der aufgrund dieser Informationen die Oberfläche generiert und verwaltet. Ein weiterer Aspekt, der für dieses Vorgehen spricht, ist die Integration von Expertensystemen [Collet 90]. Aufgrund von Zustandsänderungen der Wissensbasis zur Laufzeit der Anwendung können Dialoge in ihrem Inhalt und Ablauf variieren. Auch die Anforderung unterschiedliche Benutzergruppen mit ihnen gerecht werdenden Dialogen zu versorgen, spricht für den dynamischen Ansatz.

In dem folgenden Abschnitt wird der Aufbau der Dialogschicht beschrieben. Ihm folgt der Entwurf einer Dialogbeschreibungssprache, mit Hilfe der die Schnittstelle zwischen Anwendung und ihrer Benutzerschnittstelle realisiert wird.

4.5.4 Aufbau der Dialogschicht

1. Funktionales Modell

Das funktionale Modell gleicht dem des klassischen UIMS, wie es bereits beschrieben wurde. Der Unterschied liegt in der Verwendung einer Dialogbeschreibungssprache, mit der die Dialogspezifikation zur Laufzeit geschieht. Des weiteren ist in diesem Ansatz die vollständige Trennung von Anwendungs- und Oberflächenfunktionalität enthalten (siehe Abbildung 4.17).

Die Dialogschicht und deren Schnittstelle (Dialogschnittstelle) sind objektorientiert aufgebaut. Mit Hilfe von dialogorientierten Objektklassen kann die Anwendung Dialogobjekte bilden, die die abstrakte, allgemeine, von der Visualisierung unabhängige Benutzerschnittstelle darstellen. Die auf ihnen definierten Methoden enthalten bereits eine objektbezogene Dialogsteuerung. Beispiels-

weise wird durch das Selektieren eines Menübalkeneintrages durch den Benutzer das entsprechende Menü angezeigt. Das Dialogobjekt enhält eine Kopie der Daten des Anwendungsobjektes und wird durch die Präsentationsschicht (Presentation layer, s.u.) auf die Objekte des zugrundeliegenden Fenstersystems abgebildet (Präsentationsobjekte). Somit findet die dialogorientierte Ablaufsteuerung zwischen den Dialogobjekten und Präsentationsobjekten statt. Benötigt das Präsentationsobjekt beispielsweise durch die Aufforderung des Benutzers zum Scrollen in einem Window weitere Informationen, so werden diese vom Dialogobjekt zur Verfügung gestellt. Derartige Informationen können Dialogobjekte oder bzw. und deren Inhalte betreffen. Die Kommunikation zwischen Anwendungs- und Dialogobjekt beschränkt sich somit auf die eigentliche Anwendungsfunktionalität.

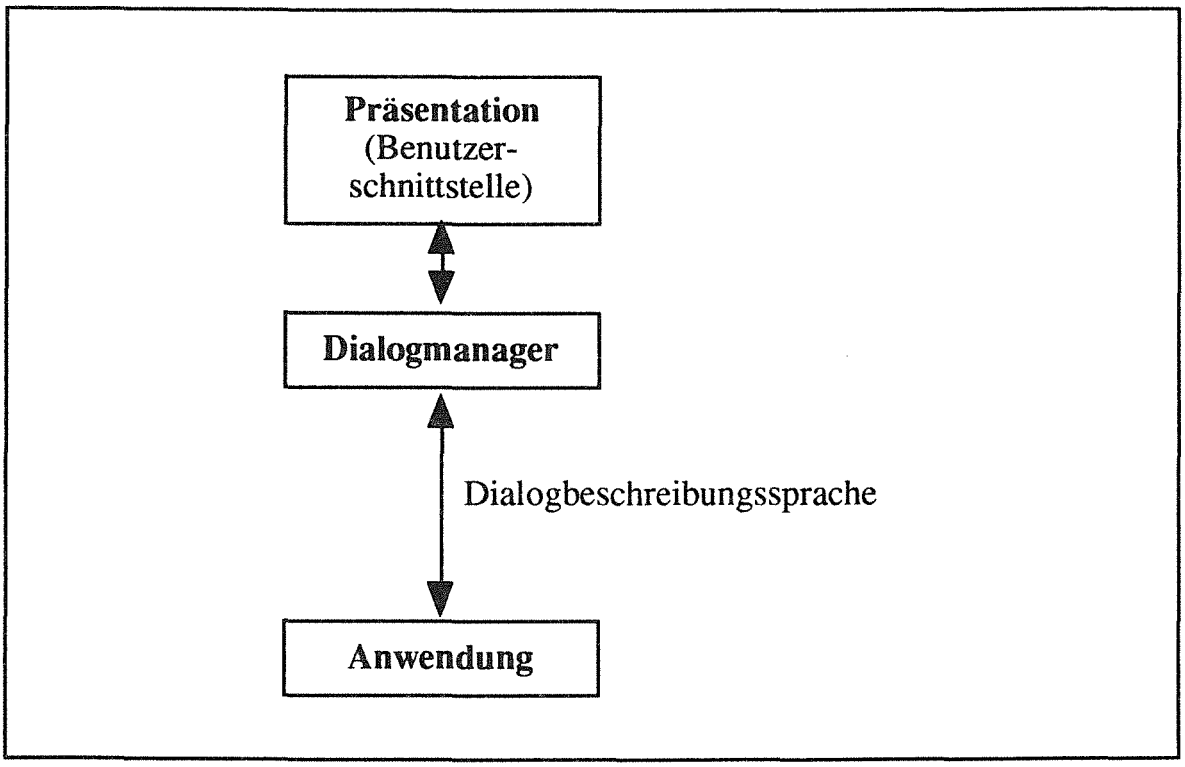


Abb. 4.17: Funktionales Modell

Die Anwendung selbst entscheidet über die Art der Verbindung zu ihren Dialogobjekten. Sowohl die benutzerinitiierte als auch die dialoginitiierte Ablaufsteuerung werden unterstützt. Mit Hilfe einer Dialogbeschreibungssprache können einzelne Dialogbjekte aber auch komplette Dialoge (Formulare, Menüs,...) generiert und der Datenaustauch zwischen Anwendungs- und Dialogobjekt (message-passing) vollzogen werden. Die hierbei versendeten Nachrichten können sowohl Informationen zu einem einzelnen Dialogobjekt (objektorientierter Ansatz) als auch einen Teil des gesamten Dialogs (anwendungsorientierter bzw.

dienstorientierter Ansatz) beinhalten. Näheres entnehme man dem Abschnitt 4.6 Dialogbeschreibungssprache.

2. Architektur

Die Dialogschicht hat die Aufgabe, Dialogklassen und auf ihnen definierte Methoden zur Verfügung zu stellen, damit einerseits die Umsetzung der Dialogbeschreibung erfolgen und andererseits die eigentliche Dialogdynamik realisiert werden kann.

Der Aufbau und die Position innerhalb des Basissystems ist aus der Abbildung 4.18 ersichtlich.

Es liegt nicht in der Zuständigkeit der Dialogschicht, die Transformation der abstrakt beschriebenen Dialogobjekte auf das aktuelle Fenstersystem vorzunehmen; dies wird durch die Präsentationsschicht gewährleistet.

Präsentationsschicht

Ihre Aufgabe ist die Abbildung des abstrakten Dialoges auf das aktuelle Window Management System. Dies beinhaltet einerseits die Abbildung der abstrakt beschriebenen Objekte auf die des zugrunde liegenden Fenstersystems, andererseits bedarf es einer entsprechenden Anpassung der Dialogsteuerung. Dabei ist eine Transformation der Ereignisse des Fenstersystems (ausgelöst durch Interaktionen des Benutzers) vorzunehmen, so daß sie von dem Dialog-Control-Modul ausgewertet werden können. Gleiches gilt für die umgekehrte Richtung.

Präsentationsschicht und Dialogschicht sind voneinander getrennt. Die Realisierung dieses Konzeptes erlaubt die Portierung des Dialogmanagers auf unterschiedliche Fenstersysteme. Der Portierungsaufwand beschränkt sich im wesentlichen auf die Anpassung der Präsentationsschicht an das entsprechende System.

Da die meisten Fensterverwaltungssysteme objektorientiert strukturiert sind und zum Teil die in der Dialogbeschreibungssprache genannten Objekte enthalten, ist diese Anpassung durch Klassentransformation und Vererbung realisierbar.

Control Interface

Das Control-Interface ist die Schnittstelle zur Kontrollschicht des Basissystems. Je nach Integrationsansatz ist diese entsprechend zu erweitern (siehe Beschreibung der Kontrollschicht).

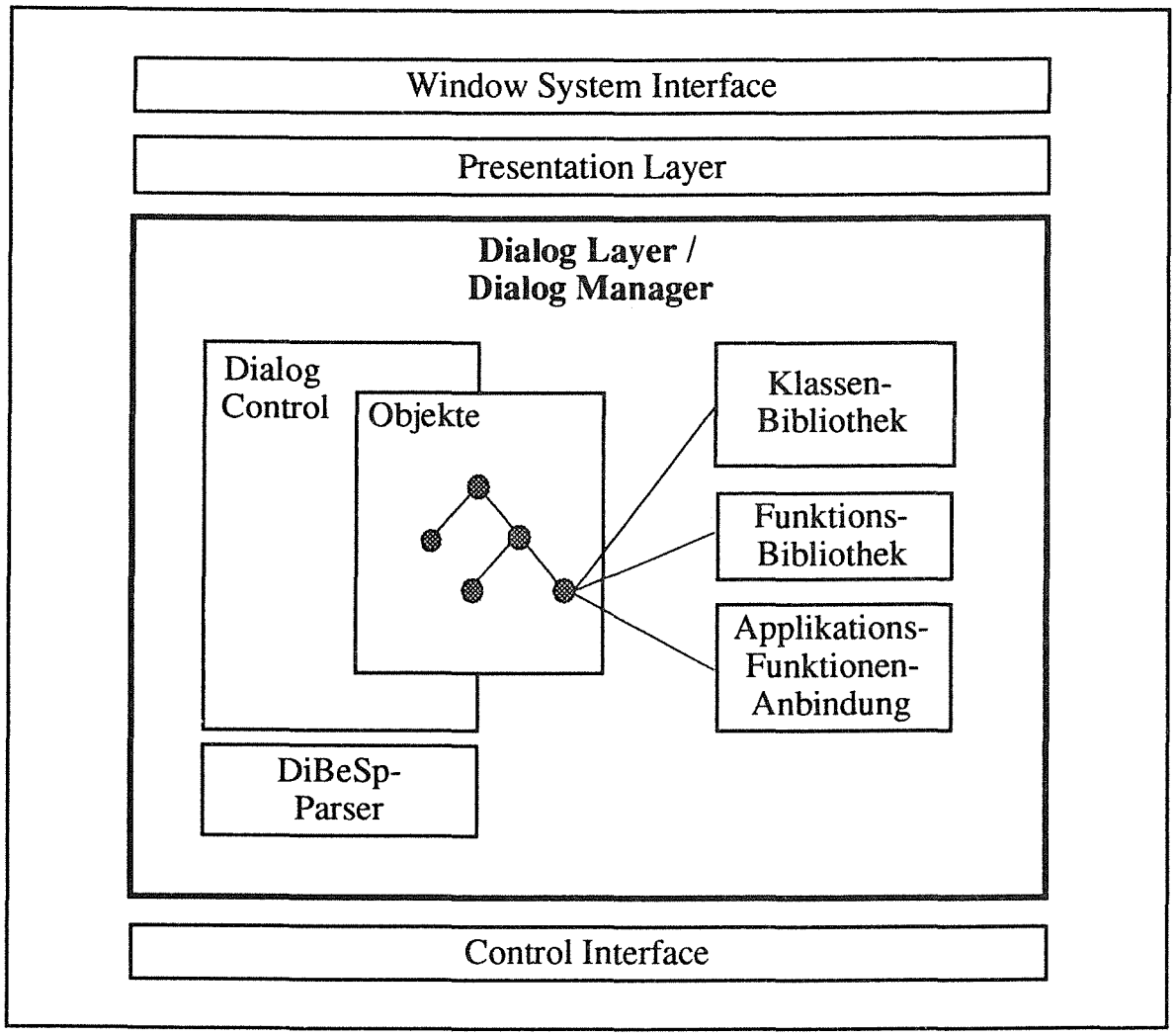


Abb. 4.18: Modell der Dialogschicht

Dialog-Control

Dieses Modul des Dialogmanagers ist der eigentliche Kern. Er verwaltet alle Objekte und ist für die Dynamik des Dialoges verantwortlich. Aus der Dialogbeschreibung der Objekte erzeugt es mit Hilfe der Klassenbibliothek die entsprechenden Instanzen, fügt sie in die Objekthierarchie des Dialoges ein und verwaltet sie. Die verwendeten internen Datenstrukturen (Klassen) sind unabhängig vom Fenstersystem.

Dialogbeschreibungssprachen - Parser (DiBeSp-Parser)

Dieser Parser hat einerseits die Aufgabe, die von der Kontrollschicht gelieferte Dialogbeschreibung in die interne Repräsentation umzusetzen. Andererseits ist

er für die Transformation der internen Objektstruktur in die Dialogbeschreibungssprache zuständig, die der Anwendung übermittelt wird.
Beispiel : Aufgrund einer Benutzeraktion erkennt das Dialog-Control-Modul, daß ein Teildialog (z.B. ein Formular) vom Benutzer beendet worden ist. Es übergibt nun dem DiBeSp-Parser die vom Benutzer geänderten Dialogobjekte, der daraus die entsprechende Dialogbeschreibung generiert und sie der Anwendung zukommen läßt.

Klassenbibliothek

Diese Bibliothek stellt die Basisklassen, zusammengesetzten Klassen und auf ihnen definierte Methoden zur Verfügung, die die Dialog-Control benötigt.

Funktionsbibliothek

In der Funktionsbibliothek sind Funktionen definiert, die keiner spezifischen Klasse zugeordnet werden können. Solche sind Plausibilitätsprüfungen, Exitfunktionen und allgemeine Funktionen, die beispielsweise der Transformation von Datentypen dienen.

Applikationsfunktionen - Anbindung

Mit Hilfe dieses Modules können applikationsspezifische Funktionen verwaltet werden. Diese Funktionen müssen dem Basissystem bekannt sein, bevor der Dialog mit einer Anwendung gestartet wird. Dies kann dadurch realisiert werden, daß Applikationsbibliotheken zur Verfügung gestellt werden.

Solche Applikationsfunktionen sind vorgesehen für Funktionalitäten, die von der Dialogbeschreibungssprache nicht oder nur unzureichend abgedeckt werden. Sie sind statisch und bilden damit eine Ausnahme in dem dynamischen Ansatz der Dialogsteuerung. Dies impliziert, daß ein Benutzer der Anwendung, der keinen Zugang zu diesen Funktionen hat, teilweise oder ganz auf die Nutzung dieser Anwendung verzichten muß.

Andererseits kann durch dieses Modul eine Visualisierung von graphischen Informationen erfolgen, trotz der Tatsache, daß die Dialogbeschreibungssprache nicht dafür ausgelegt ist.

4.6 Dialogbeschreibungssprache

Ziel der Dialogbeschreibungssprache (DiBeSp) ist es, den Dialog so genau wie möglich zu beschreiben, so daß die Benutzeroberfläche der Anwendung in ihrer vollen Funktionalität abgebildet werden kann. Andererseits muß sie so allgemein wie möglich gehalten und nicht an irgendein Fenstersystem gebunden sein, um eine Abbildung auf beliebige Fenstersysteme zu erlauben.

Zunächst soll auf die Objekte eines Dialoges eingegangen werden.

4.6.1 Objekte, Basisklassen und zusammengesetzte Klassen

Eine wesentliche Vorraussetzung für die Integration von mehreren Anwendungen in ein UIS ist das Vorhandensein eines Fenstersystems. Mit dessen Hilfe können zusammengehörige Objekte als solche gekennzeichnet werden und von anderen separiert werden. Die Zusammengehörigkeit wird durch Fenster realisiert, die einen Rahmen für die entsprechenden Objekte bilden.

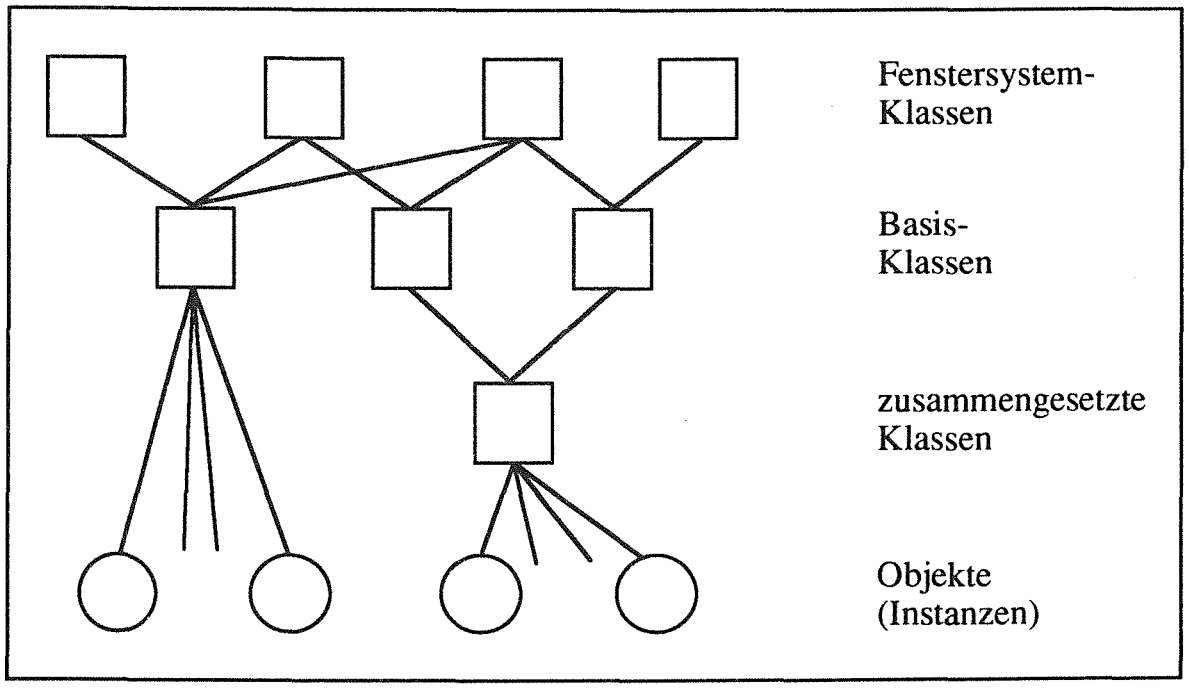


Abb. 4.19: Klassen, zusammengesetzte Klassen und Instanzen

Man unterscheidet Basis-, zusammengesetzte Klassen und Objektinstanzen. Instanzen können entweder von zusammengesetzten oder von Basisklassen

gebildet werden und erben alle Eigenschaften der Klasse. Näheres zum objektorientierten Ansatz entnehme man [Cox 86], [Schlüter 90].

Die Präsentation der Basisklassen wird durch die Abbildung auf die Klassen des Fenstersystems vorgenommen.
In fast allen Dialogsystemen kommen gewisse Primitive von Dialogobjekten vor. Aus diesen wurden die einzelnen Basisklassen gebildet. Da diese Klassen bereits eine Funktionalität enthalten, ist mit ihrer Definition auch ein Teil der Dynamik eines Dialoges beschrieben. Entsprechend dem objektorientierten Ansatz wird die Repräsentation jedes Objektes von ihm selbst vorgenommen (durch Verwendung der ererbten Methoden der Fenstersystemklassen). Die zusammengesetzten Klassen stellen bestimmte typische Dialogarten zur Verfügung (s.u.).

Die Begriffe für die nun folgenden Klassen sind an den OSF/Motif-Style-Guide [OSF-SG 90] und den ISA-Dialogmanager [ISA 91] angelehnt.

Basisklasse	Beschreibung
Dialog	Kennzeichnet den gesamten Dialog einer Anwendung. Er dient als Vater aller zu dieser Anwendung gehörenden Dialogobjekte.
Window	Fenster, in dem Objekte definiert werden können, die logisch zusammengehören und einen abgeschlossenen Teil eines Dialoges beschreiben.
Groupbox	Mit Hilfe der Groupbox können innerhalb eines Windows mehrere Dialogobjekte als zusammengehörig definiert werden. Diesen Objekten dient die Groupbox als Vater.
Poptext	besteht aus einer Liste von Auswahlmöglichkeiten, aus denen der Benutzer wählen kann, wobei immer nur der selektierte angezeigt wird.
Poptextitem	Ein Element der Auswahlliste eines Poptextes.
Listbox	besteht wie der Poptext aus einer Liste von Auswahlelementen, von denen entweder eines oder mehrere ausgewählt werden können. Im Gegensatz zum Poptext werden alle Elemente ständig angezeigt.
Listboxitem	Ein Element aus der Auswahlliste einer Listbox
Pushbutton	Ein Pushbutton kann dazu genutzt werden, bestimmte Aktionen innerhalb eines Dialoges durch den Benutzer ausführen zu lassen (Help, OK, Cancel)

Radiobutton	dient der Darstellung von Wahrheitswerten. Werden mehrere Radiobuttons definiert, so kann immer nur einer von diesen vom Zustand Wahr sein.
Checkbox	wird verwendet, um einen Wahrheitswert darzustellen. Im Gegensatz zum Radiobutton, können auch mehrere Checkboxen den Wert Wahr besitzen.
Edittext	ist ein allgemeines Eingabefeld für beliebige alpha-numerische Informationen.
Statictext	ist ein nicht modifizierbarer Text. Er dient der Information des Benutzers, wobei er keine relevante Dialogfunktionalität besitzt.
Menubox	Die Menubox dient als Kontainer für mehrere Menüeinträge. Sie wird in der Menüzeile des zugehörigen Windows eingetragen und zeigt bei Aktivierung die in ihr enthaltenen Einträge an, aus denen der Benutzer einen auswählen kann. Die Menubox kann ebenfalls Kind eines beliebigen anderen Objektes sein. In diesem Falle wird es jedoch nicht durch einen Menübalken dargestellt, sondern bleibt bis zur Selektion ihres Vaterobjektes unsichtbar.
Menuitem	Ist ein Eintrag in einer Menubox. Mit Hilfe dieser Einträge können anwendungsbezogene Funktionen aufgerufen werden (Datei laden, Sichern, Hilfe, ...)
Menusep	Der Menuseperator dient der optischen Abgrenzung von Menüeinträgen.

Die genannten Basisklassen erlauben es, beliebige Dialoge zu erstellen. In der Praxis kommen jedoch bestimmte Dialogtypen wiederholt zum Einsatz (Menüs, Hilfefenster, Statusanzeigen usw.), weshalb der Sprachumfang der Dialogbeschreibungssprache um die "zusammengesetzten Klassen" erweitert wurde, die dem Benutzer zur Vereinfachung der Dialogbeschreibung dienen.

zusammengesetzte Klasse	**Beschreibung**
Choose	Choose wird zur Auswahl eines oder mehrer Elemente aus einer Liste verwendet, wobei entweder alle Einträge oder nur das selektierte angezeigt wird.
Chooseitem	Stellt ein Element der Auswahlliste der Klasse Choose dar.

XMenu	Mit Hilfe des XMenu können klassische Menüs generiert werden. Es besteht aus einem Window, welches in die drei Teilbereiche "Allgemeine Informationen", "Auswahlbereich" und "Pushbuttons" gegliedert ist.
XMenuitem	Ist ein Menüeintrag der Klasse XMenu.
Formular	Dient der Erstellung von Formularen. Innerhalb eines Windows sind vier Teilbereiche vorhanden : "Allgemeine Informationen", "Arbeitsbereich", "Statusanzeige" und "Pushbuttons".
History	Ein History-Objekt ist eine Groupbox, innerhalb der eine erweiterbare, nicht veränderbare Liste von Strings verwaltet wird.
Table	Mit Hilfe dieser Klasse können zwei-dimensionale Tabellen erzeugt werden.
TabLine	Zeile einer Tabelle.
TabCol	Spalte einer Tabelle.
Infobox	Das Infobox-Objekt dient der Darstellung von Informationen bzw. Statusanzeigen.
Infowindow	Ist für Mitteilungen an den Benutzer vorgesehen, die dieser explizit bestätigen muß. Hierzu wird ein eigenes Window erzeugt.

Ein Dialog ist streng hierarchisch aufgebaut. Zwischen den einzelnen Objekten bestehen also Vater-Sohn- und Bruder-Beziehungen. Die Wurzel eines Dialoges wird von einer Instanz der Dialogklasse gebildet. Dieses Objekt ist für den Benutzer nicht sichtbar und dient der Zuordnung von Dialogobjekten zu einem bestimmten Kontext (z.B. einer Anwendung).

