Werkzeuge und Systeme zur Unterstützung des Erwerbs und der objektorientierten Modellierung von Wissen

Von der Fakultät für Mathematik und Informatik der Universität Stuttgart zur Erlangung der Würde eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

vorgelegt von
Wolf-Fritz Johannes Riekert
aus Stuttgart

Hauptberichter:	Privatdozent Dr.-Ing. habil. K. Hanakata
Mitberichter:	Prof. Dr. G. Barth
Tag der Einreichung:	8. August 1986
Tag der Prüfung:	10. Oktober 1986

Institut für Informatik der Universität Stuttgart
1986

Werkzeuge und Systeme zur Unterstützung des Erwerbs und der objektorientierten Modellierung von Wissen

Wolf-Fritz Riekert
Forschungsgruppe INFORM
Institut für Informatik
Universität Stuttgart
Herdweg 51
D-7000 Stuttgart 1

Kurzfassung

Ausgehend von psychologischen Modellen kognitiver Prozesse werden die Vorgänge des Wissenserwerbs und der Wissensnutzung untersucht, die bei der inhaltlichen Verarbeitung von Fachinformationen ablaufen. Es werden die Formen von Wissen charakterisiert, die - als Voraussetzung oder als Gegenstand - für diese Verarbeitungsprozesse von Belang sind. Es wird beschrieben, wie sich dieses Wissen in Form einer sogenannten Wissensbasis auf einem Computersystem darstellen läßt und welche Funktionen ein solches System bei den Vorgängen der Wissensverarbeitung übernehmen kann. Illustriert wird dies durch die Implementation des wissensbasierten Informationssystems D&I.

Diese Ergebnisse werden ausgeweitet auf umfassendere und abstraktere Formen von Wissen. Es wird auf das Faktenwissen und das begriffliche Wissen eingegangen, das wissensverarbeitenden Systemen zugrundeliegt, und eine Systematik präsentiert, nach der sich das Wissen auf objektorientierte Wissensbasen abbilden läßt. Am vom Autor implementierten Wissenseditor ZOO wird gezeigt, wie diese Wissensbasen mit Hilfe graphischer Methoden benutzergerecht dargestellt und durch direkte Manipulation modifiziert werden können.

Inhaltsverzeichnis

Liste der Abbildungen

Vorwort

In den 60er Jahren wurde Computernutzung als ein Vorgang der *Datenverarbeitung* gesehen. In den 70er Jahren setzte ein Wandel der Betrachtungsweise ein, der zum Begriff der *Informationsverarbeitung* führte. Das vielleicht wichtigste Ergebnis der Computerwissenschaft der 80er Jahre ist die Auffassung von Software als einem Stück kodierten Wissens, die schließlich zum Begriff der *Wissensverarbeitung* führte.

Ist dieses Paradigma der Wissensverarbeitung überhaupt gerechtfertigt? Wo liegt der Vorteil gegenüber herkömmlichen Ansätzen?

Für die Verwendung des Begriffs Wissen im Zusammenhang mit modernen Computersystemen sprechen zumindest zwei Gesichtspunkte:

- Komplexe Wissensgebiete, die bislang menschlichen Experten vorbehalten waren, werden heute mit Hilfe von Computern bearbeitet. Computerprogramme können Aufgaben lösen, deren Erledigung von einem Menschen umfangreiches Wissen verlangt. Dies ist sicherlich ein Indiz für einen Zusammenhang zwischen Wissen und hochentwickelter Computersoftware. Dieser Aspekt der Wissensverarbeitung liegt dem Begriff *Expertensystem* ÄFeigenbaum 77] zugrunde.
- An der Benutzerschnittstelle von Computersystemen werden Wissensaspekte sichtbar: Die Art der Präsentation von Funktion und Inhalt moderner benutzerorientierter Computersysteme kommt der menschlichen Vorstellung zunehmend näher. An die Stelle von Programmierung, Dateneingabe und Endlosausgabe auf dem Zeilendrucker tritt ein Mensch-Computer-Dialog, der einen Wissensaustausch entsprechend dem Vorbild der zwischenmenschlichen Kommunikation bezweckt. Bei der Kommunikation zwischen Menschen sind Form, Detaillierungsgrad und thematischer Schwerpunkt der auszutauschenden Informationen abhängig vom Vorwissen der Kommunikationspartner. Wenn dieses Kommunikationsverhalten auf die Interaktion mit einem Computer übertragen werden soll, wird es erforderlich, eine Entsprechung dieses Vorwissens in einer sogenannten Wissensbasis nachzubilden und im Rechner zu repräsentieren. Dies führt zur Vorstellung einer *wissensbasierten Mensch-Computer-Kommunikation* [Herczeg et al. 85], wie sie von der Forschungsgruppe INFORM vertreten wird, welcher der Autor angehört.

In der vorliegenden Arbeit werden Methoden und Hilfsmittel zur Unterstützung des Erwerbs von Wissen und zur Modellierung von Wissen in einer Wissensbasis präsentiert. Es wird aufgezeigt, wie computerunterstützter Wissenserwerb in

vielen Fällen die klassischen Tätigkeiten der Datenerfassung und der Programmierung ersetzen kann. Es werden zwei Systemimplementationen vorgestellt, die den Erwerb und die Modellierung von Wissen ermöglichen: das wissensbasierte Informationssystem Digester und Informant (D&I) sowie der Wissenseditor ZOO. Es werden die angewandten Wissenstechniken beschrieben und es wird auf die Benutzerschnittstellenkonzepte eingegangen, die es einem Benutzer erlauben, das in einer Wissensbasis gespeicherte Wissen zu untersuchen und auf den aktuellen Stand zu bringen. Zur Repräsentation des Wissens und zur Programmierung der Benutzerschnittstelle wird in beiden implementierten Systemen die objektorientierte Sprache ObjTalk [Rathke C. 86] verwendet.

Das System D&I ist ein Dokumentationssystem, das einen Sachbearbeiter beim Sammeln und der inhaltlichen Verarbeitung von Fachinformationen unterstützt. Das System besteht aus zwei Komponenten, dem D&I Anwendungssystem und dem D&I Metasystem. Das Anwendungssystem ist für einen Sachbearbeiter gedacht und ermöglicht die Modellierung von Sachwissen in einer Faktenwissensbasis. Das D&I Metasystem soll einen Anwendungsexperten bei der Umrüstung des Anwendungssystems auf neue Anwendungsgebiete unterstützen. Es ermöglicht die Modellierung von Wissen einer abstrakteren Form, nämlich des konzeptuellen Wissens, das allem darstellbaren Sachwissen zugrundeliegt.

Während das System D&I auf ein spezielles Anwendungsgebiet, die wissensbasierte Dokumentation von Fachinformationen, zugeschnitten ist, ist der Wissenseditor ZOO ein universelles Werkzeug für wissensbasierte Systeme, die in der objektorientierten Sprache ObjTalk implementiert sind. ZOO ist ein System zur Visualisierung und Manipulation von Wissensbasen in einer zweidimensionalen graphischen Darstellung. Das System ist für einen Wissensingenieur gedacht, der wissensbasierte Systeme aufbaut und weiterentwickelt. Der Wissenseditor läßt sich aber auch mit einem Anwendungssystem integrieren und ermöglicht einem fortgeschrittenen Anwender die Veränderung des Anwendungsystems innerhalb bestimmter Grenzen.

Das Informationssystem D&I wurde im Rahmen des vom Ostasien-Institut (Bonn) geförderten Projekts "Digester und Informant" entwickelt. Besonderer Dank gilt Dr.-Ing. habil. Kenji Hanakata, der dieses Projekt begründete und leitete. Viele der hier beschriebenen Ergebnisse gehen auf Ideen und Perspektiven zurück, die er in das Projekt einbrachte. Dank gilt auch meinen ehemali-

gen Projektkollegen Feodora Csima-Herrmann und Wolfgang Schopper, die zusammen mit mir das System D&I entworfen und implementiert haben, sowie Christian Rathke, der mit der Entwicklung von ObjTalk das Fundament der hier beschriebenen Systeme legte.

Die Entwicklung des Wissenseditors ZOO wurde freundlicherweise von der Firma Triumph-Adler AG, Nürnberg im Rahmen eines Kooperationsvertrags mit der Universität Stuttgart finanziert. Die Forschungsgruppe INFORM bildete die wissenschaftlich-organisatorische Umgebung für diese Arbeiten. Professor Gerhard Fischer, der diese Forschungsgruppe ins Leben gerufen und in ihrer wissenschaftlichen Ausrichtung maßgeblich geprägt hat, hat mit seinem hilfreichen Rat viel zum Zustandekommen dieser Arbeit beigetragen. Dank gebührt auch meinen Kollegen in INFORM, die alle an den beschriebenen Forschungsergebnissen in irgendeiner Form beteiligt waren. Dies gilt vor allem für Franz Fabian, Michael Herczeg und wiederum Christian Rathke, deren konzeptuelle Arbeiten und Systemimplementationen die Grundlage für die Entwicklung des Wissenseditors bildeten.