Die direkten Nachfahren der Dialogklasse sind die Fenster (Instanzen der Window-Klasse). Alle weiteren Abhängigkeiten und Vererbungsmöglichkeiten gehen aus der Abbildung 4.20 hervor.

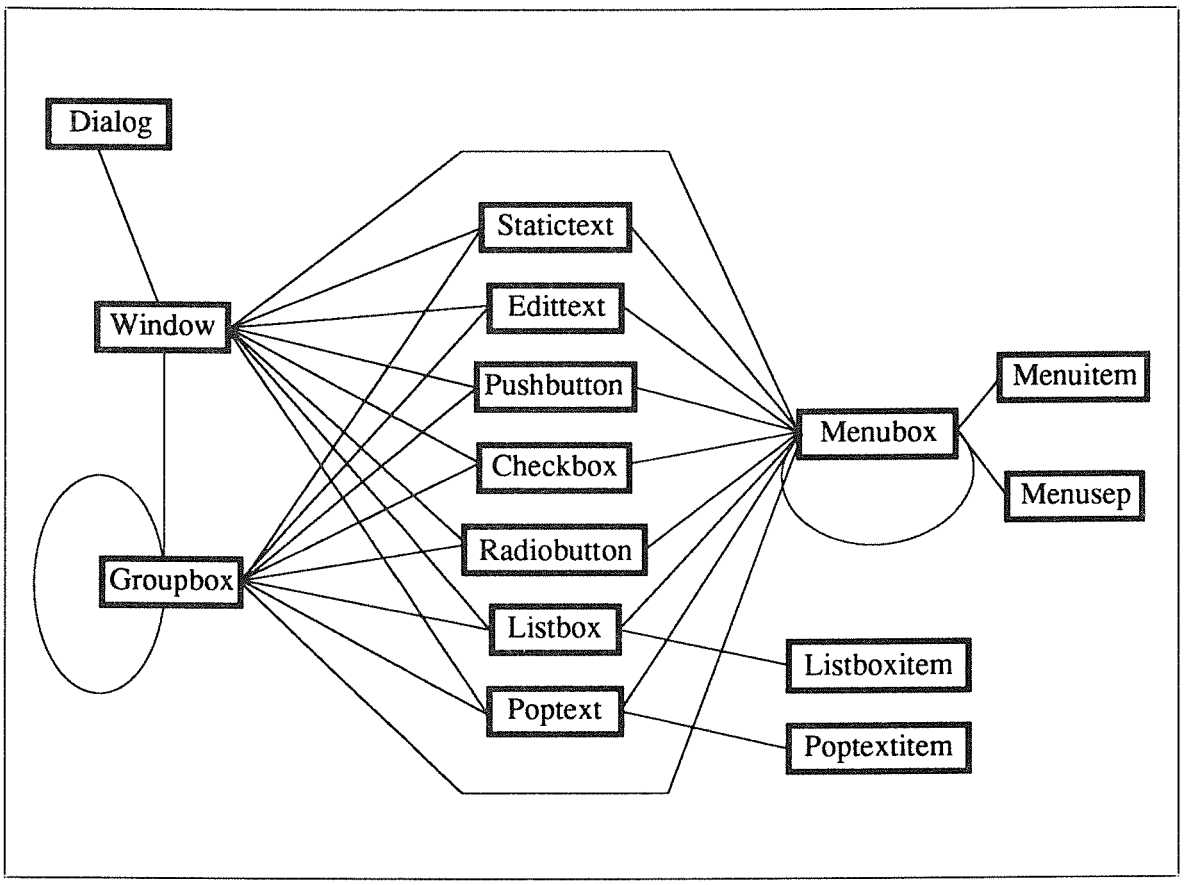


Abb. 4.20: Klassenprimitive und Vererbungsmöglichkeiten

4.6.2 Objekt - Attribute

Die Attribute eines Objektes und deren Bedeutung hängen von ihrer Klasse ab. Wird der Wert eines Attributes eines Objektes verändert, so ist die Änderung nur lokal auf das Objekt bezogen. Die Klasse bleibt davon unbetroffen. Bestimmte Attributwerte haben jedoch die Eigenschaft, daß sie für alle Kinder des Objektes gelten. Wird beispielsweise ein Vaterobjekt unsichtbar, so wirkt sich diese Änderung auf alle Kinder aus - sie werden ebenfalls unsichtbar. Dahingegen wirkt sich das Sichtbar-Werden eines Vaterobjektes nur auf die Kinder aus, die vor dem Unsichtbar-Werden eines Vaterobjektes sichtbar waren; alle vorher unsichtbaren Kindern bleiben in diesem Zustand.

Zur besseren Unterscheidung der im folgenden verwendeten Bezeichner werden ihnen eindeutige Kennungen vorangestellt, sofern dies nicht aus dem Zusammenhang ersichtlich ist.

Attr_	: Attribut
DT_	: Datentyp
DCmd_	: Dialogbeschreibungssprachen Befehl (Command)
DRes_	: Dialogbeschreibungssprachen Antwort (Response)
DCB_	: Dialogbeschreibungssprachen Ereignis (Callback)

Attribut	Beschreibung
Attr_geoBehaviour	Wird für die Positionierung eines Objektes und seines geometrischen Verhaltens im Bezug zu seinem Vaterobjekt verwendet (wird näher ausgeführt).
Attr_size	Größenangabe des Objektes (wird näher ausgeführt).
Attr_inputRequired	Ein derart markiertes Objekt muß vom Benutzer bearbeitet werden.
Attr_multisel	Innerhalb einer Auswahl können entweder mehrere Selektionen gemacht werden oder nur eine. (Bsp.: Listbox, Choose)
Attr_selected	Ein Element einer Auswahlliste ist selektiert oder nicht.
Attr_showAll	Alle Elemente einer Auswahl sollen ständig sichtbar sein oder nur die jeweils selektierten.
Attr_sensitive	Das Objekt kann vom Benutzer bearbeitet werden oder ist ihm nicht zugänglich.
Attr_visible	Sichtbarkeit eines Objektes für den Benutzer.
Attr_active	Markiert das Objekt, welches der Benutzer derzeit bearbeitet. Durch Kennzeichnen eines Objektes mit Attr_active kann beim Initialisieren eines Teildialoges festgelegt werden, daß der Dialog mit diesem Objekt beginnen soll. Innerhalb eines Fensters kann maximal ein Objekt aktiv sein.
Attr_delOnSelect	Selektiert der Benutzer das Objekt, so wird dessen Inhalt gelöscht oder bleibt erhalten.

Attr_alert	Dies ist ein Attribut, welches nur bei Fenstern vorkommen kann. Ist es gesetzt, so wird die Bearbeitung aller anderen Fenster gesperrt. Es kann zur Bestätigung wichtiger Nachrichten an den Benutzer verwendet werden, die dieser explizit bestätigen muß, bevor er weiterarbeiten kann.
Attr_alignment	Dieses Attribut dient der Beschreibung, wie die Kinder eines Objektes dargestellt werden sollen. Mögliche Werte sind horizontal oder vertikal. Bei horizontaler Ausrichtung werden alle Kindobjekt zeilenorientiert dargestellt, bei vertikaler spaltenorientiert. Dieses Attribut kann von dem Dialogmanager oder dem Fenstersystem ignoriert werden.
Attr_type	Beschreibt den Datentyp des Objektinhaltes (siehe Datentypen).
Attr_constraint	Mit Hilfe dieses Attributes können Funktionen angegeben werden, die die Wertebereichsüberprüfung des Objektinhaltes vornehmen. Somit kann der Benutzer zu einer bestimmten Eingabe gezwungen werden, ohne daß die Anwendung selber diese Überprüfung übernehmen muß.
Attr_content	Inhalt (Wert) des Objektes. Dieses Attribut wird nur gebraucht, wenn es neben dem anzuzeigenden Namen des Objektes noch einen expliziten Inhalt gibt (Edittext).
Attr_appContent	Wird dieses Attribut beim Setzen des Inhaltes des Objektes angegeben, so wird der bereits vorhandene Inhalt um den zu setzenden erweitert.
Attr_format	Der Inhalt eines Objektes kann in einem bestimmten Format angegeben werden. (siehe Textattribute - Formatierung)
Attr_name	Jedes für den Benutzer zugängliche Objekt kann durch einen Namen gekennzeichnet werden, durch den sein Inhalt für den Benutzer beschrieben wird (siehe auch Attr_content).

Attr_help	Dieses Attribut kann von der Anwendung genutzt werden, um Hilfetexte zu speichern, die bei Erklärungsbedarf des Benutzers angezeigt werden. Die Verwaltung und Darstellung der Hilfetexte obliegt der Anwendung (Statuszeile, Hilfe-Fenster, ...), mit der Ausnahme der zusammengesetzten Klasse Formular (s.u.).
Attr_exitFlag	Ist dieses Attribut gesetzt und selektiert der Benutzer das entsprechende Objekt bzw. verläßt es nach dessen Bearbeitung, so wird der aktuelle Teildialog unterbrochen und eine Meldung mit Hilfe einer Exit-Funktion an die Anwendung weitergeleitet. (siehe Exitfunktionen)
Attr_exitFunc	Für dieses Attribut kann eine vorher definierte Funktion angegeben werden, die ausgeführt wird, sobald der Benutzer die Bearbeitung des aktuellen Objektes beendet hat (siehe exitFlag). Sie wird nur ausgelöst, wenn <exitFlag> gesetzt ist. (siehe hierzu den Abschnitt Anbindung an die Anwendung)
Attr_defaultExit	Wird dieses Attribut eines Objektes gesetzt und kann aufgrund einer bestimmten nicht objektbezogenen Benutzerinteraktion ein Teildialog beendet werden, so wird die <exitFunc> dieses Objektes ausgelöst.(Bsp.: Der Benutzer drückt während der Bearbeitung eines Objektes eine bestimmte Taste (z.B. F10), um den Teildialog zu beenden).
Attr_userID	Jedes Objekt kann zu seiner Identifizierung für die Anwendung mit einer ID versehen werden.
Attr_userData	Mit Hilfe dieses Attributes kann die Applikation beliebige Daten zu einem Objekt ablegen. Diese werden nicht dargestellt und dienen zur (internen) Information für die Anwendung.
Attr_menuStyle	Beschreibt das Verhalten der Menüeinträge einer Menübox. (siehe Zuordnung Attribute zu Objekten)
Attr_cl_nr	Dieses Attribut findet nur bei Tabellen Anwendung und bezeichnet eine Spalte bzw. eine Zeile innerhalb der Tabelle.

Eine Sonderstellung nimmt das Attribut ***Object_ID*** ein. Jedes Objekt verfügt über eine eindeutige Kennzeichnungen, die bei seiner Definition angegeben wird. Sie ist nicht veränderbar. Näheres zur Object_ID entnehme man dem Befehlssatz.

Es bestehen bestimmte Abhängigkeiten zwischen Attributen, auf die nun näher eingegangen wird.

sensitive - visible - active

Ist ein Objekt sichtbar, so kann es selektierbar sein oder nicht. Ist es dahingegen nicht sichtbar, so kann es nicht selektiert werden, obgleich das Attribut *sensitive* gesetzt sein kann. Das gleiche gilt für das Attribut *active*. Des weiteren sind diese Attribute von der Hierarchie abhängig, in der sich das Objekt befindet. Sie können von denen der Vorfahren überlagert werden. Ist der Vater des Objektes unsichtbar, so ist es auch das Objekt selbst. Gleiches gilt für *sensitive* mit den entsprechenden Auswirkungen auf *active* .

selected - multisel

Ist das Attribut *multisel* eines Auswahlelementes nicht gesetzt, also nur ein Element der Auswahl selektierbar, so kann maximal ein Element das Attribut *selected* gesetzt haben.

showAll - multisel

Die Kombination von *showAll* nicht gesetzt und *multisel* gesetzt bewirkt, daß im nicht aktivierten Zustand des Objektes nur die selektierten Auswahlelemente angezeigt werden.

exitFlag - exitFunc - defaultExit

Ist *exitFlag* gesetzt, so wird die in *exitFunc* spezifizierte Funktion ausgelöst. Wird *exitFunc* nicht angegeben, so wird in der Hierarchie nach dem ersten Vorfahren des Objektes gesucht, welcher eine ExitFunktion spezifiziert hat, welche dann ausgeführt wird.

4.6.3 Geometrieattribute

Die Geometrie eines Objektes wird beschrieben durch seine Lage in Bezug zu seinem Vater und Angaben über seine Größe.

Die relative Lage zum Vaterobjekt wird beschrieben durch die Angabe des Abstandes zu den Begrenzungen des Vaters.

Die Größe eines Objektes kann auf verschiedene Weisen angegeben werden. Sie kann automatisch berechnet werden in Bezug auf die Begrenzungen des Vaterobjektes oder aufgrund des Inhaltes des Objektes. Oder sie kann durch explizite Angabe einer Größe festgelegt werden (vergleiche auch [ISA 91]).

Dabei gilt folgende Vorrangregel :

Automatische Berechnung zur Vater-Begrenzung
vor direkter Größenangabe
vor automatischer Berechnung aufgrund des Inhaltes.

Die genaue Syntax der Geometriebeschreibung lautet :

<*Attr_GeoBehaviour*> ::= <Boundaries> <Automatics>;

<Boundaries> ::= <x-Boundary> <delta-x> <y-Boundary> <delta-y>;

<Automatics> ::= <x-AutoBoundary> <delta-x> <y-AutoBoundary> <delta-y>;

<*Attr_Size*> ::= <x-Coord> <y-Coord>;

„Boundary" gibt dabei die entsprechende Begrenzung des Vaters an und der „delta"-Wert den Abstand zu dieser Begrenzung. Dieser Wert bleibt bei einer Veränderung der Größe oder Position des Vater-Objektes immer konstant. Das Objekt behält seine Position im Bezug zum Vater also konstant bei.

Es gibt vier Begrenzungen :

Linke, obere, rechte und untere Begrenzung (border).

In Abhängigkeit von den <Boundaries> können nur bestimmte <Automatics> angegeben werden. Die folgende Tabelle zeigt diese Abhängigkeiten auf:

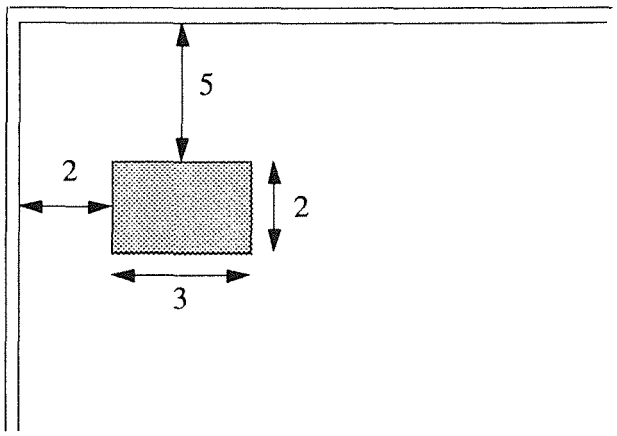
x-Boundary	y-Boundary	=>	x-AutoBoundary	y-AutoBoundary
left	top		right	bottom
left	bottom		right	top
right	top		left	bottom
right	bottom		left	top

Wird ein Objekt zum Beispiel bündig zur oberen und linken Begrenzung des Vater-Objektes positioniert, so kann seine Größe nur noch im Bezug auf die rechte und bzw. oder die untere Begrenzung automatisch berechnet werden.

Die <AutoBoundaries> sind optional; werden sie nicht benötigt, so wird der entsprechende <delta> - Wert mit der Konstante *not_defined* belegt.

Anhand der folgenden Beispiele soll dieses Prinzip verdeutlicht werden:

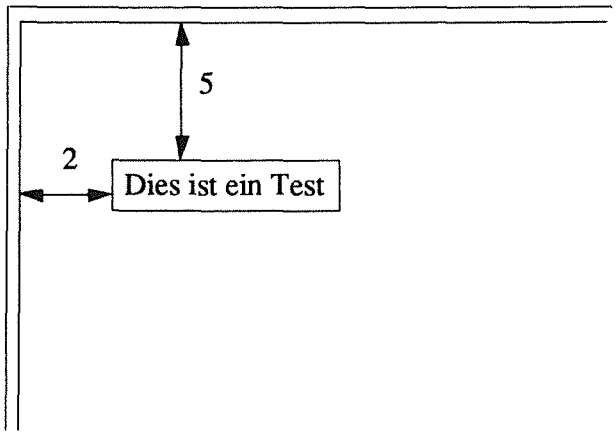
Beispiel A:



<x-Boundary> <delta-x>	:	left	2
<y-Boundary> <delta-y>	:	top	5
<x-Coord>	:	3	
<y-Coord>	:	2	

Das Objekt ist mit festem Abstand von 5 Einheiten an die obere Begrenzung und mit 2 Einheiten an die linke Begrenzung des Vaterobjektes gebunden. Dieser Abstand bleibt konstant unabhänig von der aktuellen Position und Größe des Vaters. Es erfolgt keine automatische Größenberechnung, da keine <AutoBoundaries> angegeben wurden. Die Größe wird aufgrund der direkten Angaben festgelegt.

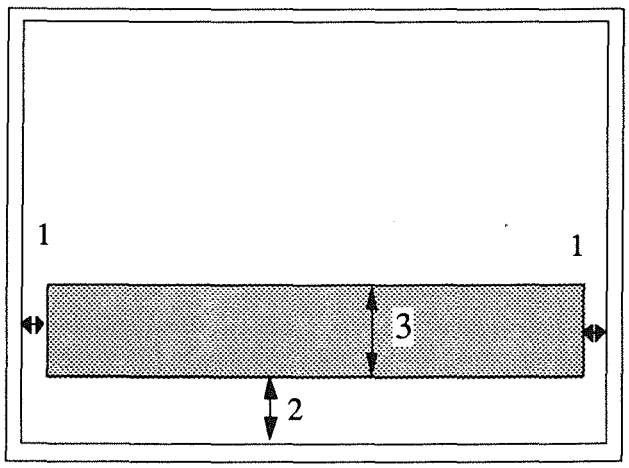
Beispiel B:



<x-Boundary> <delta-x>	:	left	2
<y-Boundary> <delta-y>	:	top	5

Wie in Beispiel A ist das Objekt bündig zur linken und oberen Begrenzung des Vaters. Da weder <AutoBoundaries> noch Größenangaben vorhanden sind, erfolgt die Berechnung der Größe auf der Basis des Inhaltes („Dies ist ein Test").

Beispiel C :



<x-Boundary> <delta-x>	:	left	1
<y-Boundary> <delta-y>	:	bottom	2
<x-AutoBoundary> <delta-x>	:	right	1
<y-Coord>	:	3	

Unabhängig von der Breite des Vaterobjektes wird das Objekt immer mit festem Abstand von jeweils einer Einheit zum linken und rechten Rand sowie zwei

Einheiten zum unteren Rand positioniert. Bei Veränderung der Größe des Vaters wird die des Objektes entsprechend angepaßt.

4.6.4 Textattribute - Formatierung

Mit Hilfe des Attributes *format* kann man Eingabemasken definieren, die dem Benutzer bei der Bearbeitung eines Textfeldes (Edittext) als Hilfestellung dienen. Sie ermöglichen die Formatierung von Eingaben während der Eingabe.

Die nun folgende Beschreibung des Attributes *format* ist [ISA 91] entnommen.

Der Wert des Attributes *format* ist ein String, welcher folgende Syntax hat :

<formatString> ::= <numberFormat> | <textFormat>;

<numberFormat> ::=

%{-}<length>[.<decimalDigits>][<sign>]<NumberingSystem>;

<textFormat> ::=

A | U | X | C | N | S [<S-Char>] | <formatingSign>;

<formatingSign> ::=

"/" | "," | "." | "-" | ":" | " " ;

Dabei bedeutet :

Symbol	Erklärung
%	Kennzeichnung für Zahleneingabe
-	Rechts- oder linksbündige Darstellung („-“ angegeben => linksbündig)
<length>	Gesamtzahl von Zahlen die eingegeben werden können, einschließlich der Nachkommastellen
<decimalDigits>	Anzahl der Nachkommastellen

<sign>	Kontrolliert, ob negative Zahlen eingegeben werden können. Werte : „u“ und „s“. Default ist „u“. Dabei steht „u“ für unsigned, also nur positive Eingaben und „s“ für signed. Im letzteren Fall sind sowohl negative als auch positive Eingaben erlaubt.
<NumberingSystem>	Angabe des verwendeten Zahlensystems, wobei folgende Möglichkeiten bestehen: d dezimal (Basis 10) h hexadezimal (Basis 16) o oktal (Basis 8) b binär (Basis 2)
A	alphabetische Zeichen
U	alphabetische Zeichen, nur Großbuchstaben
X	keine Einschränkung
C	Großbuchstaben und Zahlen
N	Zahleingabe
S	Secret, geheime, nicht sichtbare Eingabe, wird <S-Char> angegeben, dann wird dieses Zeichen bei der Eingabe ausgegeben für jedes Zeichen, das der Benutzer eingibt.

4.6.5 Datentypen

Jedes Objekt wie auch seine Attribute haben einen bestimmten Datentyp.

Folgende Datentypen sind definiert :

Datentyp	Erläuterung
DT_IDString	ASCII String. Nullterminierter String. Erstes Zeichen ist ein Großbuchstabe, "$" oder "_". Alle weiteren Zeichen sind aus der Menge {a,..,z,A,..,Z,0,..,9,_,$}.

DT_String	ASCII String. Nullterminierter String. Alle Zeichen erlaubt. Formatierungszeichen "\n" zur Zeilentrennung. Das Zeichen "\" wird durch Verdopplung dargestellt "\\".
DT_uInt	unsigned Integer Länge : 32 Bit (31 ... 0) MSB = 31. Bit (most significant byte)
DT_Int	signed Integer Länge : 32 Bit (31 ... 0) MSB = 30. Bit Sign = 31. Bit (0 : Wert ≥ 0, 1 : Wert < 0)
DT_Boolean	Wahrheitswert. Länge : 32 Bit FALSE <=> 0 TRUE <=> != 0
DT_Char	Character Länge : 8 Bit (7 ... 0) Kodiert nach ISO 8859/1
DT_Size	siehe Geometrieattribute. <x-Coord> und <y-Coord> sind vom Typ DT_uInt.
DT_GeoBehaviour	siehe Geometrieattribute. <x-Boundary>, <y-Boundary>, <x-AutoBoundary>, <y-AutoBoundary> sind vom Typ DT_Char, wobei diese beschränkt sind auf die Menge {L,R,T,B}. (L=left, R=right, T=top, B=bottom) <delta-x> und <delta-y> sind vom Typ DT_uInt.
DT_AnyData	<size> <data> Ein <size> Byte großer Datenblock beliebigen Datentyps. <size> ist vom Typ DT_uInt. (Länge eines Bytes = 8 Bit)
DT_Date	Datumsangabe vom Typ DT_String im Format: "TT.MM.JJJJ" oder ein Leerstring. T = Tag, M = Monat, J = Jahr z.B. "04.05.1991"

DT_Range	Bereichsangabe vom Typ DT_String. Das erste Zeichen ist aus der Menge {"<", ">"}.
DT_IntString	DT_String bestehend aus einem Leerstring oder der Zeichendarstellung eines Wertes vom Typ DT_Int.
DT_uIntString	DT_String bestehend aus einem Leerstring oder der Zeichendarstellung eines Wertes vom Typ DT_uInt.
DT_ExitFunc	Anbindung an die Anwendung. <ExitFunc> [<ObjectID>] [<changed_only>] [<pause_dialog>] Dabei sind <ExitFunc> vom Typ DT_uInt, <changed_only> und <pause_dialog> vom Typ DT_Boolean (siehe Anbindung der Anwendung).
DT_Constraint	Beschreibung einer Constraint-Funktion (wird näher ausgeführt).

Constraint-Funktionen

Constraintfunktionen dienen der Wertebereichsüberprüfung von Benutzereingaben und werden als Attributwerte des Attributs *constraint* für jedes Objekt angegeben.

Die Angabe einer Constraint-Funktion geschieht in folgender Notation :

<Constr.Fkt> <DataType> <Para> {<Para>}

Der Funktionsname <Constr.Fkt> ist vom Typ DT_uInt und gibt die Funktion an, die ausgeführt werden soll. Diese muß dem System bekannt sein. <DataType> gibt den Datentyp der folgenden Funktionsparameter an und <Para> sind die Parameter der Funktion.
Constraints beziehen sich nur auf die vom Benutzer eingegebenen Werte und nicht auf Werte, die aus einer Auswahl übernommen werden. Ist beispielsweise zu einem Edittext eine Menubox definiert und wählt der Benutzer einen Menüeintrag als Eingabe aus, so wird diese nicht überprüft.