Herrn Professor Barth danke ich für sein Interesse und die Bereitschaft zur Übernahme des Mitberichts. Herrn Professor Gunzenhäuser, der die hier vorgestellte Arbeit von Anfang an verantwortungsvoll begleitete und auf dessen Beistand ich stets zählen durfte, gilt mein ganz besonders herzlicher Dank für alle erhaltene Unterstützung.

1. Einleitung

Schriftliche Entscheidungshilfen müssen von extremer Kürze sein. Memoranden von mehr als 30 Schreibmaschinenzeilen betrachte ich schon als Roman.
(US-Innenminister William P. Clark)

1.1 Ausgangspunkt

Menschliches Handeln unterliegt heute einer wachsenden Zahl von äußeren Einflüssen. Auf vielen Tätigkeitsfeldern ist die Orientierung erschwert, so zum Beispiel in der Wirtschaft, in der Arbeitswelt, in der wissenschaftlichen Forschung, bei der Gesundheitsvorsorge oder in Angelegenheiten der Bürokratie. Betroffen sind Entscheidungsträger aus Politik, Verwaltung und Wirtschaft ebenso wie ein einfacher Sachbearbeiter in einem Büro oder ein Bürger bei seiner privaten Lebensgestaltung:

- Internationale Verflechtung bewirkt, daß auf dem Markt eine Vielfalt von Produkten aus aller Welt angeboten wird. Dem potentiellen Käufer stellt sich die Frage, welche davon seinen Ansprüchen entsprechen. Dem Produzenten stellt sich die Frage, für welche Produkte an welchem Ort ein Bedarf besteht. Was sind sinnvolle Kriterien für derartige Entscheidungen?

- Forscher sind mit einer Vielzahl von Arbeitsergebnissen aus aller Welt konfrontiert. Welche davon sind für die eigene Tätigkeit relevant und müssen berücksichtigt werden? Welche Forschungen und Entwicklungen sind es wert, von der staatlichen Förderungspolitik finanziell unterstützt zu werden?

- Innovationszyklen werden immer kürzer. In der Arbeitswelt führt dies zu bedeutsamen Veränderungen von Berufsbildern. Es ist schwierig, dem wissenschaftlichen und technologischen Fortschritt geistig zu folgen. Vertraute Berufe verschwinden, neue Arbeitsabläufe setzen sich durch. Beispielsweise in der Medizin steht ein Arzt vor dem Problem, während seiner täglichen Praxis immer wieder neue Diagnosemethoden und neue Therapien zu erlernen.

- Von der Bürokratie und den Organen der Gesetzgebung werden fortgesetzt neue Vorschriften und Regelungen erlassen, alle mit dem Ziel einer höheren Gerechtigkeit. In den Genuß dieser Gerechtigkeit kommt jedoch nur derjenige, welcher diese Paragraphen auch kennt und in Anspruch nimmt.

Jeder ist erreichbar geworden für diese Einflüsse und kann sich ihnen nur schwer entziehen. Auch Vorgänge, die sich weitab ereignen, haben Auswirkun-

gen auf Individuen und menschliche Gemeinschaften. Der Mensch als Privatwesen oder Staatsbürger, als Erwerbstätiger oder als Konsument muß sein Verhalten immer mehr an globalen Faktoren ausrichten.

Der modernen Kommunikationstechnik kommt hierbei eine Doppelfunktion zu: Zum einen verstärkt sie die Wirkung der genannten äußeren Einflüsse. Die Medien übermitteln täglich eine Flut von Informationen, die sich praktisch nicht mehr bewältigen läßt. Die Orientierung wird erschwert, nicht durch ein Zuwenig, sondern durch ein Zuviel an Information. Zum andern aber liefern gerade die Kommunikationsmedien die Indikatoren, die nötig sind, um sich in der heutigen Welt zurechtzufinden. Wer diese Indikatoren nicht berücksichtigt, gerät bei der Wahrnehmung seiner Interessen ins Hintertreffen.