Derzeit ist eine allgemeine Constraint-Funktion vorgesehen :

Boolean XConstr_Range(<dataType>, <minVal>, <maxVal>);

Sie überprüft, ob der Wert des Objektes in den Grenzen <minVal> und <maxVal> einschließlich liegt. <minVal> und <maxVal> sind vom Datentyp

<dataType>, der nicht mit dem des Objektes übereinstimmen muß. Es erfolgt eine automatische Konvertierung der Datenformate. Wahlweise kann <minVal> durch die Konstante "Minimum" bzw. <maxVal> durch "Maximum" ersetzt werden, um keine explizite untere bzw. obere Grenze vorzugeben.

4.6.6 Zuordnung von Objektattributen zu Datentypen

Attribut	Datentyp
Attr_geoBehaviour	DT_GeoAttribute
Attr_size	DT_Size
Attr_inputRequired	DT_Boolean
Attr_multisel	DT_Boolean
Attr_selected	DT_Boolean
Attr_showAll	DT_Boolean
Attr_sensitive	DT_Boolean
Attr_visible	DT_Boolean
Attr_active	DT_Boolean
Attr_delOnSelect	DT_Boolean
Attr_alert	DT_Boolean
Attr_alignment	DT_uInt
Attr_type	DT_uInt
Attr_constraint	DT_Constraint
Attr_content	DT_String
Attr_appContent	DT_String
Attr_format	DT_String
Attr_name	DT_String
Attr_help	DT_String
Attr_exitFlag	DT_Boolean
Attr_exitFunc	DT_ExitFunc
Attr_defaultExit	DT_Boolean
Attr_userID	DT_String
Attr_userData	DT_AnyData
Attr_menuStyle	DT_uInt
Attr_cl_nr	DT_uInt

Der Wert des Attributes Attr_content wird als String angegeben und entsprechend des Attributes Attr_type behandelt.

4.6.7 Konstanten

Da die Attribute Attr_menuStyle (s.u.) und Attr_alignment durch die Abbildung auf die oben genannten Datentypen nur unzureichend beschrieben sind, seien

folgende Konstanten vom Typ DT_uInt definiert, mit denen die entsprechenden Attributwerte beschrieben werden :

Der Wert von ***Attr_menuStyle*** ist aus der Menge

{ Pushbutton_Style, Radiobutton_Style, Checkbox_Style }.

Der Wert von ***Attr_alignment*** ist aus der Menge

{ horizontal, vertical }.

Für die **Constraint-Funktionen** sind folgende Konstanten definiert:

{ Minimum, Maximum}.

Die optionale Angabe von <AutoBoundary> der **Geometrieattribute** wird ermöglicht durch die Konstante

not_defined.

4.6.8 Zuordnung von Attributen zu Objektklassen

In Abhängigkeit von der Klasse haben die Attributnamen eine bestimmte Bedeutung. Ebenso sind für bestimmte Klassen nicht alle Attribute zugelassen. Deshalb wird nun auf die Attribute der einzelnen Klassen im speziellen eingegangen.

Folgende Attribute werden nicht explizit aufgeführt, da sie für alle Klassen definiert sind und sich aus dem Kontext ergeben (Ausnahme hierbei ist der Menüseparator, siehe unten):

Attr_help,
Attr_userID,
Attr_userData,
Attr_sensitive,
Attr_active,
Attr_model,
Attr_size,
Attr_visible.

Dialog

Die Dialogklasse selber hat keine Attribute, die vom Anwendungsprogramm modifiziert werden können. Sie dient dem Dialogmanager dazu, eine Zuordnung zwischen Dialog und Anwendung herzustellen.

Window

Die Window-Klasse dient der Bildung von abgeschlossenen Teildialogen, bestehend aus Objekten, die in einem Fenster gruppiert werden.

Attribut	Window
Attr_geoBehaviour	Das geometrische Verhalten wird auf das Display des Benutzers bezogen. Die Begrenzungen sind die des zugrunde liegenden Fenstersystems.
Attr_name	Name (Titel) des Fensters, der als Überschrift eingeblendet wird.
Attr_alert	Andere Fenster können nicht bearbeitet werden.
Attr_alignment	Ausrichtung der Objekte innerhalb des Fensters.

Groupbox

Attribut	Groupbox
Attr_geoBehaviour	Das geometrische Verhalten wird auf das Vaterobjekt der Groupbox bezogen, diese kann entweder eine Groupbox oder ein Window sein.
Attr_alignment	Ausrichtung der Objekte innerhalb der Groupbox.

Statictext

Attribut	Statictext
Attr_geoBehaviour	Geometrieverhalten bezogen auf das Vaterobjekt
Attr_content	Anzuzeigender Text vom Typ DT_String

Edittext

Attribut	Edittext
Attr_geoBehaviour	Geometrieverhalten
Attr_name	anzuzeigender Name des Objektes
Attr_inputRequired	Es muß ein Wert eingegeben werden
Attr_delOnSelect	Inhalt ist zu löschen bei Selektion durch Benutzer

Attr_type	Datentyp des Inhaltes (Attr_content)
Attr_content	Inhalt des Edittextes
Attr_format	Eingabeformat
Attr_exitFlag	Nach Beendigung der Bearbeitung des Edittextes wird der Teildialog mit der Exit-Funktion verlassen.
Attr_exitFunc	s.o.
Attr_defaultExit	s.o.

Pushbutton

Attribut	Pushbutton
Attr_geoBehaviour	Geometrieverhalten
Attr_selected	Der Pushbutton wurde gedrückt
Attr_name	Text des Pushbuttons
Attr_exitFlag	Nach Drücken des Pushbuttons soll der Teildialog (Window) beendet werden.
Attr_exitFunc	Auszuführende Funktion beim Verlassen des Teildialoges
Attr_defaultExit	s.o.

Checkbox

Attribut	Checkbox
Attr_geoBehaviour	Geometrieverhalten
Attr_name	anzuzeigender Name des Objektes
Attr_selected	Status der Checkbox ist "selektiert".
Attr_exitFlag	Nach Verändern des Status der Checkbox wird der Teildialog verlassen mit der in Attr_exitFunc angegebenen Exit-Funktion
Attr_exitFunc	s.o.
Attr_defaultExit	s.o.

Radiobutton

Attribut	Radiobutton
Attr_geoBehaviour	Geometrieverhalten
Attr_name	anzuzeigender Name des Objektes

Attr_selected	Status des Radiobuttons ist "selektiert"
Attr_exitFlag	siehe Checkbox
Attr_exitFunc	" "
Attr_defaultExit	" "

Listbox

Die Listbox dient als Kontainer für die einzelnen Listboxitems. Daher sind die wesentlichen Attribute bei den Listboxitems zu finden.

Attribut	Listbox
Attr_geoBehaviour	Geometrieverhalten
Attr_multisel	Es können mehrere Elemente der Listbox ausgewählt werden.

Listboxitem

Das Listboxitem selber hat kein Geometrieverhalten und keine Größe. Diese werden durch das Vaterobjekt, die Listbox bestimmt. Das Attribut Attr_defaultExit steht ebenfalls nicht zur Verfügung, da ein Default-Exit nur über ein dem Benutzer sichtbares Objekt abgewickelt werden soll. Da es ist nicht vorherbestimmbar, ob ein Listboxitem sichtbar ist oder nicht, kann es dieses Attribut nicht besitzen. Die Sichtbarkeit (nicht zu verwechseln mit Attr_visible) wird bedingt durch die Anzahl der Listboxitems und die Größe der Listbox. Es können also Listboxitems im nicht sichtbaren Bereich der Listbox liegen. Ein über ein solches verstecktes Listboxitem ausgelöster Default-Exit ist für den Benutzer nicht nachvollziehbar.

Attribut	Listboxitem
Attr_selected	Das Listboxitem ist selektiert.
Attr_name	Anzuzeigender Text vom Typ DT_String
Attr_exitFlag	Aufgrund der Selektion bzw. Reselektion eines Listboxitems wird der Teildialog verlassen mit der in Attr_exitFunc angegebenen Funktion
Attr_exitFunc	s.o.

Poptext

Im Gegensatz zur Listbox zeigt der Poptext immer nur den jeweils selektierten Poptexteintrag an. Es kann demnach auch nur maximal ein solcher selektiert werden. Aktiviert der Benutzer einen Poptext (beispielsweise durch Anklicken),

so erhält er mit Hilfe eines kleinen Fensters die Liste von Poptextitems angezeigt, aus der er wählen kann.

Das Attribut *defaultExit* kann bei dem Poptext selber oder aber bei einem bzw. mehreren Poptextitems gesetzt werden, jedoch nicht bei beiden zur gleichen Zeit.

Attribut	Poptext
Attr_geoBehaviour	Geometrieverhalten
Attr_name	anzuzeigender Name des Objektes
Attr_exitFlag	s.o.
Attr_exitFunc	s.o.
Attr_defaultExit	s.o.

Poptextitem

Genau wie das Listboxitem hat auch das Poptextitem kein geometrisches Verhalten, da dieses durch den Poptext(-vater) bestimmt wird.

Attribut	Poptextitem
Attr_name	anzuzeigender Name des Objektes
Attr_selected	das derzeit selektierte Poptextitem hat als einziges den Status "selektiert"
Attr_exitFlag	siehe Poptext
Attr_exitFunc	" "
Attr_defaultExit	" "

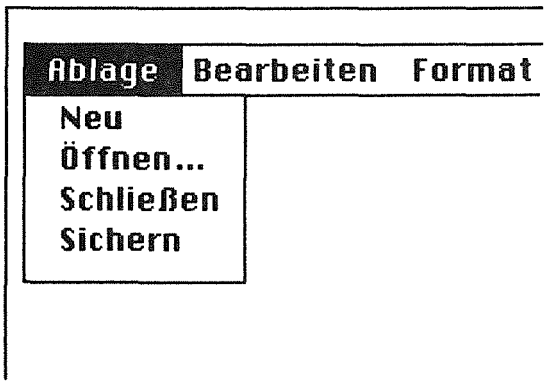
Menubox

Die Menubox stellt eine Ausnahme unter den Objektklassen dar. Sie kann Kind von jeder anderen Klasse sein und entsprechend ist ihre Darstellung und Funktionalität abhängig von der des Vaters. Ist der Vater ein Window, so wird jede Menubox als ein Menüpunkt in der Menüzeile des Windows angezeigt.

Durch Selektieren dieses Menüpunktes werden die entsprechenden Menuitems als Pull-Down-Menu angezeigt, aus denen dann ein Element ausgewählt werden kann.

Ist die Menubox dahingegen Kind eines anderen Objektes (Ausnahme Dialog), so ist die Menubox nicht sichtbar. Erst wenn der Benutzer das Objekt selektiert mit einer bestimmten Interaktion (rechte Maustaste zum Beispiel), wird die

mit einer bestimmten Interaktion (rechte Maustaste zum Beispiel), wird die Menubox mit ihren Einträgen sichtbar. Der Auswahlmechanismus bleibt davon unbeeinflußt.



Weiterhin unterscheiden sich Windows von allen anderen Objekten darin, daß sie als direkte Nachfolger mehrere Menuboxen haben können, wohingegen jedes andere Objekt nur eine Menubox als Kind besitzen kann. Da aber verschachtelte Menus erlaubt sind, kann man diese Menubox als Kontainer für weitere Menuboxen benutzen, um so die gleiche Funktionalität zu erhalten wie beim Window.

Wird Menubox als Kind eines Objekts der Klasse Edittext definiert und wird ein Eintrag aus diesem Menü ausgewählt, so wird der Inhalt dieses Eintrages in den Edittext übernommen.

Des weiteren verfügt die Menubox als einziges über das Attribut Attr_Menustyle. Dieses Attribut beschreibt das grundlegende Verhalten eines Menüs und seiner Menüeinträge. Dabei sind drei Möglichkeiten gegeben. Es kann sich wie eine Liste von zusammengehörigen Radiobuttons verhalten, wie mehrere Pushbuttons oder wie mehrere Checkboxen. Ist die Alternative Radiobuttons selektiert, erhalten die Menuseperatoren eine besondere Bedeutung. Dienten sie ursprünglich zur optischen Trennung der einzelnen Menupunkte, so grenzen sie nun eine Gruppe von zusammengehörigen Radiobuttons ein. Es können demnach mehrere solcher Gruppen innerhalb einer Menubox definiert werden. Von einer Gruppe kann maximal ein Eintrag selektiert sein.

Da die Menubox als Kontainer für die einzelnen Menüeinträge dient, hat sie keine Exit-Eigenschaften. Diese sind bei den einzelnen Menuitems zu definieren.

Attribut	Menubox
Attr_geoBehaviour	Geometrieverhalten
Attr_name	Name der Menübox
Attr_menuStyle	s.o.

Menuitem

Das Menuitem erbt die Geometrieeigenschaften seines Vaters (Menubox). In Abhängigkeit von dem Menustyle der Menubox kann ein einzelnes Menuitem als selektiert oder nicht selektiert gekennzeichnet sein (nur Radiobutton- und Checkboxstyle). Andererseits können nur die Menuitems einer Menubox mit Menustyle Pushbuttonstyle die Exit-Eigenschaften besitzen. Da ein Menü häufig den Befehl "Verlassen" o.ä. enthält, wenn es direkter Nachfahre eines Windows ist, ist es nur in diesem Fall möglich, die Default-Exit Eigenschaft an das Menuitem zu vergeben.

Attribut	Menuitem
Attr_name	Name des Menuitems
Attr_selected	Der Eintrag ist selektiert
Attr_exitFlag	s.o. (nur bei Menustyle = Pushbutton_Style)
Attr_exitFunc	s.o.
Attr_defaultExit	s.o.

Menusep

Der Menuseperator hat wie bereits erwähnt hauptsächlich optische Aufgaben. Er ist nicht selektierbar, verfügt demnach nur über ein Attribute. Da er dadurch eine Sonderstellung einnimmt, werden hier alle (1) Attribute beschrieben.

Attribut	Menusep
Attr_visible	Seperator ist sichtbar oder nicht.

Choose

Objekte der Klasse Choose werden in Abhängigkeit ihrer Attribute auf die Basisklassen Checkbox, Radiobutton, Menubox oder Listbox abgebildet. Sie werden zur Auswahl eines oder mehrer Elemente aus einer Auswahlliste, bestehend aus Chooseitems, verwendet. Die Attribute *Attr_showAll* und *Attr_multisel* sind nach der Definition des Objektes nicht mehr veränderbar.

Attribut	Choose
Attr_name	Name der Auswahl
Attr_geoBehaviour	Geometrieverhalten
Attr_showAll	Alle Chooseitems sollen angezeigt werden.
Attr_multisel	Es können mehrere Elemente ausgewählt werden.

Chooseitem

Ein Chooseitem stellt ein Element der Auswahlliste eines Objektes der Klasse Choose dar.

Attribut	Chooseitem
Attr_selected	Das Element ist selektiert
Attr_name	Anzuzeigender Text vom Typ DT_String
Attr_exitFlag	Aufgrund der Selektion bzw. Reselektion eines Chooseitems wird der Teildialog verlassen mit der in Attr_exitFunc angegebenen Funktion
Attr_exitFunc	s.o.

XMenu

Diese Objektklasse dient der Erzeugung von Menüs, wie sie in klassischen DV-Systemen Anwendung finden. Ein XMenu besteht aus einem Window, welches in vier Bereiche unterteilt ist :

- Allgemeine Informationen
- Auswahlbereich
- Pushbuttons

Im Auswahlbereich werden die einzelnen Menüpunkte dargestellt (XMenuitems). Alle als direkte Kinder des XMenu definierten Pushbuttons werden am unteren Rand des Windows plaziert (Pushbutton-Bereich). Durch "allgemeinen Informationen" kann eine Überschrift angegeben werden. Wird das Atttribut Attr_multisel gesetzt, so ist sicherzustellen, daß mindestens ein Pushbutton vorhanden ist.

Attribut	XMenu
Attr_geoBehaviour	Geometrieverhalten
Attr_alert	Andere Fenster können nicht bearbeitet werden.

Attr_multisel	Es können mehrere Menüeinträge selektiert werden.
Attr_name	allgemeine Informationen

XMenuitem

Attribut	XMenuitem
Attr_geoBehaviour	Geometrieverhalten
Attr_name	anzuzeigender Name des Objektes
Attr_selected	Der Menüeintrag ist ausgewählt.

Formular

Mit Hilfe der Objektklasse Formular wird ein Rahmen für die Erstellung komplexer Formulare angeboten. Es besteht aus einem Fenster, welches unterteilt ist in die Bereiche :

- Allgemeine Informationen (siehe XMenu)
- Arbeitsbereich
- Statusanzeige
- Pushbuttons (siehe XMenu)

Innerhalb des Arbeitsbereiches können beliebige Objekte plaziert werden, deren Hilfetext (*Attr_help*) bei deren Selektion in der Statusanzeige dargestellt wird. Die Statusanzeige ist oberhalb des Pushbutton-Bereiches plaziert.

Attribut	Formular
Attr_geoBehaviour	Geometrieverhalten
Attr_name	allgemeine Informationen
Attr_alert	Andere Fenster können nicht bearbeitet werden.

History

Das History-Objekt erlaubt die Umsetzung einer erweiterbaren Liste von Strings, die in der Reihenfolge ihrer Definition untereinander angeordnet werden.

Attribut	History
Attr_geoBehaviour	Geometrieverhalten
Attr_name	anzuzeigender Name des Objektes
Attr_content	Inhalt der History, kann durch Attr_appContent erweitert werden.

Table

Eine Tabelle besteht aus einer beliebigen Kombination von Zeilen und Spalten. Diese sind aufgeteilt in einen Definitionsteil und eine Liste von Werten. (siehe TabLine / TabCol)

Attribut	Table
Attr_geoBehaviour	Geometrieverhalten
Attr_name	anzuzeigender Name des Objektes

TabLine und TabCol

Stellt eine Zeile bzw. eine Spalte einer Tabelle dar. Die Attribute dieses Objektes gelten für alle Elemente der Zeile / Spalte. Die einzelnen Elemente der Tabelle können überschrieben werden.

Jede Zeile oder Spalte verfügt über folgende Attribute :

Attribut	TabLine / TabCol
Attr_name	anzuzeigender Name des Objektes
Attr_inputRequired	Es muß ein Wert eingegeben werden
Attr_cl_nr	Laufende Nummer der Zeile bzw. Spalte
Attr_delOnSelect	Inhalt ist zu löschen bei Selektion durch Benutzer
Attr_type	Datentyp des Inhaltes (Attr_content)
Attr_content	Inhalt des Zeile/Spalte; besteht aus einer Liste von Werten vom Datentyp Attr_type. Dieses Attribut muß bei der Definition als letztes angegeben werden !
Attr_constraint	Plausibilitätsüberprüfung für alle Werte der Zeile bzw. Spalte
Attr_format	Eingabeformat

Infobox

Eine Infobox ist eine Groupbox, die zwei statischen Texten als Kinder besitzt. Mit Hilfe des ersten wird eine Überschrift generiert und das zweite dient der Anzeige der eigentlichen Statusinformationen.

Attribut	Infobox
Attr_geoBehaviour	Geometrieverhalten
Attr_name	Überschrift
Attr_content	auszugebende Statusinformation

Infowindow

Das Infowindow dient der Nachrichtenanzeige. Mit Hilfe eines Windows wird der Nachrichtentext und eine Überschrift erzeugt. Per Default enthält es einen Pushbutton, mit dessen Hilfe der Benutzer das Fenster schließen und die Nachricht somit bestätigen kann. Der Name des Pushbuttons setzt sich zusammen aus dem Namen des Infowindows, gefolgt von "_PB" (Beispiel "Mein_Infowindow_PB"). Es ist darauf zu achten, daß dieser Name nicht zur Definition von anderen Objekten verwendet wird.

Attribut	Infowindow
Attr_geoBehaviour	Geometrieverhalten
Attr_alert	Andere Fenster können nicht bearbeitet werden.
Attr_name	Überschrift
Attr_content	Nachricht

Nachdem nun die Objektklassen und ihre Attribute beschrieben sind, soll der eigentliche Kern der Dialogbeschreibungssprache dargestellt werden, mit der Dialoge erzeugt werden können.

4.6.9 Befehlssatz / Methoden der Dialogbeschreibungssprache

Die Dialogbeschreibungssprache ist als Protokoll zur dialogbezogenen Kommunikation zwischen Anwendungsserver und Client definiert. Die ausgetauschten Nachrichten beziehen sich auf die einzelnen Dialogobjekte. Die Objektinstanz einer Anwendung kann mit Hilfe der Dialogbeschreibung seine Präsentation festlegen. Der Nachrichtenaustausch ist mit dem message-passing in einem

objektorientierten System vergleichbar. Andererseits kann mit Hilfe der Dialogbeschreibungssprache eine klassische Dialogsteuerung realisiert werden.

Im folgenden wird von Befehlen der Dialogbeschreibungssprache gesprochen. Wie gerade erwähnt, kann man diese auch als objektbezogene Methoden betrachten.

Der Befehlssatz der Dialogbeschreibungssprache besteht aus den Befehlsgruppen :

Erzeugen von einzelnen Objekten oder Objekthierarchien
Setzen von Objektattributen
Abfragen von einzelnen Objektattributen oder Objekthierarchien
Entfernen von Objekten

Die entsprechenden Befehle sind :

DCmd_Create
DCmd_SetObjectAttribute
DCmd_Hide
DCmd_HideChildren
DCmd_Show
DCmd_ShowChildren
DCmd_GetObject
DCmd_GetObjectAttribute
DCmd_Destroy
DCmd_DestroyChildren
DCmd_Initialize
DCmd_StopDialog

Die allgemeine Syntax von Befehlen ist dabei :

(<Befehl> <Parameter>)

Jeder Befehl ist begrenzt durch die Symbole "(" und ")". Dadurch ist es möglich, mehrere Befehle auf einmal zu erteilen (Liste von Befehlen) und diese auch als solche zu erkennen. Die Parameter eines jeden Befehls sind abhängig vom Befehl selber und werden an entsprechender Stelle erläutert.

Zur Erhöhung der Lesbarkeit sind zwischen die einzelnen Symbole Leerzeichen eingefügt und ist mit Einrückungen gearbeitet worden.
Allgemein verwendete Bezeichner werden im Anschluß an die Befehlsdefinitionen erläutert.

Nun zu den Befehlen im einzelnen.

DCmd_Create

Mit Hilfe des Create-Befehls können einzelne Objekte aber auch eine Hierarchie von Objekten erzeugt werden.

(DCmd_Create <Parent_ID> <x_Object_Defs>)

<x_Object_Defs> ::=

(<Object_ID> <Class_ID> <Attributliste>
[<x_Object_Defs>]) [<x_Object_Defs>] ;

<Attributliste> ::= { <Attribut_ID> <Value> };

Die "<Attribut_ID> <Value>" - Paare können in beliebiger Reihenfolge angegeben werden. Ausnahme bildet hierbei die Klasse "Table", deren Attribut Attr_content am Ende der Attributliste positioniert werden muß.

Jedes Objekt benötigt eine eindeutige Identifikation (<Object_ID>), die nicht dem Namen (Attr_name) des Objektes entsprechen muß. Mit Hilfe dieser Kennzeichnung können alle Objekte referenziert werden.

Einige der Dialogobjekte werden zur Erklärung oder zu Statusanzeigen verwendet. Nachdem sie erzeugt wurden, besteht kein Bedarf nach einem Bezug zu ihnen. Für solche Objekte muß die <Object_ID> nicht angegeben werden. Dies impliziert, daß jeglicher Bezug auf diese Objekte ausgeschlossen ist.

Zu jedem Objekt muß sein Vater (<Parent_ID>) angegeben werden. Bei der Definition eines Fensters ist die <Dialog_ID>, die bei der Initialisierung angegeben wurde (siehe DCmd_Initialize) als Vater zu verwenden.

Jedes Objekt ist einer bestimmten Klasse bzw. eines Modelles dieser Klasse zugehörig (<Class_ID>).

Mit Hilfe der <Attributliste> können bereits bei der Definition des Objektes Attributwerte belegt werden. Nicht angebene Attribute werden mit den Defaultwerten der Klasse belegt.

Die Definition von <x_Object_Defs> ist rekursiv. Die Klammerung hat nun nicht nur begrenzende Aufgaben, sondern kennzeichnet gleichfalls die Hierarchiezugehörigkeit des Objektes durch die aktuelle Klammerungstiefe. Jedes

Objekt einer tieferen Klammerungsebene ist Kind des Objektes der nächst höheren Ebene. Die explizite Angabe des Vaters ist nur einmal notwendig.

Es muß mindestens ein zu erzeugendes Objekt angegeben werden; das Erzeugen einer Hierarchie von Objekten ist optional. Es ist demnach auch möglich, durch eine Kette von einzelnen Create-Befehlen die gewünschte Struktur zu erzeugen.

Ein Create angewandt auf ein bereits existierendes Objekt führt zu einem Entfernen desselben gefolgt von der Neudefinition mittels der angegebenen Parameter. Das Entfernen bewirkt, daß auch alle Nachfahren des Objektes entfernt werden.

DCmd_SetObjectAttribute

Dieser Befehl erlaubt die nachträgliche Änderung von Objektattributen.

(DCmd_SetObjectAttribute <Object_ID> <Attributliste>)

Nicht angegebene Attribute bleiben unbeeinflußt.

DCmd_Hide

Ein "Verstecken" von Objekten vor dem Benutzer ist mittels dieses Befehls möglich. Es wird jeweils das Attribut Attr_visible und Attr_sensitive des Objektes und damit aller seiner Nachfahren auf FALSE gesetzt. Selbst wenn der Benutzer in der Lage ist, das Objekt wieder sichtbar zu machen, so kann er es dennoch nicht bearbeiten.

(DCmd_Hide <Object_ID>)

DCmd_HideChildren

Die Funktion dieses Befehles ist dieselbe wie die von DCmd_Hide. Jedoch bleibt das Objekt selber sichtbar; der Befehl bezieht sich nur auf seine Nachfahren.

(DCmd_HideChildren <Object_ID>)

DCmd_Show

Dieser Befehl hebt die Wirkung von DCmd_Hide auf. Das Objekt und alle seine Nachfahren werden für den Benuzter wieder zugänglich.

(DCmd_Show <Object_ID>)

DCmd_ShowChildren

Analog zu DCmd_Show bewirkt DCmd_ShowChildren, daß alle Nachfahren eines Objektes sichtbar werden. Ist das referenzierte Objekt allerdings nicht sichtbar gewesen, so hat dieser Befehl keine Wirkung.

(DCmd_ShowChildren <Object_ID>)

Mit Hilfe der Befehle DCmd_Hide, DCmd_Show bzw. DCmd_HideChildren und DCmd_ShowChildren ist es möglich, komplexe Dialoge zu definieren, ohne daß der Benutzer von dem eventuell zeitaufwendigen Vorgang Kenntnis bekommt, weil dieser verdeckt abläuft.

DCmd_GetObject

DCmd_GetObject erlaubt es, die Attribute des Objektes und aller seiner Nachfahren zu erfragen. Die Syntax der auf diesen Befehl eintreffenden Antwort ist dieselbe wie die des Befehles DCmd_Create.

(DCmd_GetObject <Object_ID>)

Die Antwort auf diesen Befehl hat die Form :

(DRes_Object <Object_ID> <Attributliste> [<x_Object_Attribs>])

<x_Object_Attribs> ::=

(<Object_ID> <Attributliste> [<x_Object_Attribs>])
[<x_Object_Attribs>];

DCmd_GetObjectAttribute

Zur Ermittlung von objektbezogenen (nicht hierarchiebezogenen) Informationen über Objektattribute dient der Befehl DCmd_GetObjectAttribute. Ihm wird eine Liste von Attributen mitgegeben, von denen die Attribute ermittelt werden sollen. Wird diese nicht angegeben, so werden nur die inhaltsrelevanten Objektattribute zurückgegeben (siehe in diesem Kapitel: Anbindung der Anwendung).

(DCmd_GetObjectAttribute <Object_ID> { <Attribut_ID> })

Als Antwort erfolgt :

(DRes_ObjectAttribute <Object_ID> <Attributliste>)

DCmd_Destroy

Für das Entfernen von Objekten ist dieser Befehl zuständig. Es werden das Objekt selber und alle seine Nachfahren aus der Dialoghierarchie gelöscht. Ein Referenzieren auf enfernte Objekte ist nicht mehr möglich.

Der Befehl wird sofort umgesetzt und alle noch laufenden Benutzerinteraktionen oder durch sie bewirkte Abläufe werden unterbrochen. Um Seiteneffekte zu vermeiden, sollten alle von dem Löschen betroffenen Objekte mittels DCmd_Hide bzw. DCmd_HideChildren zunächst deaktiviert werden. Das Objekt Dialog selber kann nicht entfernt werden. Hierfür dient der Befehl DCmd_StopDialog.

(DCmd_Destroy <Object_ID>)

DCmd_DestroyChildren

Um nur die Nachfahren eines Objektes zu entfernen, das Objekt selber aber beizubehalten, wird DCmd_DestroyChildren verwendet. Die im Befehl DCmd_Destroy ausgesprochene Warnung gilt hier analog.

(DCmd_DestroyChildren <Object_ID>)

DCmd_Initialize

Mit Hilfe dieses Befehls wird der Dialog beim Basissystem des Clients angemeldet und die entsprechenden Initialisierungsroutinen gestartet. Dieser Befehl ist die Basis für alle anderen und muß als erstes ausgeführt werden, bevor das Objekt Dialog referenziert werden kann. Dieses wird erst durch diesen Befehl beim Client erzeugt.

(DCmd_Initialize <Dialog_ID>)

DCmd_StopDialog

Analog zu DCmd_Initialize wird hiermit ein Dialog abgemeldet. Alle noch vorhandenen Objekte werden entfernt, eventuell noch ausstehende Antworten auf Befehle werden nicht mehr gesendet.

(DCmd_StopDialog <Dialog_ID>)

4.6.10 Anbindung der Anwendung

Die Koppelung zwischen Anwendung und Dialog wird durch das Attribut Attr_ExitFunc der Objekte vollzogen. Löst der Benutzer durch eine entspre-

chende Interaktion den Aufruf einer solchen Funktion aus, wird die Applikation über dieses Ereignis informiert. Wie detailliert die dabei zurückgelieferte Information ist, hängt von der Art der Exit-Funktion ab, die bei der Definition des Objektes angegeben wurde. Es werden bestimmte Funktionen von der Dialogschicht angeboten; es können aber ebenso anwendungsspezifische angegeben werden, die dem System bekannt sein müssen.

Bei der Angabe einer Exitfunktion kann bestimmt werden, ob der Dialog nach Auslösen der Exitfunktion angehalten werden soll oder ob der Benutzer weiterhin Interaktionen durchführen kann (<pauseDialog>). Mit diesem Parameter ist es auch klassischen DV-Systemen möglich, ihre Dialogsteuerung wahrzunehmen, ohne Anpassung an Ereignissteuerung u.ä. vornehmen zu müssen.

Die Exitfunktionen im einzelnen :

DCB_Event <pauseDialog>

Es wird nur die <Object_ID> des Objektes zurückgeliefert, welche die Exitfunktion ausgelöst hat. Das Format der Nachricht, die der Anwendung geschickt wird, ist :

(DRes_Event DCB_Event <Object_ID>)

DCB_Content <pauseDialog>

Der aktuelle Inhalt (Attr_content bzw. Attr_name) wird der Anwendung übermittelt.

(DRes_Event DCB_Content <Object_ID> <Content>)

DCB_GetObjectAttrib <Object_ID> <changed_only> <pauseDialog>

Die Attribute des angegebenen Objektes werden der Anwendung zugänglich gemacht (siehe zurückgelieferte Objektattribute). Wird <changed_only> gesetzt, so werden nur die Attributwerte gesendet, die sich seit der Definition bzw. derem letzten Ändern durch die Anwendung selber verändert haben. Die Antwort für die Anwendung :

(DRes_Event DCB_GetObjectAttrib <Object_ID> <Attributliste>)

DCB_GetObject <Object_ID> <changed_only> <pauseDialog>

Diese Exitfunktion erlaubt, die komplette Hierarchie von Objektattributwerten, die das Objekt <Object_ID> als Vater haben einschließlich des Vaters selbst, an die Anwendung zu übergeben.

(DRes_Event DCB_GetObject <Object_ID> <Attributliste>
[<x_Object_Attribs>])

<x_Object_Attribs> ::=

(<Object_ID> <Attributliste> [<x_Object_Attribs>])
[<x_Object_Attribs>];

In Abhängigkeit der einzelnen Objekte und deren Definition werden unterschiedliche Objektattribute an die Anwendung zurückgegeben, wenn eine Exitfunktion ausgelöst wird.

Zurückgelieferte Objektattribute

Mit Hilfe des Befehls DCmd_GetObjectAttribute und der Exitfunktionen DCB_GetObject sowie DCB_GetObjectAttrib können Attribute einzelner Objekte oder einer Hierarchie an die Anwendung übermittelt werden. Während beim erstgenannten Befehl die Angabe der erforderlichen Attribute möglich ist, sind die Exitfunktionen auf das Zurückliefern der inhaltsrelevanten Attribute beschränkt. Wird keine Attributliste dem Befehl DCmd_GetObjectAttribute beigefügt, so werden ebenfalls nur inhaltsrelevante Ausprägungen gesendet, welche in diesem Abschnitt erläutert werden.

Die inhaltsrelevanten Attribute gliedern sich in Standard- und objektbezogene, wobei erstgenannte für jedes Objekt definierte Objekt angegeben werden. Es ist dabei ein Spezialfall zu beachten : Wird bei der Definition eines Objektes keine Object_ID angegeben und soll das Objekt trotzdem (aufgrund einer Benutzerinteraktion) zurückgeliefert werden, so ist die Object_ID undefiniert. Wird beispielsweise in einer Listbox ein Element ausgewählt, dessen Kennzeichnung (Object_ID) bei der Objektdefinition nicht angegeben wurde, so ist die zurückgelieferte Object_ID nicht definiert (also ein beliebiger nullterminierter String).

Folgende Attribute werden immer angegeben (Standard-Attribute) :

Attribut	Erklärung
Object_ID	Kennzeichnung des Objektes.
Attr_exitFlag	gesetzt, falls die Exitfunktion dieses Objektes ausgelöst wurde.
Attr_userID	
Attr_userData	

Diese Liste wird um die objektbezogenden Attribute erweitert, die aus der folgenden Tabelle hervorgehen.

Objekt	Attribut	Inhalt bzw. Bemerkungen
Dialog		keine weiteren Angaben
Window		keine weiteren Angaben
Groupbox		keine weiteren Angaben
Poptext		Es wird nur das selektierte Poptextitem zurückgeliefert.
Poptextitem	Attr_name	Name des Items
Listbox	Attr_name	
Listboxitem	Attr_name	Es wird nur die Liste der selektierten Items angegeben. Nicht selektierte bleiben unberücksichtigt.
Pushbutton	Attr_name	Es wird jeweils nur der ausgewählte Pushbutton zurückgegeben. Alle anderen Pushbuttons, sofern nicht deren Exitfunktion verwendet wurde (Attr_DefaultExit) bleiben unberücksichtigt.
Radiobutton	Attr_name	Es wird nur der selektierte Radiobutton angegeben.
Checkbox	Attr_name	Es werden nur die selektierten Checkboxen angegeben.
Edittext	Attr_content	Inhalt des Edittext

Statictext		keine weiteren Angaben
Menubox		Die Object_ID wird nur angegeben, um die Zugehörigkeit zu den folgenden Einträgen herzustellen.
Menuitem	Attr_name Attr_selected	Name des Menüeintrages Status des Menueintrages (nur bei Checkbox_Style und Radiobutton_Style)
Menusep		keine weiteren Angaben
Choose		keine weiteren Angaben
Chooseitem	Attr_name	Es werden nur die selektierten Elemente der Auswahlliste angegeben.
XMenu		keine weiteren Angaben. Es werden nur die selektierten Elemente der Auswahlliste angegeben.
XMenuitem	Attr_name	Name des Menüeintrages, der selektiert wurde.
Formular		keine weiteren Angaben
History		keine weiteren Angaben
Table		keine weiteren Angaben
TabLine	Attr_content	Die Liste der Werte der Zeile wird angegeben.
TabCol		Die Liste der Werte der Spalte wird angegeben.
Infobox		keine weiteren Angaben.
Infowindow		Zusätzlich wird als Kind der Pushbutton zurückgeliefert, der im Infowindow definiert ist. (s.o.)

4.6.11 Erklärung der verwendeten Bezeichner

Zum Abschluß dieses Kapitels werden die Bezeichner erläutert, die in dem Befehlssatz verwendet werden, sofern dies nicht schon im Text geschehen ist.

Bezeichner	Datentyp	Beschreibung
<Object_ID>	DT_IDString	Eindeutige ID eines Objektes
<Parent_ID>	DT_IDString	Object_ID des Vaters des Objektes
<Dialog_ID>	DT_IDString	ID des Dialoges des Objektes
<Class_ID>	DT_IDString	Name der Klasse
<Attribut_ID>	DT_uInt	Kennzeichnung des Attributes
<Value>		Wert eines Attributes. Sein Typ ergibt sich aus dem Typ des Attributes.
<changed_only>	DT_Boolean	TRUE, wenn nur geänderte Attributwerte an die Anwendung zurückgeliefert werden sollen. FALSE, wenn alle Attributwerte zurückgeliefert werden sollen.
<pauseDialog>	DT_Boolean	TRUE, wenn der Dialog nach Auslösen der Exitfunktion angehalten werden soll.

4.7 Kontrollschicht

Die Kontrollschicht stellt die zentrale Systemkomponente innerhalb des Basissystemes dar. In Abhängigkeit des Integrationsgrades der Anwendungen in das UIS hat sie unterschiedlich hohen Anforderungen gerecht zu werden.

Da der Schwerpunkt dieser Arbeit bei der anwendungsorientierten Integration liegt, steht dieser Gesichtspunkt auch in diesem Kapitel über den Entwurf der Kontrollschicht im Vordergrund.

4.7.1 Anforderungen

Aufgabe der Kontrollschicht ist die Sicherstellung der Verbindung zwischen Anwendung und ihrer Benutzerschnittstelle. Sie hat im wesentlichen die Funktion eines Verteilers. Auf der Seite der Kommunikationsschicht benötigt sie daher eine ...

Verbindungsverwaltung

Dieser obliegt die Steuerung der Kommunikation zwischen Anwendung und der Kontrollschicht. Das beinhaltet Verbindungsaufbau, -abbau, Nachrichtenübertragung, Transaktionsverwaltung und Synchronisation. Im Vergleich zur Kommunikationsschicht hat die Kontrollschicht eher überwachenden Charakter.

Koppelung von Anwendung und Dialogschnittstelle

Ist eine Verbindung zwischen Anwendung und der Kontrollschicht aufgebaut, müssen dialogbezogene Nachrichten zwischen Dialogschnittstelle und Anwendung vermittelt werden. Da mehrere Anwendungen gleichzeitig mit dem UIS in Verbindung stehen können, ist eine logische Zuordnung zwischen einer Anwendung und ihrem Dialog notwendig.

Anwendungsverwaltung

Auf Seiten des Benutzers wird eine Anwendungsverwaltung benötigt, über die er mit den Anwendungen in Verbindung treten kann. Sie stellt des weiteren eine Übersicht über die verfügbaren Applikationen bereit. Um diese Funktionalität umsetzen zu können, werden eigene Dialoge benötigt, mit denen die diesbezügliche Kommunikation mit dem Benutzer durchgeführt werden kann.

Überwachungsaufgaben

Aus Kosten- und Verfügbarkeitsgründen soll gewährleistet werden, daß Verbindungen nicht unnötig aufrechterhalten bleiben und die Dienste der Anwendung nicht blockiert werden.

4.7.2 Architektur

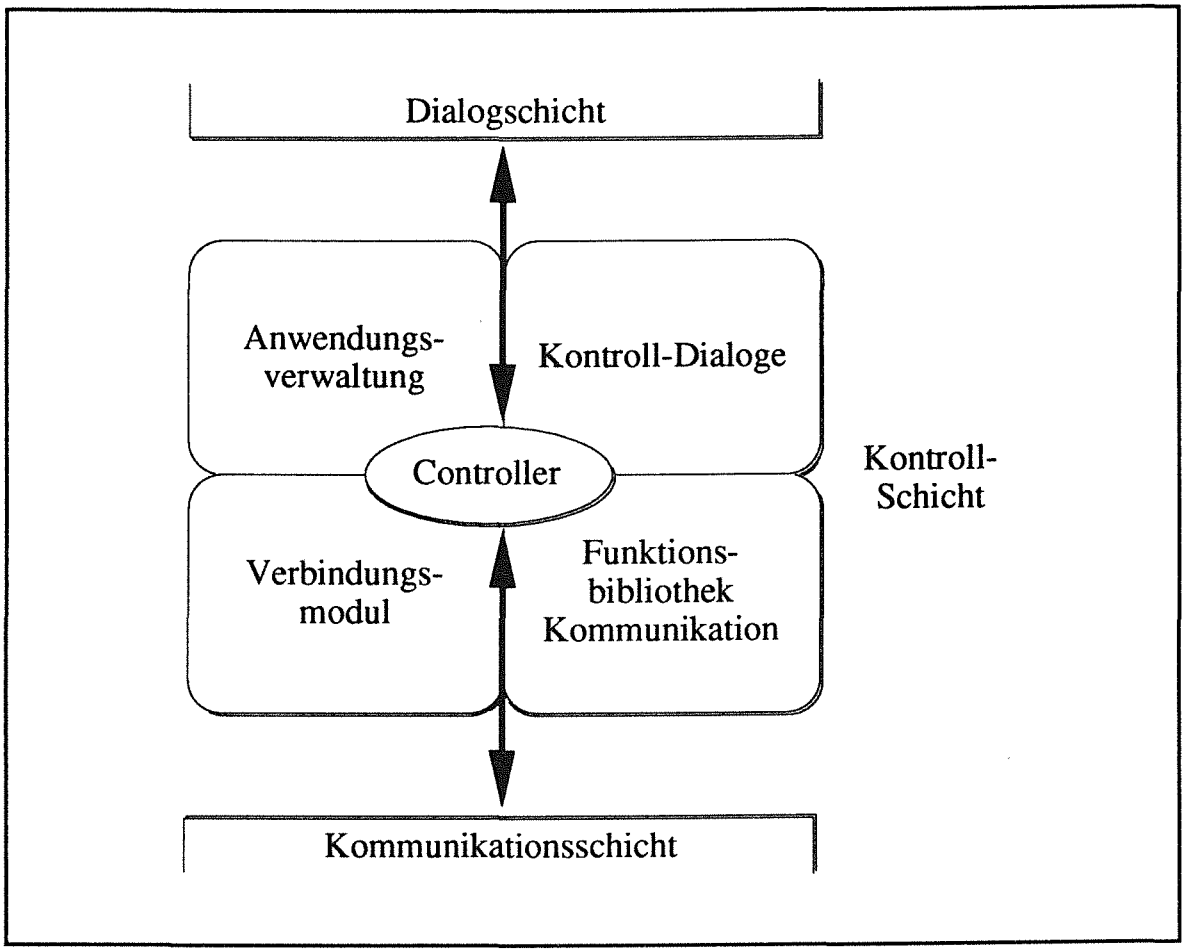


Abb. 4.21: Architektur der Kontrollschicht

Es folgt eine Beschreibung der Architektur der Kontrollschicht und der Zuständigkeiten ihrer Komponenten. (siehe Abbildung 4.21)

Anwendungsmodul

Das Anwendungsmodul (Anwendungsverwaltung) enthält Informationen über die Zugangspunkte der einzelnen Anwendungen im Netzwerk und eine allgemeine Beschreibung der Anwendung. Außerdem bietet sie die Möglichkeit, diese Daten zu warten. Letzteres wird mittels eigener Dialoge vorgenommen, die in der Dialogbeschreibungssprache spezifiziert sind.

Verbindungsmodul

Die Aufgabe des Verbindungsmoduls ist die Verwaltung der aktuellen Verbindungen zwischen den Anwendungen und der Dialogschicht. Für je eine dieser Beziehungen wird ein Eintrag in einer Tabelle vorgenommen, die die Zuordnung von Kommunikationsverbindung zu der Anwendung und ihrem zugehörigen Dialog ermöglicht. In dieser Tabelle können auch allgemeine Statusinformationen abgelegt werden, die der Überwachung dienen. Des weiteren werden verbindungsorientierte Daten über maximale Transaktionslänge u.ä. dort gespeichert.

Wird eine Verbindung über einen längeren Zeitraum nicht genutzt, so kann die Kontrollschicht eine Anfrage an den Benutzer stellen, ob die Anwendung (Verbindung) von ihm noch benötigt wird.

Controller

Dieses Modul ist der Kern der Kontrollschicht. Mit Hilfe des Verbindungsmodules wertet es Nachrichten der Kommunikationsschicht aus und weist sie den entsprechenden Dialogen der Dialogschicht zu. In umgekehrter Richtung ermittelt es aufgrund der Informationen der Dialogschicht die zugehörige Verbindung (Anwendung) aus und sendet ihr die entsprechende Nachricht der Dialogschicht.

Unterstützt wird der Controller durch eine Funktionsbibliothek für höhere Kommunikationsaufgaben zur Realisierung von Synchronisationsaufgaben, Transaktionsverwaltung und ähnlichem (**Funktionsbibliothek Kommunikation**).

Kontroll-Dialoge

Um die Bearbeitung von Daten der Anwendungsverwaltung durch den Benutzer bzw. den Systemverwalter zu ermöglichen, muß die Kontrollschicht über eigene Dialoge verfügen, die in dieser Komponente abgelegt sind. Verwendet wird dabei die Dialogbeschreibungssprache.

4.7.3. Koppelung der einzelnen Systemkomponenten

Die Kontrollschicht hat einerseits auf Ereignisse der Kommunikationsschicht zu reagieren und andererseits auf die der Dialogschicht. Beide Schichten müssen voneinander entkoppelt sein, um ihren Anforderungen gerecht zu werden. Demnach erfolgt die Verbindung der Kontrollschicht mit der Kommunikations- bzw. der Dialogschicht asynchron mittels Ereigniswarteschlangen. Um bevorzugte Nachrichten behandeln zu können, müssen diese Warteschlangen prioritätsgesteuert sein (zum Prinzip der Warteschlangen siehe [Wettstein 87, S.92-94], [Schmitt 83, S.43-60]).

5. Integration von XUMA in ein UIS

5.1 Ziele

Das Altlastenexpertensystem XUMA soll als Dienstleistung in einem offenen System verfügbar gemacht werden. Dabei steht die Integration in Umweltinformationssysteme im Vordergrund.

Eine mögliche Anwendergruppe von XUMA sind die Wasserwirtschaftsämter (WWA) des Landes Baden-Württemberg, denen XUMA zugänglich gemacht werden soll. Als Informationssystem wird in den WWAs die "Arbeitsdatei für wasser- und abfallwirtschaftliche Objekte" (KIWI) eingesetzt, die ein Bestandteil des landesweiten Umweltinformationssystems (UIS[1]) von Baden-Württemberg ist (siehe hierzu Kapitel 3 und [Bartl 86], [Mayer-Föll 89]).

Die für die Bewertung bzw. Beurteilung von Altlasten benötigten Falldaten sollen in KIWI abgelegt werden. Sie setzen sich aus den Ergebnissen der geologischen Untersuchungen und der historischen Erkundungen zusammen.

Es ist das Ziel, XUMA und das Informationssystem KIWI zu koppeln bzw. zu integrieren, so daß die mit der Altlastenbewertung beschäftigten Sachbearbeiter der Wasserwirtschaftsverwaltung in ihrer Tätigkeit unterstützt werden.

In den folgenden Abschnitten werden zunächst Anforderungen zusammengestellt, die sich im Zusammenhang mit der Integration von XUMA ergeben, sowie eine Analyse der Ausgangssituation vorgenommen. Im Anschluß wird das Integrationskonzept für XUMA vorgestellt.

---

[1] Begriffliche Problematik : UIS als Abkürzung von Umweltinformationssystem und als Kennzeichnung des Umweltinformationssystems des Landes Baden-Württemberg.

5.2 Anforderungen

Die in Kapitel 4 vorgestellten Anforderungen können ohne Modifikationen auf die gegebene Problematik übertragen werden, weshalb an dieser Stelle auf deren Wiederholung verzichtet wird.

XUMA als offenes System

Das Altlastenexpertensystem XUMA soll als eine Dienstleistung in einem offenen System einem möglichst großen Anwenderkreis zur Verfügung gestellt werden, sofern dieser über die notwendigen Schnittstellen und eine entsprechende Autorisierung verfügt.

Portierung von XUMA

Eine mögliche Portierung des Systems XUMA auf eine andere Hardware-Plattform ist zu berücksichtigen, so daß keine Einschränkungen der Verfügbarkeit für den Benutzer entstehen.

Verfügbarkeit der Falldatenhaltung aus KIWI für XUMA

Die von XUMA zur Bewertung benötigen Fall- und Analysendaten sind, wie bereits erwähnt, in dem Informationssystem KIWI abgelegt. Derzeit werden die für die Bearbeitung notwendigen Daten in XUMA per Hand eingegeben. Der Entwurf des Integrationskonzeptes soll eine Kopplung zwischen KIWI und XUMA in Bezug auf die Fall- und Analysendatenübertragung berücksichtigen.

Anforderungen an XUMA

Die Grundvoraussetzung für die anwendungsorientierte Integration von XUMA ist die Umsetzung und die Einhaltung der Schnittstellen für die Kommunikation mit dem UIS und den Dialog mit dem Benutzer (des UIS). Es ist bei der Schnittstellenrealisierung darauf zu achten, daß die Effizienz und die Verfügbarkeit des Systemverbundes XUMA-UIS nicht nachteilig beeinflußt wird.

5.3 Ist - Analyse

Bevor auf die Umsetzung der o.g. Anforderungen eingegangen wird, soll eine Systembeschreibung von XUMA und seinem zukünftigen Einsatzgebiet erfolgen.

XUMA

Das Expertensystem wird auf einem LISP-Rechner Explorer II von Texas Instruments mit der Expertensystem-Entwicklungsumgebung Inference ART und dem relationalen Datenbanksystem RTMS entwickelt. Letzteres ist speziell an die Programmiersprache LISP angepaßt und verfügt über keine Standard-SQL-Schnittstelle. Die Programme sind in ART und LISP implementiert, das Bereichswissen ist in Form von Regeln, Frames und Relationen repräsentiert.

Es sind derzeit zwei TI-Explorer II im Einsatz, die mit einer Hauptspeicherkapazität von 16 bzw. 8 MByte (virtuell max. 128 MByte) ausgestattet sind. Die Kommunikation zwischen den Explorern erfolgt über ein Local Area Network (LAN) mit Chaosnet, das auf dem Ethernet-Protokoll aufsetzt. Funktionen wie File Transer, Remote Access und Electronic Mail sind verfügbar. Desweiteren sind die TI-Explorer mit DECnet ausgerüstet, das den Anschluß an die Institutsrechner und das KfK-LAN ermöglicht. Über letzteres ist der Zugang zum Datex-P Netz der Deutschen Bundespost möglich.

Der TI-Explorer ist ein Single-User-, Multi-Tasking - System, dessen Systemressourcen jedoch durch XUMA weitestgehend beansprucht werden. Bei der Umsetzung des Integrationskonzeptes ist deshalb auf eine möglichst geringe zusätzliche Belastung des Systems zu achten.

Die Benutzerschnittstelle von XUMA basiert auf einem graphischen Fenstersystem mit Mausunterstützung. Die Dialogsteuerung ist rechnerinitiiert (vergleiche Kapitel 4.6.2) und durch eine hierarchische Menüstruktur geprägt. Neben den Menüs, die unter anderem der Auswahl der Kernfunktionen (Analysenplanerstellung, Beurteilung, Bewertung) dienen, wird die eigentliche Bearbeitung der Daten mit Hilfe von Formularen vorgenommen, die in Abhängigkeit des aktuellen Zustandes der Wissensbasis in Inhalt und Struktur variieren können.

Wasserwirtschaftsverwaltung Baden-Württemberg

Die Wasserwirtschaftsämter (WWA) sind mit identischer Hard- und Software ausgestattet, welche von der zuständigen DV-Abteilung des Umweltministeriums entwickelt und gewartet wird. Die derzeit verwendete Hardware-Plattform ist vom Typ MicroVAX 2 der Firma DEC und verfügt über 16 MByte Haupt-

speicher- und ca. 500 MByte Plattenspeicherkapazität, wobei als Betriebssystem VMS in der Version 5.4 verwendet wird. Der Zugang für den Benutzer zu diesen Systemen wird durch alphanumerische Terminals der Typen VT220, VT240, VT320, VT340 und VT420 über V.24 - Schnittstellen mit einer maximalen Übertragungsrate von 9600 baud ermöglicht. Mit Hilfe des Datex-P-Netzes und DECNet sind die einzelnen Systeme der WWAs untereinander und mit der DV-Stelle des Umweltministeriums verbunden.

In den WWAs kommen eine Vielzahl von Anwenderprogrammen zum Einsatz. Das Informationssystem KIWI ist fester Bestandteil dieses Software-Pools und ist mit Hilfe des relationalen Datenbanksystems ADABAS implementiert, wobei als Datenmanipulationssprache NATURAL bzw. SuperNATURAL eingesetzt wird. Letzt genannte wird auch für die Realisierung der menügeführten Benutzeroberfläche eingesetzt, in die die einzelnen Anwendungsprogramme eingebettet sind. Zwischen den Anwendungen selbst besteht jedoch keine Verbindung; sie operieren unabhängig voneinander auf eigenen Datenbeständen.

KIWI ist derzeit auf allen WWA-Systemen als lokales, nicht verteiltes System verfügbar. In regelmäßigen Abständen werden die Datenbestände durch die DV-Stelle des Umweltministeriums gewartet und aktualisiert. Um diesen Aufwand zu minimieren, ist der Einsatz eines verteilten, dezentralen Datenbanksystems, mit dessen Hilfe die WWAs untereinander gekoppelt werden sollen, geplant.

Die von XUMA benötigten Analysenergebnisse der Labors werden derzeit manuell in das KIWI-System eingegeben. Auch hier wird an einem automatisierten Verfahren mittels Datenträgertausch gearbeitet.

5.4 Integrationskonzept für XUMA

In diesem Abschnitt wird ein Integrationskonzept für XUMA vorgestellt, welches die Anforderungen und die Ist-Analyse berücksichtigt.

Integration in den WWAs

In den einzelnen WWAs werden Softwaresysteme eingerichtet, die das Basissystem (siehe Kapitel 4) umsetzen. Eine Kopplung mit den vorhandenen Applikationen ist nicht möglich, da diese abgeschlossene Systeme darstellen. Allerdings ist das Data Dictionary von KIWI zugänglich und somit ist ein Zugriff mit Hilfe der Datenmanipulationssprache SuperNATURAL auf die Falldaten möglich. Dahingegen ist die Nutzung der Benutzeroberfläche, die SuperNATURAL anbietet, nicht gegeben, da die entsprechenden Routinen nicht verfügbar sind.

5.4.1 Client-Server-Host - Modell

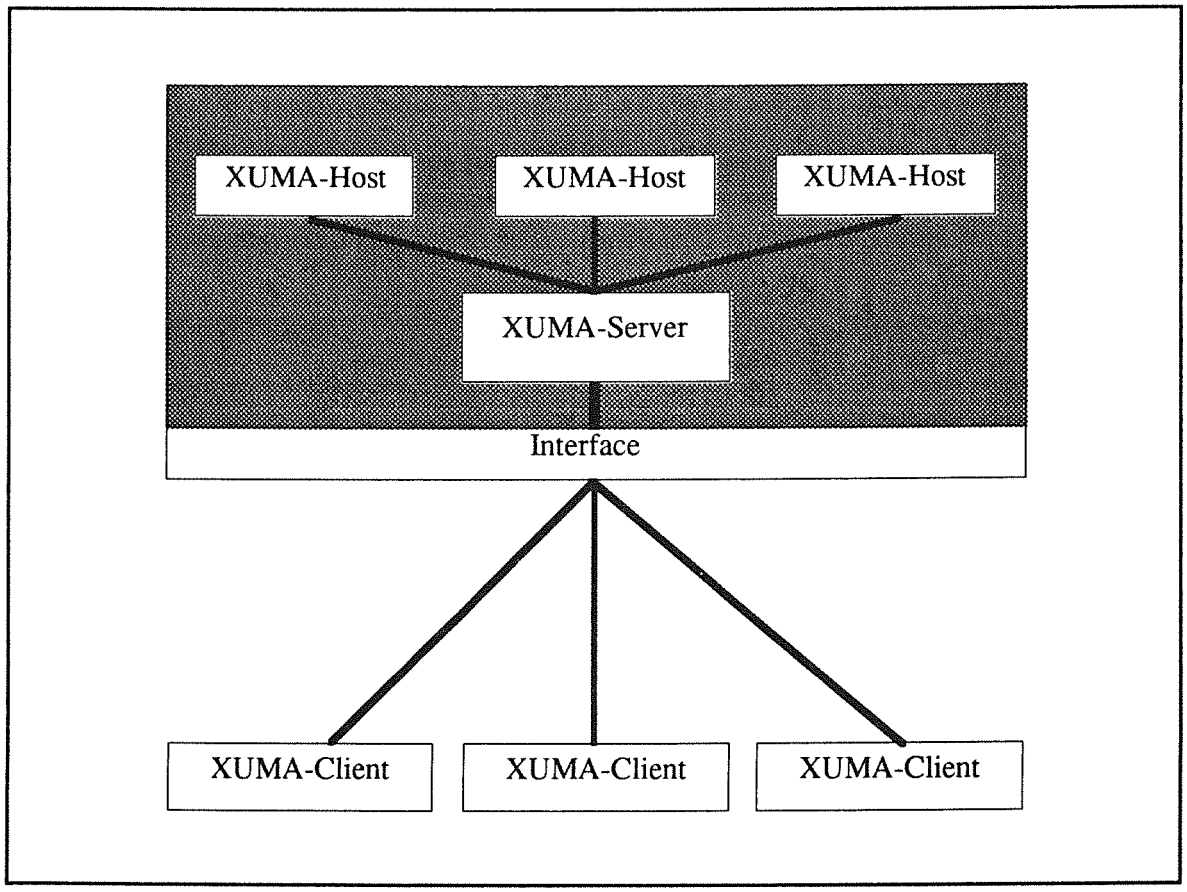


Abb. 5.1: Client-Server-Host Modell

Das Integrationskonzept sieht drei Komponenten vor : den Client, den Server und den Host (siehe Abbildung 5.1). Unter den Begriffen Client und Server sollen wie bisher das UIS (Client, KIWI) und der Anwendungsserver (XUMA-Server) verstanden werden. Der XUMA-Server stellt die Schnittstellen Dialog und Kommunikation für die Integration in das UIS zur Verfügung und wird mit Hilfe eines Vorrechners realisiert. (Dieser Rechner kann eine mögliche Plattform für die Portierung sein.) XUMA selbst bleibt ohne große Modifikationen als Stand-Alone-System auf dem TI-Explorer bestehen (XUMA-Host). Mit Hilfe einer speziellen Netzschnittstelle werden der Explorer und der Vorrechner gekoppelt, so daß eine möglichst geringe Belastung für den XUMA-Host entsteht (siehe Ist-Analyse).
Wie bereits erwähnt wurde, ist der TI-Explorer ein Single-User-System. Um die Verfügbarkeit von XUMA zu erhöhen, ist es wünschenswert, daß das System für mehrere Benutzer gleichzeitig zugänglich ist. Mit einem einzelnen Explorer ist dieses Vorhaben aufgrund der Auslastung der Ressourcen durch XUMA nicht realisierbar (siehe Ist-Analyse), weshalb die Möglichkeit bestehen soll, mehrere Explorer einzusetzen (siehe Abbildung 5.1).

Portierung von XUMA

Das Client-Server-Host-Modell unterstützt die Portierung von XUMA auf eine andere Hardware-Plattform. Der XUMA-Server und die XUMA-Hosts bilden eine "black box" (siehe Abbildung 5.1), deren Schnittstelle zum Client unverändert bleibt. Damit können ohne Einschränkungen der Verfügbarkeit Teilfunktionen vom XUMA-Host auf den XUMA-Server übertragen werden.

In den folgenden Abschnitten wird auf die Komponenten XUMA-Host und XUMA-Server eingegangen; Aufbau und Funktionsweise des Client wurden in Kapitel 4 bereits ausführlich behandelt.

XUMA-Server

In der Abbildung 5.2 wird der Aufbau und die Struktur der einzelnen Komponenten und ihre Verbindungen zueinander dargestellt. Die einzelnen Clients kommunizieren dabei ausschließlich mit dem XUMA-Server; für sie ist nicht ersichtlich, mit welchem XUMA-Host sie verbunden sind. Der XUMA-Server hat die Aufgabe, die einzelnen Verbindungen zwischen den Clients und den Hosts herzustellen, zu überwachen und nach Beendigung wieder abzubauen. Des weiteren wandelt er die vom XUMA-Host kommenden Dialogbeschreibungen, die in einem speziellen Format vorliegen, in die Dialogbeschreibungssprache um (siehe hierzu Abschnitt 5.4.2 und 4.6.3) und verfährt dementsprechend mit den Dialogbeschreibungen, die vom Client kommen. Desweiteren soll der XUMA-Server eine Warteliste führen, in der Benutzer verwaltet werden, die derzeit aufgrund ausgelasteter Ressourcen keinen Zugang zu einem XUMA-Host erhalten können.

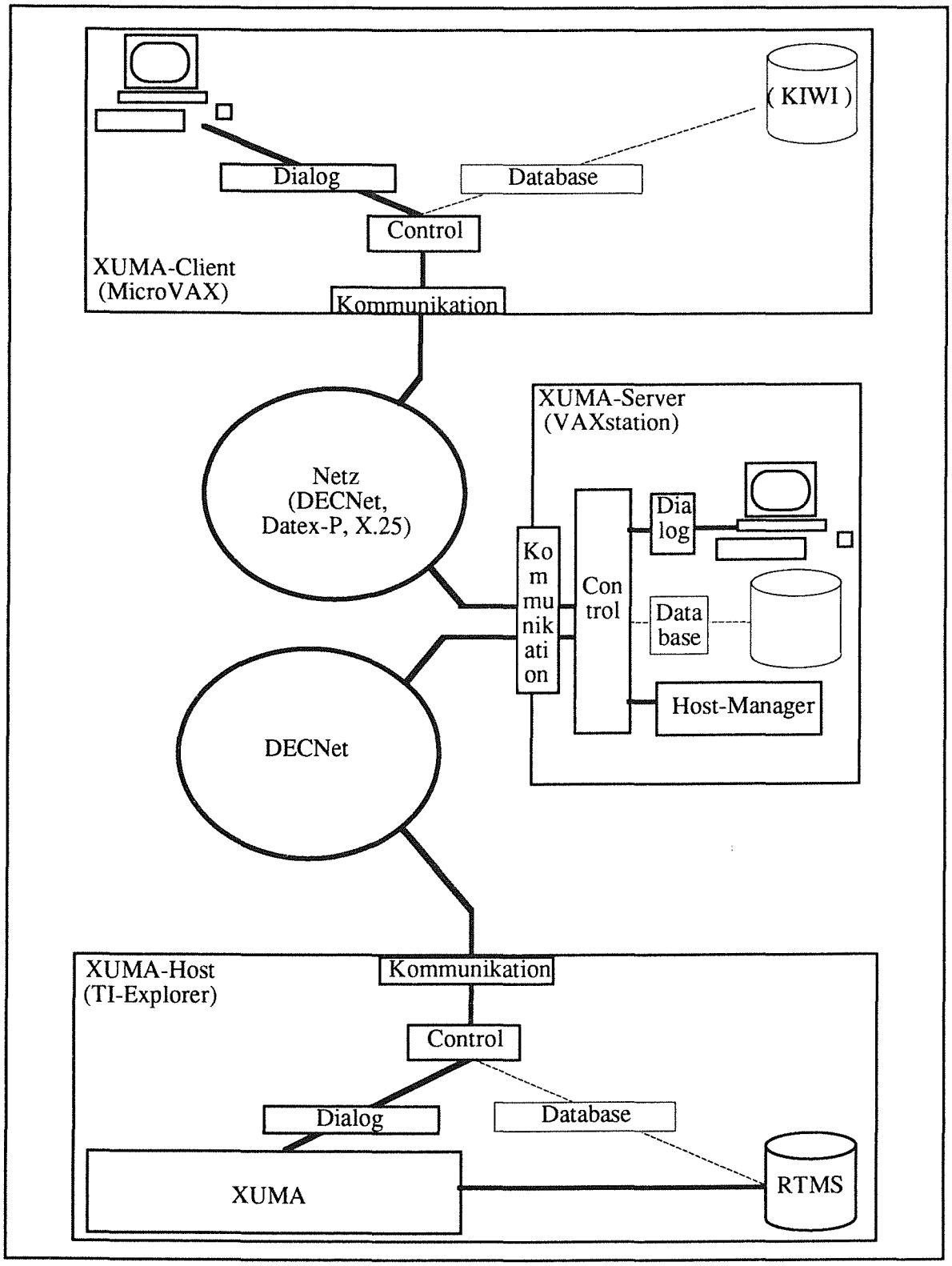


Abb. 5.2: Aufbau und Verbindung der Komponenten Server, Client und Host

Wird ein Host freigegeben, so erhält der am längsten wartende Benutzer diesen zugeteilt, unter der Voraussetzung, daß die Verbindung für die Dauer des Wartens aufrechterhalten wurde. Beschränkungen der Länge der Warteliste ergeben sich aus der maximalen Anzahl gleichzeitig aufgebauter Verbindungen, die der XUMA-Server verwalten kann bzw. ihm durch das Betriebssystem vorgegeben werden. Bricht ein wartender Benutzer die Verbindung ab, so wird er aus der Warteliste entfernt; ein Rückruf erfolgt nicht.

Neben diesen Transformations- und Überwachungsaufgaben, die in der Abbildung 5.2 in den Zuständigkeitsbereich des Host-Managers fallen, soll der XUMA-Server für die Zwischenspeicherung der Falldaten eingesetzt werden, wobei nur eine Falldatenkopie von KIWI angelegt wird, die für die zügige Bearbeitung eines Falles durch XUMA benötigt werden. Der hierdurch entstehende Aufwand und die Redundanz sind gerechtfertigt: Je Fall werden ca. 0,5 bis 1 MByte Daten benötigt, die für jede Bewertung bzw. Beurteilung eines Falles angefordert werden müssen. In der Regel konsultiert ein Sachbearbeiter das System für die Bearbeitung eines Falles mehrfach, so daß die Übertragung der Falldaten bei jeder Konsultation notwendig wäre.

XUMA erbringt nur den Dienst der Bewertung bzw. Beurteilung und ist nicht für die Bearbeitung der Falldaten einzusetzen; dies ist in KIWI durchzuführen.

Von einer ständigen Speicherung der Falldaten beim XUMA-Host ist abzusehen, da der Benutzer bei verschiedenen Konsultationen unterschiedliche Hosts zugewiesen bekommen kann, womit ein weiterer Transfer zwischen den Hosts notwendig wäre, der sich negativ auf die Performance der einzelnen Systeme auswirkt.

Zu Beginn einer Sitzung mit XUMA wählt der Benutzer den zu bearbeitenden Fall aus. Da die Falldaten beim Client bzw. bereis beim XUMA-Server abgelegt sind, ist eine diesbezügliche Verständigung zwischen Host und Server notwendig. Die Fallauswahl kann aber alternativ zum XUMA-Server verlagert werden, der bereits für die Falldatenübertragung vom Client über den Server zum Host zuständig ist, wodurch der XUMA-Host entlastet würde.
Um einen unnötig großen Falldatenbestand ("Datenfriedhof") zu vermeiden, werden die Falldaten nur über einen bestimmten Zeitraum hinweg beim XUMA-Server gespeichert.

Es ist sicherzustellen, daß der Benutzer immer mit den aktuellsten Daten arbeitet. Werden Änderungen an KIWI-Falldaten vorgenommen, welche auch auf dem XUMA-Server abgelegt sind, so sind letztere entsprechend zu aktualisieren.

XUMA-Host

Der XUMA-Host hat neben der Erbringung der eigentlichen Dienstleistung XUMA keine weiteren Aufgaben für die Umsetzung des Integrationskonzeptes zu erfüllen. Mit Hilfe zweier Schnittstellen (Kommunikations- und Dialog-) wird er in den Systemverbund Host-Server eingebunden. Die in Abbildung 5.2 dargestellte Control-Komponente hat die Aufgabe, den Datentransfer vom Server zum Host zu gewährleisten, ohne daß davon die eigentliche Anwendung XUMA betroffen wird.

Im Kapitel 6 wird auf die spezifischen Schnittstellenproblematiken innerhalb des XUMA-Systemverbundes eingegangen.

5.4.2 Analyse des XUMA-Dialog-Interface

Die Dialogbeschreibung des XUMA-Host liegt in einem speziellen, LISP-verwandten Format vor und ist durch spezifische Dialogtypen geprägt, die bei XUMA zum Einsatz kommen. In diesem Abschnitt wird das XUMA-Dialog-Interface (XUMA-DIF) auf die Dialogstrukturen hin analysiert und die Syntax (Protokoll) der Dialogbeschreibungen beschrieben.

Das XUMA-DIF unterscheidet folgende Dialogtypen:

- Status - Dialog
- Textausgabe - Dialog
- Info - Dialog
- Auswahlmenü - Dialog
- Mehrfachauswahlmenü - Dialog
- Formular - Dialog

XUMA erwartet in Abhängigkeit des in der Dialogbeschreibung verwendeten Dialogtyps eine bestimmte Antwort vom Client. Jede der nun folgenden Schnittstellen der Dialogtypen gliedert sich in einen *Export* - und einen *Import* - Teil. Unter Export wird die von XUMA ausgegebene Dialogbeschreibung verstanden und unter Import die erwartete Antwort des Client.

Status - Dialog

Der Status-Dialog dient der Anzeige von Statusinformationen über XUMA.

Schnittstelle :

Export

(:status
:art
:version | :benutzer | :fall | :funktion | :system
:text
<Liste von Strings> | nil)

Import

Es wird keine Rückgabe erwartet.

Dabei werden die Informationen bestimmten Gruppen zugeteilt :

:version
aktuelle Version des XUMA-Systems

:benutzer
Angaben über Benutzername und Benutzergruppe

:fall
Kennzeichnung des aktuellen Falles (Name, Standort)

:funktion
Zur Zeit aktive Funktion innerhalb der XUMA-Anwendung (Bewertung, Beurteilung, ...)

:system
Aktueller Zustand des Systems (Falldaten laden, Initialisierung, ...)

Textausgabe - Dialog

Mit Hilfe des Textausgabe-Dialoges können dem Benutzer umfangreiche Informationen gegeben werden. Sie werden in einer Liste (History) solange gespeichert, bis sie durch einen bestimmten Befehl (:output :clear) gelöscht werden.

Schnittstelle

Export

(:output
:clear)

(:output
:text <Liste von Strings>
[:header <String>])

Import

Es wird keine Rückgabe erwartet.

Bezeichner

:header
bezeichnet den Titel der Informationen.

Auswahlmenü - Dialog

Klassische Menüs können durch die Verwendung des Auswahlmenü-Dialoges realisiert werden. Er besteht aus einer Überschrift und einer Liste von Auswahlmöglichkeiten, aus denen der Benutzer eine einzelne selektieren kann. Die Elemente der Liste müssen nicht alle selektierbar sein, sondern können auch zur Formatierung verwendet werden. Als Antwort erwartet XUMA das ausgewählte Element und eine Information, wie die Selektion durchgeführt wurde (welche Maustaste gedrückt wurde).

Belegung der Maustasten und deren Bedeutung :

left :	Selektion
middle :	Abbrechen bzw. Zurück
right :	Anforderung von Hilfe zu einem Dialogobjekt

Schnittstelle

Export

(:single-menu
:header <String>
:items <Item-Liste>
:middle?
t | nil)

Import

(:single-menu
:item <Item-Value>
:button
:left | :middle | :right)

Bezeichner

```
<Item> ::=
      <String>                                          |     /* Type 1 */
      ( <String>   :value <Item-Value>
                   [ :highlighted <boolscher Wert> ]
                   [ :noselect <boolscher Wert> ]);          /* Typ 2 */
```

Typ 1 und Typ 2 dürfen nicht in einer gemeinsamen Item-Liste auftreten.

```
<Item-Liste> ::=
      ( <Item> { <Item> } )

<Item-Value> ::=
      <String>    |        /* bei Item vom Typ 1 */
      <Symbol>;            /* bei Item vom Typ 2 */

<boolscher Wert> ::= T | NIL;
```

:highlighted
Ein Item ist vorgeschlagen.

:noselect
Ein Element kann nicht ausgewählt werden.

Info - Dialog

Wird zu einem Dialog Hilfe angefordert, so kann diese durch einen Info-Dialog dargestellt werden. Der Benutzer muß das angezeigte Hilfefenster explizit schließen, um weiterarbeiten zu können, wofür XUMA eine Bestätigung erwartet.

Schnittstelle

Export

```
(:info
      :header      <String>
      :text  <Liste von Strings> )
```

Import

```
(:info   :ende )        /* Quittung */
```

Mehrfachauswahlmenü - Dialog

Das Mehrfachauswahlmenü ist eine Erweiterung des Auswahlmenüs, in dem mehrere Elemente der Auswahlliste selektiert werden können. Am unteren Rand des dargestellten Fensters werden sogenannte Margin-Choices angezeigt, die mit Pushbuttons vergleichbar sind. Mit ihnen kann der Benutzer beispielsweise den Dialog beenden oder weitere Informationen anfordern.
XUMA erwartet als Antwort die ausgewählten Listenelemente und das selektierte Margin-Choice.

Schnittstelle

Export

```
( :multiple-menu
      :header                 <String>
      :items                  <Item-Liste>
      :margin-choices         <Liste von margin-choices>
      :highlighted-items   <Item-Liste> )
```

Import

```
( :multiple-menu
      :items       <Liste von <Item-Value>>
      :choice      <margin-choice-value> | :right
      :info-item   <Item-Value> | <margin-choice-value>  )
```

Bezeichner

```
<margin-choice> ::=
      ( <String> :value <Symbol> | <Keyword> )

<margin-choice-value> ::=
      <Symbol> |          /* wenn <Symbol> hinter :value stand */
      <Keyword>           /* wenn <Keyword> hinter :value stand */
```

:choice
Wert einer Margin-Choice, die zum Verlassen des Dialoges selektiert wurde.

:info-item
zu diesem Dialogobjekt wurde Hilfe angefordert.

Formular - Dialog

Für die Erstellung komplexer Formulare wird der Formular-Dialog verwendet, mit dessen Hilfe der Benutzer Daten bearbeiten kann. An XUMA werden nach Abschluß dieses Dialoges alle geänderten Werte sowie das selektierte Margin-Choice zurückgeliefert.

Schnittstelle (Formular)

Export

```
( :cvv
      :header <String>
      :cvv-items <Liste von cvv-Items>
      :cvv-margin-choices <Liste von Strings> )
```

Import

```
( :cvv
      :cvv-items ( ... ( <Name als Symbol> :value <Wert> ) ... )
      :choice <String> )
```

Bezeichner

Anmerkung : cvv (choose variable values) ist die interne Bezeichnung des Formular-Dialoges.

cvv-Item

Folgende Typen sind möglich :

1. Textausgabe, nicht selektierbar :
   <String>

2. Einfache Felder :

```
( <Name als Symbol>  :value <Wert>
                     :label <String>
                     [ :constraint <constraint-function> ]
                     [ :documentation <String> ]
                     [ :edit ]
                     <Type-Spez.> )
```

3. Tabellen
(<Name als Symbol> :value <Liste von Werten>
{ <Spaltendefinition> })

Je Tabelle sind 1 bis n Spalten möglich, je Spalte sind 0 bis 2 Modifier erlaubt.

Erläuterungen :

:value
bezeichnet den Wert (einfaches Feld) oder eine Werteliste (Tabelle)

:label
ist die Beschreibung des Feldes für den Benutzer

:constraint
bezeichnet eine Art der Wertebereichsüberprüfung

:documentation
eine zusätzliche Information für den Benutzer, um ihn über den erwarteten Wert zu informieren

:edit
legt fest, ob das Feld zu löschen ist, wenn es aktiviert wird.

<Spaltendefinition> =
(<Lfd. Nr.; Integer> <Überschrift; String>
[:constraint <constraint-function>]
[:documentation <String>]
[:edit]
<Type-Spez.>);

spezielle Datentypen von XUMA:

<Type-Spez.> ::=
:choose <Liste von Strings> |
:set <Liste von Strings> |
:menu <cvv-Menu-Item-Liste> |
:multiple-menu <cvv-Menu-Item-Liste> |
:string |
:xuma-numeric |
:xuma-free |
:xuma-fix-or-free <Werte-Liste> |

```
    :xuma-datum                                    |
    :zahl-oder-string;

<cvv-Menu-Item-Liste> ::=
    ( <cvv-Menu-Item> { <cvv-Menu-Item> } );

<cvv-Menu-Item> ::=
    <String>                                       |
    ( <String>  :value <Item-Value>
                [ :highlighted <boolscher Wert>]
                [ :noselect <boolscher Wert> ]     |
                [ :menu-choose <Liste von Strings> ]);
```

:choose
Auswahl eines Elementes aus einer Liste, die vollständig dargestellt wird

:set
Auswahl mehrerer Elemente aus einer Liste, die vollständig dargestellt wird.

:menu
Pop-up Menü, Auswahl eines Elementes. Wird das cvv-Item nicht vom Benutzer bearbeitet, so ist nur das ausgewählte Element sichtbar.

:multiple-menu
Pop-up Menü, Auswahl mehrerer Elemente. Es werden alle selektierten Elemente dargestellt, wenn das cvv-Item nicht bearbeitet wird.

:menu-choose
Ein Wert eines :menu oder :multiple-menu kann sich aus einer weiteren Auswahlliste bestehend aus Strings ergeben.

:string
beliebiger Text

```
:xuma-numeric ::=              /* Grundtyp :string */
    ["< "] <Zahl>          |
    "nicht bestimmt"       |
    "nicht bestimmbar"     |
    "nicht nachweisbar";
```

```
:xuma-free ::=                  /* Grundtyp :string */
    "nicht bestimmt"            |
    "nicht bestimmbar"          |
    "nicht nachweisbar"         |
    <beliebiger Text>;

:xuma-fix-or-free ::=           /* Grundtyp :string */
    <String aus einer Liste von Strings>    |
    "nicht bestimmt"                        |
    "nicht bestimmbar"                      |
    "nicht nachweisbar"                     |
    <beliebiger Text> ;

:xuma-datum                     Grundtyp :string
    <leerer String> |
    <korrektes Datum>;

:zahl-oder-string               Grundtyp :string
    <Wert>          |
    <leerer String>;
```

"nicht bestimmt" kann mit "n.b.", "nb", "n.b", "nb." abgekürzt werden; "nicht bestimmbar" mittels "n.bbar.", "n.bar." und "nicht nachweisbar" durch "n.n.", "n.n", "nn.", "nn".

Constraint-Funktionen (Plausibilitätsüberprüfung)

fall-check-lfd-nr
Eingabe ist Leerstring oder Ganzzahl größer Null

para-number
Eingabe ist Leerstring oder Ganzzahl größer gleich 0

ae-check-lfd-nr
Eingabe ist Ganzzahl größer als Null

check-methode
Eingabe ist Ganzzahl größer 1 und kleiner 999

Es ist die Aufgabe des XUMA-Servers, die in dem oben vorgestellten Format vorliegenden Dialogbeschreibungen in die Dialogbeschreibungssprache umzusetzen. Vergleicht man die XUMA-DIF und die Dialogbeschreibungssprache, so ist eine problemlose Abbildung möglich, da die DiBeSp eine Obermenge der

XUMA-DIF darstellt. Auf die eigentliche Transformation wird an dieser Stelle nicht näher eingegangen (siehe Kapitel 6. Realisierung).

5.4.3 Kommunikationsschnittstelle

Die Kommunikationsschnittstelle zum XUMA-Server besitzt eine ausgesprochen einfache Struktur. Sie besteht im wesentlichen aus zwei Variablen, einer für die zu sendenden und einer für die zu empfangenden Nachrichten. Mit Hilfe eines Semaphor-Mechanismus wird sichergestellt, daß keine Nachricht verloren geht, bevor sie nicht entweder gelesen oder gesendet wurde.

6. Realisierung

6.1 Zielsetzung

Das in den Kapiteln 4 und 5 vorgestellte Integrationskonzept soll mit Hilfe einer prototypischen Teilimplementierung auf seine Durchführbarkeit hin untersucht werden. Der Schwerpunkt der Arbeiten liegt im Bereich Dialogschnittstelle, wobei folgende Anforderungen bzw. Einschränkungen vorliegen:

- Es wird nur eine Verbindung vom XUMA-Host über den -Server zum -Client betrachtet.
- Umfang des Client-Systems:
  1. Prototypische Implementierung der Dialogschicht mit Schwerpunkt auf der Umsetzung der für XUMA spezifischen Dialogtypen.
  2. Realisierung der Präsentationsschicht (OSF/Motif), soweit wegen 1. erforderlich.
- Umfang des XUMA-Server
  3. Umsetzung des XUMA-DIF in die Dialogbeschreibungssprache und umgekehrt.
  4. Verwaltung eines Hosts und eines Clients
5. Realisierung der Grundkomponenten für die Kopplung der einzelnen Schichten.

Bei der Realisierung dieser Anforderungen ist darauf zu achten, daß der Prototyp entsprechend für einen späteren Ausbau zur Umsetzung des vollständigen Basissystems ausgelegt ist.

Nachdem die Spezifikation im Detail vorlag wurde untersucht, welches Produkt (Werkzeug) für die Umsetzung des Vorhaben eingesetzt werden kann. Als Ergebnis dieser Untersuchung wird der "Dialogmanager" der Firma ISA GmbH, Stuttgart [Raether 90], vorgeschlagen, der sich vor allem durch die Vielzahl unterstützter GUIs (Graphical User Interface) auszeichnet. Nach einer ersten

Analyse in Form einer Präsentation wurde eine Testinstallation des ISA-Dialogmanagers[1] (ISA-DM) eingeleitet. Dieses Produkt deckt in seiner Funktionalität einen Teil der Dialogschicht des Basissystems ab; des weiteren bietet die Firma ISA GmbH ein Fenstersystem für alphanumerische Terminals an, welches über eine Schnittstelle zum ISA-DM verfügt. Mit Hilfe dieser Werkzeuge erschien die Realisierung der Integration von XUMA in einer wesentlich kürzeren Zeit durchführbar, als ursprünglich angenommen wurde. Zunächst galt es allerdings, die Eignung des ISA-DMs für das vorgesehene Einsatzgebiet zu überprüfen, d.h. inwieweit eine Einbindung in das Integrationskonzept möglich ist, und den hierfür notwendigen Aufwand abzuschätzen.

Um eine unnötige Implementierung von Funktionalitäten zu vermeiden, die eventuell durch den ISA-DM bereits gegeben sind, wurde die Zielsetzung des Realisierungsteils der vorliegenden Arbeit modifiziert:

- Die Präsentationsschicht wird durch den ISA-DM abgedeckt; deren Umsetzung entfällt somit.
- Die im Rahmen des XUMA-Servers anfallenden Arbeiten werden durch Mitarbeiter des Projektteams übernommen.
- Alle anderen Punkte bleiben Bestandteil der Zielsetzung und werden erweitert durch :

  1. Analyse des ISA-DM auf seine Eignung für das vorliegende Integrationskonzept.
  2. Vornehmen der notwendigen Änderungen am Basissystem (Dialogschicht), ohne die Spezifikationen der Schnittstellen zu verändern und unter Berücksichtigung aller dem Integrationskonzept zugrundeliegenden Anforderung, besonders der der System- bzw. Herstellerunabhängigkeit. (Die Verwendung eines bestimmten Werkzeuges soll nicht zu einer Voraussetzung für die Integration von XUMA werden.)
  3. prototypische Umsetzung der Dialogbeschreibungssprache mit Hilfe des ISA-DMs und Entwicklung gegebenfalls notwendiger Ergänzungen.

[1] Der Begriff "Dialogmanager" wurde in dieser Arbeit bereits in einem allgemeineren Kontext eingeführt. Es soll daher im folgenden "ISA-Dialogmanager" bzw. "ISA-DM" für den Dialogmanager der Firma ISA GmbH verwendet werden, sofern dies nicht unmittelbar aus dem Zusammenhang ersichtlich ist.

6.2 Analyse des ISA - Dialogmanagers

6.2.1 Positionierung des ISA-Dialogmanagers

Der ISA-Dialogmanager ist ein Werkzeug, das die getrennte Entwicklung von Applikation und Benutzerschnittstelle ermöglicht. Es basiert auf dem in Kapitel 4.5.2 vorgestellten Konzept des User Interface Management System (siehe Abbildungen 4.13 und 4.15) und ist in der UIMS-Architektur in der Dialogschicht angesiedelt. Mit seiner Hilfe ist die Entwicklung graphisch-interaktiver Benutzerschnittstellen möglich, ohne über bestimmte Kenntnisse einer Programmiersprache und das zugrunde liegende Graphical User Interface (GUI), wie beispielsweise OSF/Motif oder den Presentation Manager, zu verfügen. Aus diesem Grund wird er als User Interface Design System (UIDS) bezeichnet [Rumpf 90]. Neben der Präsentation bietet dieses Tool eine Ablaufsteuerung für Dialoge an sowie Mechanismen zur Anbindung der Anwendung.
Im folgenden wird sich im wesentlichen auf [ISA 91] und [Raether 90] bezogen.

6.2.2 Komponenten des ISA-DM

Der ISA-DMs ist in folgende Komponenten unterteilt:

- Ablaufkomponente (Runtime)
- Entwicklungsumgebung

Mit Hilfe einer Dialogbeschreibung, die in einer Textdatei abgelegt wird (Dialog Definition Language (DDL[1])), können komplexe Dialoge spezifiziert werden. Diese Datei enthält u.a. Definitionen der :

- Objekte, Modelle, Defaults und ihnen zugeordnete Ereignis-Steuerungen
- mehrfach verwendbare Ressourcen (Farben, Fonts, Funktionen, etc.)
- Regelbasis, die die Dialogablaufsteuerung übernimmt.

1. Ablaufkomponente

Beim Programmstart wird zunächst die DDL-Datei geladen und der benutzerinitiierte Dialog begonnen. Die Ablaufkomponente ist für die Objekt- und Ressourcenverwaltung zuständig und bildet die einzelnen Objekte anhand ihrer Attribute und deren Ausprägungen auf das zugrundeliegende GUI ab.

---

[1] Die Abkürzung DDL wird auch im Zusammenhang mit Datenbanken verwendet und ersetzt dort "Database Description Language".

Ein Vorteil des ISA-DMs ist die vergleichbar große Anzahl der unterstützten Fenstersysteme (X-Windows mit OSF/Motif, Open Look; Microsoft Windows; OS/2 Presentation Manager; Alpha Windows[1], etc.) als auch die der Betriebssysteme (Unix, VMS, IX386, NeWSOS, etc.). Die Dialogbeschreibung wird auf das jeweilige "look-and-feel" des GUI abgebildet, wodurch die durch einen Wechsel des GUI oder des Betriebssystemes entstehenden, aufwendigen Anpassungen entfallen.

2. Verteilung der Anwendung

Die Anbindung der Anwendung an ihren Dialog wird durch die Angabe von Funktionen in der Dialogbeschreibung definiert, wobei die Funktionen nicht auf derselben Plattform vorliegen müssen, sondern auch auf einem anderen Rechner (mit unterschiedlichen GUI und Betriebssystem) lokalisiert sein können. Voraussetzung ist, daß alle an diesem verteilten System beteiligten Komponenten den ISA-DM als aktiven Prozess installiert haben. Der Aufruf der Funktion selbst wird vom ISA-DM an den entsprechenden Rechner weitergeleitet und mittels des dort geladenen Pendants ausgeführt. Das Ergebnis der Funktion gelangt auf dem gleichen Weg wieder zurück. Diese Funktionalität unterstützt die Entwicklung verteilter Anwendungen, ohne den hohen Kommunikationsaufwand von z.B. X-Windows zu implizieren. Leider ist die verwendete Kommunikationsschnittstelle des ISA-DM nicht offengelegt und damit auch nicht für andere Zwecke zu verwenden.

3. Entwicklungsumgebung

Die Entwicklungsumgebung umfaßt vier Komponenten:

- Editor
- Simulationskomponente
- Testumgebung
- Schnittstelle für die Programmiersprachen C und COBOL

Der Editor ermöglicht den interaktiven Entwurf der Benutzerschnittstelle der Anwendung. Mit seiner Hilfe können das Layout und die Dialogsteuerung ohne besondere Kenntnisse der DDL generiert, in Form einer DDL-Datei gespeichert und bereits existierende DDL-Dateien modifiziert werden.

---

[1] Alpha Windows ist ein Fenstersystem für alphanumerische Terminals, welches von der ISA GmbH vertrieben wird.

Um das Testen eines erstellten Dialoges zu ermöglichen, ohne dabei auf die Anwendung angewiesen zu sein, wird die Simulationskomponente in Verbindung mit der Testumgebung verwendet. Dieses Konzept erlaubt nicht nur eine vollkommen getrennte Entwicklung von Anwendung und ihrer Benutzerschnittstelle, sondern unterstützt auch die Programmiertechnik des Rapid-Prototyping.

Des weiteren wird eine Programmierschnittstelle angeboten, mit deren Hilfe die Änderung eines geladenen Dialoges zur Laufzeit möglich ist. Derartige Manipulationen sind auf die Erzeugung und das Entfernen von Objekten sowie das Verändern von Objekt-Attributen beschränkt; die Dialogsteuerung ist nur bedingt beeinflußbar.

6.2.3 Sprachumfang

Die DDL des ISA-DM ist objekt- und ereignisorientiert. Objekte werden mittels Instanziierung von Klassen oder Modellen gebildet. Modelle werden dabei als spezielle Unterklassen bezeichnet, die über die gleiche Funktionalität und Attributierung wie deren Vater-Klasse verfügt und durch eine Defaultbelegung der Attributwerte gekennzeichnet ist. Mit Hilfe der objektgebundenen Ereignissteuerung wird die Dialogsteuerung realisiert. Zu einem Objekt kann mit Hilfe einer Regelsprache angegeben werden, wie auf eine bestimmte Benutzeraktion reagiert werden soll; die Trennung von dialog- und anwendungsorientierter Kommunikation ist hiermit realisierbar.

Klassen und Objekte

Die DDL des ISA-DM bietet Dialogobjektklassen [ISA 91] an, mit deren Hilfe Modelle und Objekte gebildet werden können und die in Funktionalität und Bezeichnung ähnlich der der Dialogbeschreibungssprache sind (siehe Kapitel 4.6).
In der unten aufgeführten Liste werden nur die Klassen beschrieben, die eine andere Spezifikation besitzen als die der Dialogbeschreibungssprache bzw. in dieser nicht vorkommen.

Window

Groupbox

Listbox

Der Inhalt einer Listbox besteht aus einer Liste von nicht attributierten Einträgen in Form von Strings, die nicht vergleichbar sind mit den Listboxitems.

Statictext
Edittext
Poptext
Pushbutton
Radiobutton
Checkbox
Scrollbar
Image
Dient der Darstellung von Graphiken (einfachen pix-maps).

Rectangle
graphisches Rechteck; dient der Übersichtlichkeit

Canvas
wird zur Realisierung von graphischen Kontexten (X-Windows) verwendet.

Dialog

Es werden die in Kapitel 4.6.1 vorgestellten Basisklassen im wesentlichen abgedeckt, jedoch im Falle des Listboxitems und der zusammengesetzten Klassen besteht keine direkte Abbildungsmöglichkeit. Diese müßten durch zusätzliche Datentypen realisiert werden, da es nicht möglich ist, mit Hilfe generischer Methoden, komplexere Klassenstrukturen aus den Klassen des ISA-DM zu bilden. Ähnliches gilt für die Attribute der Klassen, welche auch nicht vollständig durch den ISA-DM abgedeckt werden können.
Eine genaue Beschreibung des ISA-DM entnehme man [ISA 91].

6.2.4 Eignung des ISA-DM für das Integrationskonzept

Die während der Testinstallation gesammelten Erfahrungen in Bezug auf die Umsetzung der Dialogschicht des Basissystementwurfs sollen stichpunktartig wiedergegeben werden:

- Funktionale Redundanz in Bezug auf die Kommunikationsschicht. Sowohl der Dialogmanager von ISA als auch das Basissystem haben über Kommunikationsmethoden zu verfügen. Leider sind die des ISA-DMs nicht zugänglich, um den Umfang der Redundanz zu begrenzen.
- Eine Umsetzung der dynamischen Dialoge mittels der DDL ist nicht möglich. Es bedarf einerseits zusätzlicher Klassen zur Realisierung der zusammengesetzten Klassen (generische Methoden werden vom ISA-DM nicht angeboten) und andererseits werden nicht alle Attribute der DiBeSp von der DDL abgedeckt. Des weiteren ist die DDL für dynamische Dialoge nicht ausgelegt. Der ISA-DM geht davon aus, daß zu Beginn einer Anwen-

dung deren gesamter Dialog (DDL-Datei) geladen wird, und zur Laufzeit nur unwesentlich modifiziert wird. Mit Hilfe der C-Schnittstelle jedoch ist eine gewisse Dynamik zu realisieren. Damit ist folgender Kompromiß möglich : Die DDL wird zur Definition aller Modelle und der Dialogsteuerung verwendet und mittels des C-Interface werden die dynamischen Dialoge (DiBeSp) umgesetzt. Anzumerken ist jedoch, daß das C-Interface nur die Modifikation von Objekten und Modellen zuläßt; nicht jedoch von Regeln bzw. Ereignis-Steuerungen.

- Die derzeitige Testinstallation für VMS Version 5.3 und OSF/Motif 1.0 beinhaltet noch einige Kinderkrankheiten, die die Entwicklung nicht immer erleichtern. Ein Update des ISA-DM für OSF/Motif Version 1.1 ist momentan nicht verfügbar.
- Derzeit ist keine Version von Alpha-Windows für VMS erhältlich.

Zusammenfassend läßt sich jedoch sagen, daß der ISA-DM für die Realisierung des Basissystemes geeignet ist. Er deckt leider nicht, wie anfangs erwartet, einen Großteil der Dialogschicht ab, sondern muß um zusätzliche Funktionalität und Datenstrukturen erweitert werden. Dies liegt vor allem in der Abgeschlossenheit des Systems begründet - die verwendeten Datenstrukturen sind nicht zugänglich und können demnach auch nicht direkt erweitert werden. Der so entstehende Aufwand für die notwendigen Ergänzungen ist relativ hoch und impliziert ein großes Maß an Daten- und Funktionalitätsredundanz.
Positiv zu bewerten ist dahingegen die Unabhängigkeit von der GUI und dem Betriebssystem und die Möglichkeit sowohl OSF/Motif als auch Alpha-Windows zu verwenden. Letzteres ist ein wesentlicher Faktor in Bezug auf die Zielgruppe von XUMA. Unter der Voraussetzung, daß Alpha-Windows für VMS verfügbar wird, ist dieses Tool sicherlich eine Erleichterung für die Umsetzung des Basissystems.

6.3 Realisierung des Basissystems

6.3.1 Architektur des Basissystems des Prototyps

In der Abbildung 6.1 wird der Aufbau des Prototyps für das Client-Basissystem beschrieben.

Im Unterschied zu dem in Kapitel 4.3 vorgestellten Basissystem fehlt die Kontrollschicht in der Architektur des Prototyps (siehe Abbildung 6.1). Es wurde bereits in dem o.g. Kapitel darauf hingewiesen, daß die Kontrollschicht ihren Anforderungen anzupassen und ihr Umfang von dem verwendeten Integrationsansatz abhängig ist. In dem vorliegenden Fall bestünde die wesentliche Aufgabe der Kontrollschicht in der Überwachung der Verbindung zwischen Server und Client sowie der Weiterleitung der Nachrichten von der Dialogschicht an die Kommunikationsschicht und umgekehrt. Aufgrund dieser sehr geringen Funktionalität wurden die Aufgaben der Kontrollschicht nicht in einem eigenständigen Modul bzw. Schicht umgesetzt, sondern in Form einer Bibliothek, mit deren Hilfe die Dialogschicht die zusätzlichen Aufgaben übernehmen kann (Verbindungsmodul).

In der Abbildung 6.1 ist die funktionale Redundanz bezüglich Kommunikation und Dialogobjekt-Verwaltung zu erkennen. Die hervorgehobene Dialogschicht stellt die Erweiterung des ISA-Dialogmanagers dar, mit deren Hilfe die vom ISA-DM nicht geleisteten Aufgaben bei der Umsetzung der Dialogbeschreibungssprache realisiert werden sollen. Die Schnittstellen zwischen ISA-DM und Dialogschicht sind das C-Interface und das File-Interface. In der Initialisierungsphase wird eine DDL-Datei (Initialisierungs-DDL-Datei) über die File-Schnittstelle geladen, die alle wesentlichen Modell-, Ressourcen- und Dialogsteuerungsdefinitionen enthält, die von der Dynamik der Dialogbeschreibungssprache unabhängig sind. Mit Hilfe des C-Interface werden die zur Laufzeit eintreffenden DiBeSp-Befehle dem ISA-DM weitergeleitet.
Alle weiteren Komponenten wurden bereits in den Kapiteln 4.3, 4.4 und 4.5.4 vorgestellt.

Einzelheiten der Umsetzung DiBeSp $\Leftrightarrow$ ISA-DM entnehme man der Programmdokumentation.

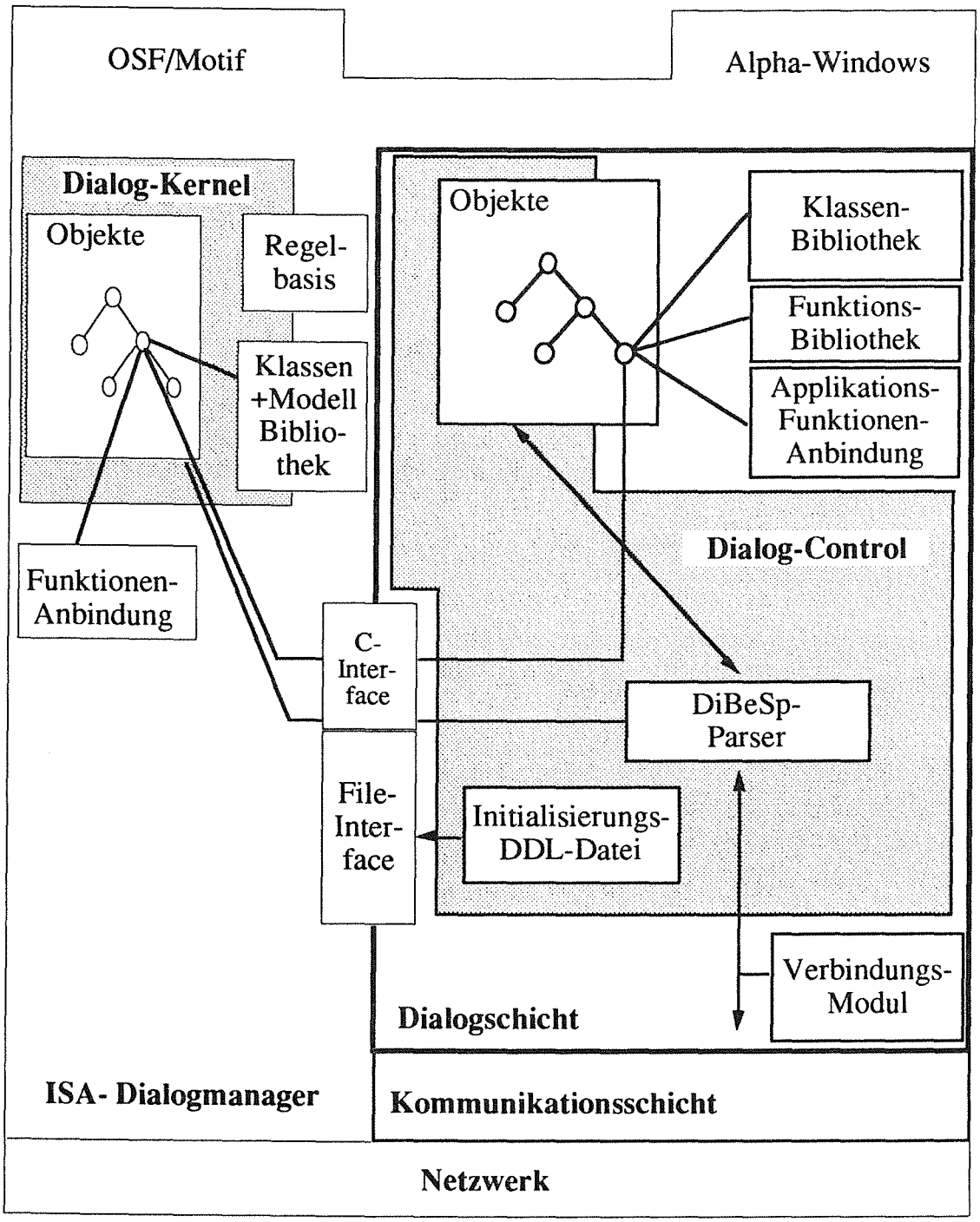


Abb. 6.1: Architektur des Basissystems des Prototyps

6.3.2 Transformation des XUMA-DIF in die DiBeSp

XUMA-DIF	DiBeSp
Status Art Text	Window InfoBox Jede Art der Statusanzeige erhält eine eigene InfoBox mit entsprechenden Namen (Attr_name) InfoBox, Attr_content
Output Text Clear	Window History, Attr_content, Attr_appContent Attr_content = "" (Leerstring)
Single-Menu Header middle? Items <String> <Item-Value> Highlighted Noselect	XMenu Attr_multisel = false Attr_name Pushbutton Attr_name = "Zurück" Attr_userData = ":middle?" (z.B.) XMenuitem Attr_name Attr_UserID (z.B.) Attr_selected = true Attr_sensitive = false
Multiple-Menu Header Items Margin-Choice <String> Value	XMenu Attr_multisel = true Attr_name XMenuitem (siehe Single-Menu) Pushbutton Attr_name Attr_userID (z.B.)
Info Header Text	InfoWindow Attr_name Attr_content

CVV Header CVV-Margin-Choice <String> CVV-Item: String <String> CVV-Item: Feld <Name als Symbol> Value Label Constraint Documentation Edit Type-Spez. CVV-Item: Tabelle <Name als Symbol> Value <Spaltendefinition> <Lfd.Nr.> <Überschrift> Constraint Documentation Edit Type-Spez.	Formular Attr_name Pushbutton Attr_name Statictext Attr_name Klasse siehe Type-Spez. Attr_userID (z.B.) siehe Type-Spez. Attr_name Attr_type, Attr_constraint (siehe Constraints) Attr_help Attr_delOnSelect siehe Type-Spez. Table Attr_userID (z.B.) Attr_content TabCol Attr_cl_nr Attr_name Attr_type, Attr_constraint (siehe Constraints) Attr_help Attr_delOnSelect siehe Type-Spez.
Type-Spez. Choose Value <Werte-Liste> einzelner Wert Set	 Choose Attr_multisel = false Attr_showAll = true Entsprechendes Chooseitem erhält Attr_selected = true Chooseitems Attr_name des Chooseitem Choose Attr_multisel = true Attr_showAll = true

Value	Entsprechende Chooseitems erhalten Attr_selected = true
<Werte-Liste> einzelner Wert	Chooseitems Attr_name des Chooseitem
Menu	Choose Attr_multisel = false Attr_showAll = false
Value	Entsprechendes Chooseitem erhält Attr_selected = true
<cvv-Menu-Liste> <String> Value Highlighted Noselect Menu-Choose	Chooseitems Attr_name Attr_userID (z.B.) Attr_selected = true Attr_sensitive = false Choose
<Liste von Strings> String	Attr_multisel = false Attr_showAll = false Chooseitem Attr_name
Multiple-Menu	Choose Attr_multisel = true Attr_showAll = false
Value	Entsprechende Chooseitems erhalten Attr_selected = true
<cvv-Menu-Liste>	Chooseitem (siehe Menu)
String	Edittext Attr_type = DT_String
Value	Attr_content
Xuma-Numeric	Edittext Attr_type = DT_Range
Value Auswahlliste einzelner String der Liste	Attr_content Menubox Menuitem Attr_name = String
Xuma-Free	Edittext Attr_type = DT_String
Value Auswahlliste einzelner String der Liste	Attr_content Menubox Menuitem Attr_name = String

Xuma-Fix-or-Free Value Auswahlliste einzelner String der Liste Xuma-Datum Value Zahl-oder-String Value	Edittext Attr_type = DT_String Attr_content Menubox Menuitem Attr_name = String Edittext Attr_type = DT_Date Attr_content Edittext Attr_type = DT_IntString Attr_content
Constraints fall-check-lfd-nr para-number ae-check-lfd-nr check-methode	 Attr_type = DT_uIntString Attr_constraint = XConstr_Range DT_uIntString "0" Maximum Attr_type = DT_uIntString Attr_type = DT_uInt Attr_constraint = XConstr_Range DT_uInt 1 Maximum Attr_type = DT_uInt Attr_constraint = XConstr_Range DT_uInt 3 998

Beispiele

Die nun folgenden Beispiele sind als Vorschläge zu betrachten, wie eine Dialogbeschreibung des XUMA-DIF in die Dialogbeschreibungssprache umgesetzt werden kann.

Initialisierung

(DCmd_Initialize "XUMA_Dialog")

Status-Ausgabe

```
( :status :art :System :text ("XUMA wird initialisiert"))
```

Bei erster Verwendung erfolgt die Initialisierung:

```
( DCmd_Create "XUMA_Dialog"

    ( "Status" Window
            Attr_name "XUMA-Status"

        ( "System" InfoBox
                Attr_name "System"
                Attr_content "XUMA wird initialisiert" )))
```

Weitere Verwendungen desselben Objektes:

```
( :status :art :system :text ("Auswahl des Falles")

( DCmd_SetObjectAttribute "System"
        Attr_content "Auswahl des Falles")
```

Text-Ausgabe

```
( :output :text ( "Ausgewählter Fall:"))
```

Erste Verwendung der Textausgabe

```
( DCmd_Create "XUMA_Dialog"

    ( "Output" Window
            Attr_sensitive false

        ( "Output_Text" History
                Attr_content "Ausgewählter Fall")))
```

Weitere Verwendung:

```
( :output :text (    "Gemeinde  : Mannheim"
                     "Betrieb   : Gaswerk"))

( DCmd_SetObjectAttribute "Output_Text"
        Attr_appContent "Gemeinde   : Mannheim"
        Attr_appContent "Betrieb    : Gaswerk")
```

Löschen der Text-Ausgabe

```
( :output :clear)

( DCmd_SetObjectAttribute "Output_Text"
            Attr_content "")
```

Single-Menu

```
( :single-menu      :header "XUMA-Hauptfunktionen"
                    :items (
                          ("" :no-select nil)
                          (AUSFALL "Auswahl des Falls")
                          (XUMA-ENDE "XUMA beenden"))
                    :middle? t)

( DCmdCreate "XUMA_Dialog"

      ( "SingleMenu" XMenu
                  Attr_name "XUMA-Hauptfunktionen"
                  Attr_showall true
                  Attr_multisel false
                  Attr_defaultExit true
                  Attr_exitFunc DCB_GetObj "SingleMenu"

            ( "" XMenuitem
                        Attr_name ""
                        Attr_sensitive false)

            ( "" XMenuitem
                        Attr_name "Auswahl des Falls"
                        Attr_userID "AUSFALL")

            ("" XMenuitem
                        Attr_name "XUMA beenden"
                        Attr_userID "XUMA-ENDE"))

      ( "Right" Pushbutton
                  Attr_name "Info"
                  Attr_userID ":button :right"
                  Attr_exitFlag true)
```

```
( "Middle" Pushbutton
        Attr_name "Zurück"
        Attr_userID ":button :middle"
        Attr_exitFlag true)

( "Left" Pushbutton
        Attr_name "OK"
        Attr_userID ":button :left"
        Attr_exitFlag true))
```

Formular

```
( :cvv      :header "Frage ?"
            :items (
                ( B-FRAGE  :value JA
                           :label "Wollen Sie diesen Fall laden ?"
                           :choose (JA NEIN)))
            :cvv-margin-choices ( "?"  "Ausführen" "Zurück"))
```

```
( DCmd_Create "XUMA_Dialog"

    ( "CVV" Formular
        Attr_name "Frage ?"
        Attr_defaultExit true
        Attr_exitFunc DCB_GetObj "CVV" true true

        ( "B_FRAGE" Choose
                Attr_name "Wollen Sie diesen Fall laden ?"
                Attr_showAll true
                Attr_multisel false

            ("" Chooseitem
                    Attr_name "JA"
                    Attr_selected true
                    Attr_userID "JA")
```

```
        ( "" Chooseitem
                Attr_name "NEIN"
                Attr_userID "NEIN"))
("" Pushbutton
        Attr_name "?"
        Attr_userID "?"
        Attr_exitFlag true)

( "" Pushbutton
        Attr_name "Ausführen"
        Attr_userID "Ausführen"
        Attr_exitFlag true)

( "" Pushbutton
        Attr_name "Zurück"
        Attr_userID "Zurück"
        Attr_exitFlag true)))
```

6.3.3 Kopplung der Schichten des Basissystems

Für die Kopplung der einzelnen Schichten des Basissystemes werden prioritätsgesteuerte Ereignis-Warteschlangen verwendet, mit deren Hilfe eine asynchrone Kommunikation zwischen den Schichten ermöglicht wird. Die Warteschlangen sind nach dem FIFO-Prinzip aufgebaut, wobei als zusätzliches Ordnungskriterium die Priorität eines Ereignisses verwendet werden kann. Ebenfalls optional kann die Warteschlange nach Absendern sortiert verwaltet werden.
Will eine Schicht einer anderen eine Mitteilung machen, so legt sie ein entsprechendes Ereignis in die Input-Warteschlange des Adressaten. Sobald der Empfänger das Ereignis (Nachricht) erhalten und ausgewertet hat, kann er seinerseits Ereignisse an den Absender mit Hilfe dessen Input-Warteschlange weiterleiten (vergleiche hierzu die Abbildungen 4.6 und 4.10).

6.3.4 Kommunikationsschnittstelle

In diesem Abschnitt werden die einzelnen Befehle, deren Ergebnisse als auch die eintretenden Ereignisse der Schnittstelle zwischen der Kontrollschicht und der Kommunikationsschicht beschrieben, welche teilweise systembedingt sind. Z.B. sind Fehlermeldungen für nicht ausreichenden Speicher möglich.
Das Protokoll entnehme man dem Abschnitt 4.4.2.

1. Befehlssatz

In der folgenden Tabelle sind die möglichen Befehle der Kommunikationsschnittstelle beschrieben, die dem Benuzter (Kontrollschicht) zur Verfügung stehen. Bei den Result Codes sind nur diejenigen angegeben, die von der Funktion selber zurückgegeben werden. Solche, die als Events an den Aufrufer gelangen sind nicht aufgeführt und werden in den folgenden Abschnitten behandelt.

Kommando	Beschreibung	Result Codes
INITIALIZE	Initialisierung der Kommunikationsschnittstelle, Anmelden im Netz, Warteschlangen initialisieren, ...	R_OK E_NO_MEMORY E_NOT_INITIALIZED E_INTERNAL E_UNDEF_ERR
RESET	Schnittstelle zurücksetzen, Warteschlangen löschen, Verbindungen abbrechen	R_OK E_NOT_INITIALIZED E_INTERNAL E_UNDEF_ERR
SEND_MSG	Senden einer Nachricht. Übernahme des Befehls wird bestätigt und allgemeine Randbedingungen werden geprüft. Konnte die Meldung nicht gesendet werden, so wird dies mit einem Event mitgeteilt.	R_OK E_TRANS_TOO_LONG E_NO_MEMORY E_NO_CONNECTION E_NOT_INITIALIZED E_INTERNAL E_UNDEF_ERR
READ_MSG	Nachricht lesen. Ist nur möglich, wenn vorher ein entsprechendes Ereignis gemeldet wurde.	R_OK E_NO_CONNECTION E_NO_MEMORY E_NO_MSG E_NOT_INITIALIZED E_INTERNAL E_UNDEF_ERR
FLUSH	Alle Nachrichten einer Verbindung werden schnellstmöglich gesendet, Warteschlange geleert	R_OK E_NO_CONNECTION E_NOT_INITIALIZED E_INTERNAL E_UNDEF_ERR

OPEN_CONNECTION	Herstellen einer Verbindung.	R_OK E_UNKNOWN_CONNECTION E_TOO_MANY_CONNECTIONS E_TIME_OUT E_CONNECTION_OPENED E_REQ_REJECTED E_NO_MEMORY E_NOT_INITIALIZED E_INTERNAL E_UNDEF_ERR
CLOSE_CONNECTION	Verbindung schließen	R_OK E_NO_CONNECTION E_REQ_REJECTED E_TIME_OUT E_NOT_INITIALIZED E_INTERNAL E_UNDEF_ERR
ABORT_CONNECTION	Verbindung sofort und bedingungslos abbrechen. Warteschlange löschen und Verbindung schließen. Gegenseite informieren.	R_OK E_NO_CONNECTION E_NOT_INITIALIZED E_INTERNAL E_UNDEF_ERR
RESET_CONNECTION	Verbindung zurücksetzen. Warteschlangen bei Sender und Empfänger löschen. Verbindung bleibt bestehen. Einheitlichen Status herstellen.	R_OK E_NO_CONNECTION E_TIME_OUT E_NOT_INITIALIZED E_INTERNAL E_UNDEF_ERR
REJECT_REQUEST	Aufforderung zum Verbindungsaufbau bzw. -abbau von aussen zurückweisen	R_OK E_NO_CONNECTION E_NOT_INITIALIZED E_INTERNAL E_UNDEF_ERR
ACKN_REQUEST	Aufforderung zum Verbindungsaufbau bzw. -abbau annehmen und durchführen.	R_OK E_NO_CONNECTION E_NOT_INITIALIZED E_INTERNAL E_UNDEF_ERR
GET_PENDING_MSGS	Liefert die Anzahl der noch anstehenden Nachrichten für eine Verbindung. -> Synchronisation / Zustandsermittlung bei fehlerhaften Verbindungen	R_OK E_NO_CONNECTION E_NOT_INITIALIZED E_INTERNAL E_UNDEF_ERR
GET_CONNECTION_DATA	Zur Verfügung stellen von Verbindungsdaten.	R_OK E_NO_MEMORY E_NO_CONNECTION E_NOT_INITIALIZED E_INTERNAL E_UNDEF_ERR

PUT_CONNECTION_DATA	Speichern von Verbindungsdaten, wenn noch nicht vorhanden, dann neu, ansonsten überschreiben	R_OK E_NO_MEMORY E_NOT_INITIALIZED E_INTERNAL E_UNDEF_ERR
DEL_CONNECTION_DATA	Verbindungsdaten löschen	R_OK E_NO_MEMORY E_NO_CONNECTION E_NOT_INITIALIZED E_INTERNAL E_UNDEF_ERR
GET_CONNECTION_LIST	Liste aller Verbindungsdaten erstellen	R_OK E_NO_MEMORY E_NO_CONNECTION E_NOT_INITIALIZED E_INTERNAL E_UNDEF_ERR
BEGIN_TRANSACTION	Transaktionsbeginn setzen.	R_OK E_NO_CONNECTION E_TRANS_SET E_NO_MEMORY E_NOT_INITIALIZED E_INTERNAL E_UNDEF_ERR
END_TRANSACTION	Transaktionsende setzen.	R_OK E_NO_CONNECTION E_TRANS_NOT_SET E_NOT_INITIALIZED E_INTERNAL E_UNDEF_ERR
ABORT_TRANSACTION	Transaktion unterbrechen. Sender und Empfänger stellen Zustand vor Transaktionsbeginn wieder her.	R_OK E_NO_CONNECTION E_TRANS_NOT_SET E_NO_MEMORY E_NOT_INITIALIZED E_INTERNAL E_UNDEF_ERR

2. Funktionsergebnisse

Im folgenden sollen die einzelnen Funktionsergebnisse erläutert werden. Es gibt durchaus Ergebnisse, die auch als eigenständige Ereignisse existieren. So kann während der Ausführung einer Funktion ein interner Fehler auftreten als auch zu einem beliebigem Zeitpunkt. Im ersten Fall liefert die Funktion E_INTERNAL zurück und im zweiten tritt ein Ereignis mit gleichem Namen ein.

Result Code	Beschreibung
R_OK	fehlerfreie Ausführung
E_CONNECTION_OPENED	Die Verbindung ist bereits geöffnet.
E_INTERNAL	interner Fehler (schwerwiegend)
E_IS_INITIALIZED	Das Modul ist bereits initialisiert.
E_NOT_INITIALIZED	Das Modul ist noch nicht initialisiert.
E_NO_CONNECTION	Eine solche Verbindung ist nicht aufgebaut.
E_NO_MEMORY	Zusätzlicher Speicher war zur Ausführung des Befehles notwendig, konnte aber nicht zur Verfügung gestellt werden.
E_TIME_OUT	Verbindungsaufbau bzw. -abbau schlägt aus nicht netzabhängigen Problemen fehl (evntl. logische Gründe).
E_TOO_MANY_CONNECTIONS	Es können keine weiteren Verbindungen aufgebaut werden.
E_TRANS_TOO_LONG	Die Länge der Transaktion ist zu lange. Die zu sendende Nachricht wird nicht angenommen.
E_UNDEF_CMD	Der angeforderte Befehl ist unbekannt.
E_UNDEF_ERR	nicht bestimmbarer Fehler (schwerwiegend)
E_UNKNOWN_CONNECTION	Verbindungsaufbau konnte nicht durchgeführt werden, da die Verbindung unbekannt ist.

3. Ereignisse

Hierbei handelt es sich um die von der Kommunikationsschnittstelle ausgelösten Ereignisse, die in die Nachrichtenwarteschlange der übergeordneten Schicht eingereiht werden. Diese sind verbunden mit der Angabe, um welche Verbindung es sich handelt, die das Ereignis ausgelöst hat.

Ereignis	Beschreibung
R_MSG_RECEIVED	Nachricht eingetroffen
R_REQ_CONNECTION_CLOSE	Anforderung auf Schließen einer Verbindung
R_REQ_CONNECTION_OPEN	Anforderung auf Öffnen einer Verbindung
R_REQ_CONNECTION_RESET	Anforderung auf Zurücksetzen der Verbindung
R_TRANS_ABORTED	Transaktion wurde abgebrochen
R_TRANS_BEGIN	Transaktionsbeginn wurde gesetzt
R_TRANS_END	Transaktionsende wurde gesetzt
E_CONNECTION_ABORTED	Verbindung unerwarteterweise abgebrochen
E_INTERNAL	interner Fehler (schwerwiegend)
E_REQ_REJECTED	Verbindungsaufbau- bzw. -abbauanforderung wurde zurückgewiesen
E_TIME_OUT	Verbindungsaufbau bzw. -abbau schlägt aus nicht netzabhängigen Problemen fehl (evntl. logische Gründe).
E_UNDEF_ERR	nicht bestimmbarer Fehler (schwerwiegend)

7. Zusammenfassung

Die Integration von einzelnen Umweltinformatik-Systemen in ein Umweltinformationssystem ist von großer Bedeutung für die Bewältigung der vielschichtigen und komplexen Probleme im Umweltbereich. In Abhängigkeit vom Grad der Integration können die anfallenden Aufgaben mit Hilfe der einzelnen Systeme sowohl zeitlich beschleunigt als auch qualitativ verbessert durchgeführt werden und es lassen sich Daten- und Funktionalitätsredundanzen minimieren. In dieser Arbeit wurden drei Ansätze vorgestellt, wie eine solche Integration prinzipiell aussehen kann. Sie sind nach Art und Umfang, in welchem eine Anwendung Dienste zur Verfügung stellt, klassifizierbar. Im Vordergrund stehen dabei die Anforderungen Benutzerfreundlichkeit, Effizienz und Zuverlässigkeit. Erstgenannte betrifft im wesentlichen das UIS, wohingegen sich die letzteren sowohl vom Benutzer an das Umweltinformationssystem als auch von diesem an die einzelnen Anwendungen und wiederum von den Anwendungen an das UIS und den Benutzer richten.

Der anwendungsorientierte Integrationsansatz ist für Einzelsysteme anzuwenden, die nicht in Dienste oder Objekte unterteilbar sind, sondern nur als ganzes zugänglich sind. Bis auf die Benutzerschnittstelle bieten diese Anwendungen keine Schnittstelle nach "außen" an. Vorteilhaft ist dabei der vergleichsweise geringe Realisierungsaufwand für die Umsetzung der Integration, jedoch ist dieser Ansatz nicht ausreichend für ein komplexes Umweltinformationssystem. Er stellt zwar die Verfügbarkeit der Einzelsysteme für den Benutzer des UIS sicher - unterstützt den Benutzer bei der eigentlichen Problemlösung jedoch nicht.

Mit Hilfe der dienstorientierten Integration kann sich der Anwender einen Überblick über die von den Anwendungen angebotenen Dienste machen, womit die Zuordnung von Problemstellung zu Problemlösungsmethoden unterstützt wird, ohne daß explizite Kenntnisse über die Anwendung selbst vorhanden sein müssen.

Bei diesen beiden Ansätzen erfolgt die Erbringung der Dienstleistung der einzelnen Anwendung über deren Benutzerschnittstelle, die auf die Rechnerumgebung des Benutzers transformiert werden muß.

Die objektorientierte Integration hingegen ermöglicht es dem Benutzer nicht nur, in seiner gewohnten Rechnerumgebung zu arbeiten, sondern gewährleistet auch, daß er von spezifischen Oberflächendetails der einzelnen Anwendungen unabhängig bleibt. Wesentlich bei diesem Ansatz ist die Möglichkeit, detaillierte Problemspezifikationen direkt auf die von den Anwendungen angebotenen Objekte und deren Methoden abzubilden. Er stellt damit den mächtigsten Ansatz dar.

XUMA erfüllt nur die Anforderungen für eine anwendungsorientierte Integration, weshalb sich die vorliegende Arbeit vornehmlich mit diesem Integrationsansatz beschäftigt. Besonderes Gewicht wird dabei auf Benutzerfreundlichkeit, Effizienz und Flexibilität gelegt.

Die Umsetzung jedes dieser Ansätze bedarf auf Seiten des Umweltinformationssystems (UIS) eines Basissystems, dessen Komplexität von den unterstützten Integrationsansätzen abhängt. Dieses System gewährleistet mindestens die Kommunikation zwischen Anwendung und dem UIS (Kommunikationsschnittstelle) und stellt Objekte und Methoden für die Abbildung der Benutzerschnittstelle der Anwendung auf die des UIS zur Verfügung (Dialogschnittstelle). Die zuletzt genannte Schnittstelle ist durch eine Dialogbeschreibungssprache (DiBeSp) spezifiziert, mit deren Hilfe dynamische Dialoge abstrakt beschrieben werden können.

Diese Dialogbeschreibungssprache besitzt folgende Eigenschaften:

- Unterstützung von rechner- und benutzerinitiierter Dialogsteuerung
- Trennung von Anwendungs- und Oberflächenfunktionalität
- Unabhängigkeit vom verwendeten Graphischen User Interface (GUI) der Anwendung als auch des UIS
- Reduzierung des Kommunikationsaufwandes zwischen Anwendung und UIS auf den anwendungsbezogenen Anteil
- Unterstützung aller Integrationsansätze
- Berücksichtigung ausschließlich alpha-numerischer Informationen

Die Verwendung einer abstrakten Dialogbeschreibungssprache impliziert Transformationen sowohl auf seiten der Anwendung als auch seiten des UIS, welches die DiBeSp auf das zugrunde liegende GUI abzubilden hat.

Die Kommunikationsschnittstelle des Basissystems stellt die Grundlage für die fehlerfreie Kommunikation zwischen Anwendung und dem UIS zur Verfügung und läßt sich wie folgt charakterisieren:

- fehlertolerante, asynchrone Nachrichtenübermittlung in einem heterogenen Netzwerkverbund
- Unterstützung von Vorrangdaten

- Unterstützung von Transaktionen
- Verwaltung mehrerer Verbindungen
- Unabhängigkeit von Rechner-Plattform und Netzwerk

Die Umsetzung des Integrationskonzeptes für die konkrete Anwendung XUMA ergab, daß auf seiten des UIS ein hoher Realisierungsaufwand für die Umsetzung der Dialogschicht für unterschiedliche GUIs besteht. Jedoch kann dieser durch den Einsatz von (kommerziellen) Tools reduziert werden, impliziert eventuell aber Funktionalitäts- und Datenstrukturredundanzen aufgrund der Abgeschlossenheit des eingesetzten Werkzeugs.
Die Transformation der anwendungspezifischen Benutzeroberfläche in die Dialogbeschreibungssprache ist für Anwendungen, die keine oder nur eine geringe Trennung von Anwendungs- und Oberflächenfunktionalität aufweisen, problematisch und aufwendig. Je stärker diese Trennung jedoch vorliegt und bzw. oder objektorientierte Techniken zum Einsatz kommen, desto einfacher ist die Transformation zu realisieren.

In dieser Arbeit wurden (geo-)graphische Informationen nicht betrachtet. Gerade aber im Umweltinformatik-Bereich spielen diese eine zentrale Rolle. Es existieren bereits eine Vielzahl von geographischen Informationssystemen (GIS), die es in einen Systemverbund zu integrieren gilt. Hier besteht ein unmittelbarer Bedarf für weitere Forschungsaktivitäten, wobei unter anderem folgende Fragen zu beantworten wären:

1. Wie kann eine Verknüpfung bzw. Beziehung zwischen einem Objekt einer Anwendung und seiner geographischen Repräsentanz in einem GIS hergestellt werden ?

2. Wie können geograpische Informationen auf unterschiedlichen Systemen gleichartig und ihrer problembezogenen Relevanz nach dargestellt werden ?

Anhang

A. Verwendete Abkürzungen

ALB	Automatisiertes Liegenschaftsbuch
ALBIS	Arten-, Landschafts-, Biotop-Informationssystem
ALK	Automatisierte Liegenschaftskarte
API	Application Programming Interface
ASCII	Americon Standard Codes of Information Interchange
AT&T	American Telephone and Telegraph Co.
ATKIS	Amtliches Topographisch-Kartographisches Informationssystem
CCITT	Comité Consultatif International Telegraphique et Telephonique
CCITT X.25	Paketvermittlung, Vermittlungsschicht
DATEX	Data Exchange
Datex-P	Paketvermittelndes Data Exchange
DEC	Digital Equipment Cooperation
DECnet	Paketvermittlungsnetz von DEC
DiBeSp	Dialogbeschreibungssprache
DIF	Dialog Interface
DIN	Deutsches Institut für Normung
DNA	Digital Network Architecture
GI	Gesellschaft für Informatik e.V.
GUI	Graphical User Interface
HP	Hewlett-Packard Co.
IBM	International Business Machines
ILR	Informationssystem ländlicher Raum
ISO/OSI-BRM	Basic Reference Modell der ISO für die OSI
ISO	International Organisation for Standardisation
JURIS	Juristisches Informationssystem der BRD
KIWI	Kommunikativ Integriertes Wasserwirtschaftliches Infosystem Arbeitsdatei für wasser- und abfallwirtschaftliche Objekte
LAN	Local Area Network
LfU	Landesanstalt für Umweltschutz Baden Württemberg
LVN	Landesverwaltungsnetz
MIT	Massachusetts Insitute of Technology
MVC	Model View Controller (Ansatz)
NeWS	Network Extensible Window System
OSF	Open Software Foundation
OSI	Open System Interconnection
PAD	Packet Assembly / Disassembly facility
PM	Presentation Manager

PM/X	Presentation Manager für GUI
PSI	Packetnet System Interface
RIPS	Räumliches Informations- und Planungssystem
SAA	System Application Architecture
SCO	Santa Cruiz Operation Inc.
SNA	Systems Network Architecture
TCP/IP	Transmission Control Protocol / Internet Protocol
TULIS	Technosphäre und Luft-Informationssystem
UFIS	Umwelt-Führungs-Informationssystem
UIDS	User Interface Design System
UIL	User Interface Language
UIMS	User Interface Management System
UIS	Umweltinformationssystem Baden-Württemberg
WWA	Wasserwirtschaftsamt
Xlib	C-Bibliothek für X-Windows
XPG	X/Open Portability Guide
Xt	X Toolkit
XUI	X User Interface
XUMA	Expertensystem Umweltgefährlichkeit von Altlasten

B. (Eingetragene) Warenzeichen

DB2	IBM
dBASE	Ashton Tate
NeWS	Sun
New Wave	HP
NeXT-Step	NeXT
Open Desktop	SCO
Open Look	AT&T, Sun, Xerox
OSF/Motif	OSF
X-Windows	MIT
XUI	DEC

Literaturverzeichnis

Atlas 89
Atlas, A., et al.:
"OSF User Environment Component - Decision Rationale Document"
Open Software Foundation, Cambridge, Januar 1989

Bartl 86
Bartl, D.:
"Konzeption für das Umweltinformationssystem in Baden-Württemberg (UIS)"
in: Jaeschke, et al.: Informatikanwendungen im Umweltbereich,
Kernforschungszentrum Karlsruhe, Kolloquium 30.9.-1.10.1986, KfK 4223, S.33-46

Balzert 82
Balzert, H.:
"Die Entwicklung von Software-Systemen"
Bibliographisches Institut, 1982, S.10-15

Bullinger 90
Bullinger, H.-J.:
"Wettbewerbsvorteile durch Software-Ergonomie"
in: Bullinger, H.-J. (Hrsg.): Software Ergonomie in der Praxis, IPA-IAO Forschung und Praxis, Band T19, IPA Stuttgart 27.11.1990, Springer-Verlag 1990, S.9-26

Burgstaller 89
Burgstaller, J.; Grollmann, J.; Kapsner, F.:
"Eine Designumgebung für das Prototyping und die Generierung von Bedienoberflächen"
in: Softwaretechnik-Trends (GI-SE), Band 9, Heft 2, Oktober 1989, S.39-52

Clausen 89
Clausen, U.:
"Eine interaktive Wissenserwerbskomponente für ein wissensbasiertes Altlastensystem"
Kernforschungszentrum Karlsruhe, KfK 4600, August 1989

Collet 90
Collet, M.; Hälker-Küsters, M.; Tischendorf, M.:
"Darstellung der Ergebnisse einer Untersuchung über den Einsatz von Expertensystemen im Bereich Umwelt"
in: Pillmann, W., et al. (Hrsg.): Informatik für den Umweltschutz, 5. Symposium 1990 Wien, Informatik Fachberichte 256, Springer-Verlag Berlin Heidelberg 1990, S.167-176

Cote-Munoz 89
Cote-Munoz, J.A.; Kapsner, F.:
"Benutzerspezifische Dialoggestaltung und deren Unterstützung durch Modellierung von Benutzerklassen"
in: Paul, M. (Hrsg.): GI - 19. Jahrestagung I - Computergestützter Arbeitsplatz, Proceedings, München 1989, Springer-Verlag Berlin Heidelberg 1989, S.198-210

Cox 86
Cox, B.,J.:
"Object Oriented Programming"
Reading Mass, Addison Wesley 1986

Fuhr 90
Fuhr, N.:
"Anfragefunktionen für Umweltinformationssysteme"
in: Pillmann, W., et al. (Hrsg.): Informatik für den Umweltschutz, 5. Symposium 1990 Wien, Informatik Fachberichte 256, Springer-Verlag Berlin Heidelberg 1990, S.27-37

Gappa 88
Gappa, U.:
"Wissensakquisition für Expertensysteme und Entwurf des Akquisitionssystems CLASSIKA für heuristische Klassifikation"
Diplomarbeit, Universität Kaiserslautern, 1988

Geiger 89
Geiger, W.; Weidemann, R.; Eitel, W.:
"Konzepte des Expertensystems XUMA für Altlasten"
in: Kernforschungszentrum Karlsruhe, KfK-Nachrichten, Jahrgang 21, 1989, Heft 3, S.133-137

Geiger 90
Geiger, W.; Weidemann, R.:
"The XUMA expert system for contaminated sites : functions, explanations and knowledge acquisition"
10. International Workshop 'Expert Systems & Their Applications', General Conference Avignon 1990, Avignon, Frankreich, 28.5.-1.6.1990, S.951-961

Geiger 91
Geiger, W.:
"Das Expertensystem XUMA für Altlasten - Unterstützung der Untersuchungsplanung"
Workshop Erkundung, Bewertung und Sanierung ehemaliger Gaswerksstandorte, Kernforschungszentrum Karlsruhe, 21.Februar 1991

Green 85/1
"Report on Dialogue Specification Tools"
in: Pfaff, E. (Hrsg.): User Interface Management System, Proceedings, Workshop on UIMS, Seeheim (BRD), 1.-3.11.1983, Springer-Verlag Berlin Heidelberg, S.8-29

Green 85/2
"Design Notations and User Interface Management Systems"
in: Pfaff, E. (Hrsg.): User Interface Management System, Proceedings, Workshop on UIMS, Seeheim (BRD), 1.-3.11.1983, Springer-Verlag Berlin Heidelberg, S.89-107

Grollmann 90
Grollmann, J.:
"Werkzeuge zur Gestaltung von Bedienoberflächen für UNIX-Arbeitsplatzrechner - Ein Überblick"
in: Bullinger, H.-J. (Hrsg.): Software Ergonomie in der Praxis, IPA-IAO Forschung und Praxis, Band T19, IPA Stuttgart 27.11.90, Springer-Verlag Berlin Heidelberg 1990, S.27-42

Heimlich 89
Heimlich, A.:
"Dialogmodellierung mit einem objektorientierten User Interface Management System (UIMS)"
in: Paul, M. (Hrsg.): GI - 19. Jahrestagung I - Computergestützter Arbeitsplatz, Proceedings, München 1989, Springer-Verlag Berlin Heidelberg 1989, S.174-185

Henning 89
Henning, I.:
"Realisierung des Umweltinformationssystems Baden-Württemberg (UIS) am Beispiel des Projektes Umwelt-Führungs-Informationssystem (UFIS)"
in: Jaeschke, A., et al. (Hrsg.): Informatik im Umweltschutz, 4. Symposium 1989 Karlsruhe, Informatik Fachberichte 228, Springer-Verlag Berlin Heidelberg 1989, S.190-202

Huber 88
Huber, K. P.:
"Erklärungskomponente für das Expertensystem XUMA unter Berücksichtigung verschiedener Benutzerklassen"
Kernforschungszentrum Karlsruhe, KfK 4478, November 1988

Hübner 90
Hübner, M.; v. Luck, K.; Weiland, U.:
"Ein Expertensystem zur Unterstützung der Bewertung in Umweltverträglichkeitsprüfungen"
in: Pillmann, W., et al. (Hrsg.): Informatik für den Umweltschutz, 5. Symposium 1990 Wien, Informatik Fachberichte 256, Springer-Verlag Berlin Heidelberg 1990, S.187-196

Ilg 90
Ilg, R.:
"Oberflächengestaltung mit OSF/Motif"
in: Bullinger, H.-J. (Hrsg.): Software Ergonomie in der Praxis, IPA-IAO Forschung und Praxis, Band T19, IPA Stuttgart 27.11.1990, Springer-Verlag 1990, S.199-213

ISA 91
"ISA Dialogmanager, User´s Manual, Programmer´s Guide"
ISA GmbH, Stuttgart, 1991

Jaeschke 89
Jaeschke, A.; Geiger, W.; Page, B.:
"Informatik im Umweltschutz", 4. Symposium, Karlsruhe November 1989, Proceedings, Informatik Fachberichte 228, Springer-Verlag Berlin Heidelberg 1989

Kneißl 87
Kneißl, F.:
"Ein Window Management mit objektorientiertem Baukastensystem"
in: GUUG 1987 Proc., Unix in Deutschland, Hagenburg, 1987

Köhler 90
Köhler, K.; Krönert, G.; Ruchowski, U.:
"Design der Benutzerschnittstelle von Dialogsystemen mit attributierten Grammatiken"
in: Softwaretechnik-Trends(GI-SE), Band 10, Heft 1, April 1990, S.30-42

Kremers 90
Kremers, H.; Line, M.P.; Neugebauer, L.; Riethmüller, R.; Windhorst, W.:
"Arbeitskreis ´Umweltdatenbanken´ - Ziele und erste Ergebnisse"
in: Pillmann, W., et al. (Hrsg.): Informatik für den Umweltschutz, 5. Symposium 1990 Wien, Informatik Fachberichte 256, Springer-Verlag Berlin Heidelberg 1990, S.1-16

Lenz 90
Lenz, G.:
"HP New Wave - Intelligente Arbeitsplatzrechner als Fenster zu kooperativen Rechnerumgebungen"
in: Bullinger, H.-J. (Hrsg.): Software Ergonomie in der Praxis, IPA-IAO Forschung und Praxis, Band T19, IPA Stuttgart 27.11.90, Springer-Verlag Berlin Heidelberg 1990, S.57-86

Mayer-Föll 89
Mayer-Föll, R.:
"Konzeption des ressortübergreifenden Umweltinformationssystems Baden-Württemberg"
in: Jaeschke (Hrsg.): Informatik im Umweltschutz, 4. Symposium 1989 Karlsruhe, Informatik Fachberichte 228, Springer-Verlag Berlin Heidelberg1989, S.178-189

Muth 89
Muth, M.; Neumann, Th.:
"Das UIMS Theseus"
in: Paul, M. (Hrsg.): GI - 19. Jahrestagung I - Computergestützter Arbeitsplatz, Proceedings, München 1989, S.186-197, Springer-Verlag Berlin Heidelberg 1989

Muth 91
Muth, M.:
"Das X-Window-System"
in: Informatik-Spektrum (GI), Band 14, Heft 1, Februar 1991, Springer-Verlag Berlin Heidelberg 1991, S.34-36

Müller 89
Müller, J.:
"Objektorientierte Bedienoberflächen auf der Basis von Standard-Fenstersystemen"
in: Paul, M. (Hrsg.): GI - 19. Jahrestagung I - Computergestützter Arbeitsplatz, Proceedings, München 1989, Springer-Verlag Berlin Heidelberg 1989, S.160-173

OSF-SG 90
"OSF/Motif Style Guide"
Open Software Foundation, Prentice-Hall Int., London 1990

Page 90/1
Page, B.; Jaeschke, A.; Pillmann, W.:
"Angewandte Informatik im Umweltschutz, Teil 1"
in: Informatik Spektrum, Band 13, Heft 1, Springer-Verlag Berlin Heidelberg 1990, S.6-16

Page 90/2
Page, B.; Jaeschke, A.; Pillmann, W.:
"Angewandte Informatik im Umweltschutz, Teil 2"
in: Informatik Spektrum, Band 13, Heft 2, Springer-Verlag Berlin Heidelberg 1990, S.86-97

Pfaff 85
Pfaff, E. (Hrsg.):
"User Interface Management Systems"
Proceedings of the Workshop on UIMS, Seeheim (BRD), 1.-3.11.1983,
Springer-Verlag Berlin Heidelberg 1985

Puppe 88
Puppe, F.:
"Einführung in Expertensysteme"
Springer-Verlag Berlin Heidelberg 1988

Puppe 90
Puppe, F.:
"Problemlösungsmethoden in Expertensystemen"
Springer-Verlag Berlin Heidelberg 1990

Raether 90
Raether, C.:
"ISA Dialogmanager - Ein Entwicklungswerkzeug zur Erstellung portabler, graphischer Benutzeroberflächen"
in: Bullinger, H.-J. (Hrsg.): Software Ergonomie in der Praxis, IPA-IAO Forschung und Praxis, Band T19, IPA Stuttgart 27.11.90, Springer-Verlag Berlin Heidelberg 1990, S.43-56

Reuter 87
Reuter, A.:
"Kopplung von Datenbank- und Expertensystemen"
in: Informationstechnik it, 29. Jahrgang, Heft 3, 1987, S.146-175

Rumpf 90
Rumpf, Ch.:
"Toolkits und User Interface Design Systeme auf X-Windows"
in: Softwaretechnik-Trends (GI-SE), Band 10, Heft 2, Oktober 1990, S.48-76

Scheifler 86
Scheifler, R.W.; Gettys, J.:
"The X Window System"
in: ACM Transactions on Graphics, Vol. 5, No. 2, April 1986, S.79-109

Schimak 91
Schimak, G.; Denzer, R.:
"Komplexe Inhalte eines Umweltinformationssystems"
in: Denzer, R., et al. (Hrsg.): Workshop ´Visualisierung von Umweltdaten´, 1990, Proceedings, Informatik Fachberichte, Band 274, Springer-Verlag 1991 (im Druck)

Schlüter 90
Schlüter, P.; Behdjati, A.; Fleischer, P.; Bagdon, S.:
"Objektorientierte Software-Entwicklung, Konzepte und Terminologie"
in: Softwaretechnik-Trends (GI-SE), Band 10, Heft 2, Oktober 1990, S.22-44

Schmitt 83
Schmitt, A.A.:
"Dialogsysteme"
Bibliographisches Institut AG, Zürich 1983

Shan 90
Shan, Y.-P.:
"An Object-Oriented UIMS for Rapid Prototyping"
in: Diaper, D., et al. (Hrsg.): Human-Computer Interaction - Interact '90, Proceedings, IFIP TC 13, Cambridge, U.K., August 1989, S.633-638

Silbert 85
Silbert, J.; Belliardi, R.; Kamran, A.:
"Some Thoughts on the Interface Between User Management Systems and Application Software"
in: Pfaff, E. (Hrsg.): User Interface Management System, Proceedings, Workshop on UIMS, Seeheim (BRD), 1.-3.11.1983, Springer-Verlag Berlin Heidelberg, S.183-192

Simóes 88
Simóes, L.; Marques, J.A.; Guimaráes, N.; Carrico, L.; Sequeira, M.:
"IMAGES - an approach to an Object Oriented UIMS"
in: New directions for UNIX, Proceedings, Cascais, Portugal, 3.-7.10.1988, Buntingford, Herts. (EUUG autumn conference proceedings), S.143-157

Tanner 85
Tanner, P.P.; Buxton, W.A.S.:
"Some Issues in Future User Interface Management Systems (UIMS) Development"
in: Pfaff, E. (Hrsg.): User Interface Management System, Proceedings, Workshop on UIMS, Seeheim (BRD), 1.-3.11.1983, Springer-Verlag Berlin Heidelberg, S.67-79

Tanenbaum 90
Tanenbaum, A.S.
"Computer-Netzwerke"
Wolfram's Fachverlag 1990

Thomas 85
"Architecture for A User Interface Management System"
in: Pfaff, E. (Hrsg.): User Interface Management System, Proceedings, Workshop on UIMS, Seeheim (BRD), 1.-3.11.1983, Springer-Verlag Berlin Heidelberg, S.81-85

Tischendorf 89
Tischendorf, M.:
"Möglichkeiten der Kontrolle und Analyse von Umweltdaten durch Kopplung von Datenbank- und Expertensystemen"
in: Jaeschke, A., et al. (Hrsg.): Informatik im Umweltschutz, 4. Symposium 1989 Karlsruhe, Informatik Fachberichte 228, Springer-Verlag Berlin Heidelberg 1989, S.377-384

Trauboth 87
Trauboth, H.:
"Was kann die Informationstechnik für den Umweltschutz tun?"
in: Automatisierungstechnik at, 35. Jahrgang, Heft 11, 1987, S.431-442

UM 88
"Altlastenhandbuch, Teil I, Altlasten-Bewertung"
Ministerium für Umwelt Baden-Württemberg (Hrsg.),
Wasserwirtschaftsverwaltung, Heft 18, Dezember 1988

Weidemann 88
Weidemann, R.; Geiger, W.; Eitel, W.:
"Entwurf eines Expertensystemes zur Beurteilung von Abfallstoffen"
in: Jaeschke, A., et al. (Hrsg.): Informatikanwendungen im Umweltbereich,
2.Symposium 1987, Karlsruhe, November 1987, Informatik Fachberichte 170,
Springer-Verlag Berlin Heidelberg 1988, S.116-126

Weidemann 89
Weidemann, R., Geiger, W.:
"XUMA- Ein Assistent für die Beurteilung von Altlasten"
in: Jaeschke, A., et al. (Hrsg.): Informatik im Umweltschutz, 4. Symposium 1989
Karlsruhe, Informatik Fachberichte 228, Springer-Verlag Berlin Heidelberg 1989,
S.385-394

Weidemann 91
Weidemann, R.:
"Beurteilung von Altstandorten mit XUMA"
Workshop Erkundung, Bewertung und Sanierung ehemaliger Gaswerksstandorte,
21.Februar 1991

Weizsäcker 88
Weizsäcker, E.U.:
"Ganzheitlicher Umweltschutz - eine Herausforderung für Politik und Informatik"
in: Jaeschke, A., et al. (Hrsg.): Informatikanwendungen im Umweltbereich, 2.Symposium
1987, Karlsruhe, November 1987, Springer-Verlag Berlin Heidelberg 1988, S.1-7

Weng-Beckmann 88
Weng-Beckmann, U.:
"Eine objekt- und fensterorientierte Bedienoberfläche für graphikfähige Workstations
unter UNIX"
in: Unix in Deutschland: Jahrestagung der GUUG, 1988, Hannover, 27.-29.9.1988, S.282-
291

Wettstein 87
Wettstein, H.:
"Architektur von Betriebssystemen" - Kap. 2: Prozessorverwaltung
3. Auflage, Hanser 1987

Wisskirchen 90
Wisskirchen, P.:
"Object - Oriented Graphics"
Kap. 3.2: "The Model-View-Controller Triad", Symbolic Computation, Springer-Verlag
Berlin Heidelberg 1990, S.61-74

XWin 90
"The X Window System", Volume 0 - 7
O´Reilly & Associates, Inc., 1990