1.2 Bedeutung von Sachwissen

Die Fähigkeit, in einem Problembereich angemessene Entscheidungen treffen zu können, erfordert häufig umfangreiche Sachkenntnisse:

- Für die Markteinführung eines Produktes ist die genaue Kenntnis des Marktes mindestens genauso bedeutsam wie beispielsweise die Qualität des Produktes.
- In den Organen der staatlichen Exekutive ist das politische Handeln der Verantwortlichen immer weniger durch politische Programme bestimmt als vielmehr durch sogenannte Sachzwänge. Ob diese auch gerechtfertigt sind, läßt sich nur mit Sachwissen entscheiden.
- In Forschung und Entwicklung sind Genialität und Erfindergeist nur dann fruchtbar, wenn sie beim neuesten Stand der wissenschaftlichen Erkenntnis ansetzen.
- Sachwissen ist die Grundlage für alle Aufgaben der Beratung und der Interessenvertretung. Beispielsweise ist für einen Anwalt heutzutage die Kenntnis von Paragraphen und Musterurteilen mindestens genauso wichtig wie Fähigkeiten der Redegewandtheit oder der Argumentationskunst.

Diese Wissensgebiete sind derart umfangreich, daß sie Lernfähigkeit und Merkfähigkeit eines einzelnen Menschen überfordern. Herkömmlich organisierte oder rechnerbasierte Informationsablagesysteme dienen dazu, das Langzeitgedächtnis des Menschen zu entlasten. Es werden *Experten* benötigt, die diese Wissensbestände verwalten und sie den Ratsuchenden verfügbar machen.

1.3 Experten und Systeme

Mit einem komplexen Wissensgebiet konfrontiert, ist der einzelne also auf das Urteil von Experten angewiesen: Steuerberater, Rechtsanwälte, Verbraucherberater, Berufsberater, Treuhänder, Sachverständige und Ärzte. Aber auch die Experten selbst haben bereits Schwierigkeiten, ihre angestammten Fachgebiete zu überschauen und müssen Teilaufgaben an Spezialisten delegieren. In beiden Fällen geht leicht die globale Sicht auf das zu lösende Problem verloren. Der von einer Entscheidung unmittelbar Betroffene überläßt seinen Entscheidungsspielraum anderen und ist sich der Grundlagen der von ihm zu verantwortenden Handlung nicht mehr voll bewußt.

Beratung durch Experten verursacht Kosten. Wenn diese Kosten höher sind als der zu erwartende Nutzen, sprechen wirtschaftliche Gründe dagegen, die Möglichkeit einer Beratung wahrzunehmen.

Die mangelnde Nachvollziehbarkeit der erhaltenen Antworten, ein verengter Handlungsspielraum und hohe Kosten sind daher häufig die Ursache, wenn die Beratung durch Experten auf geringe Akzeptanz stößt. Es stellt sich die Frage, ob ein Entscheidungsprozeß, der umfangreiches Sachwissen voraussetzt, auf andere Weise als durch die Einbeziehung von Experten unterstützt werden kann.

In amerikanischen Forschungsstätten (MIT, CMU, SRI, Xerox PARC[1]) wurde in den vergangenen Jahren eine Reihe von Softwaresystemen entwickelt, mit denen das Ziel verfolgt wird, auf einem begrenzten Wissensgebiet die Aufgaben eines Gebietsexperten zu lösen. Diese Systeme werden als *Expertensysteme* oder *wissensbasierte Systeme* bezeichnet.

Die Fähigkeit von Expertensystemen, aus einer Vielzahl von Fakten Folgerungen zu ziehen und Probleme zu lösen, reicht jedoch alleine nicht aus, um brauchbare Entscheidungshilfen anzubieten. Damit der Benutzer nicht in die Abhängigkeit von einer Maschine gerät, muß das System vor allen Dingen einfach steuerbar und leicht durchschaubar sein. Für die Entwicklung eines wissensbasierten Systems, mit dem ein Mensch selbständig große Mengen von Informationen bewältigen und umfangreiche Wissensbestände verwalten kann, gibt

---

[1]MIT: Massachusetts Institute of Technology, CMU: Carnegie Mellon University, SRI: Stanford Research Institute, Xerox PARC: Xerox Palo Alto Research Center

es also zwei globale Gestaltungskriterien: Das System soll zum einen imstande sein, umfangreiches Expertenwissen in sich aufzunehmen und in bestimmten Situationen anzuwenden. Zum andern ist es erforderlich, daß der Benutzer die Funktionsweise des Systems verstehen und nach seinen Wünschen beeinflussen kann. Aus diesem Grunde kommt dem Design der Benutzerschnittstelle eines wissensbasierten Systems mindestens ebenso große Bedeutung zu wie der Darstellung von Expertenwissen in einer maschinell interpretierbaren Form.

1.4 Überblick

Die vorliegende Arbeit befaßt sich sowohl mit Fragen der Wissensdarstellung als auch mit Fragen der Benutzerschnittstellengestaltung für wissensverarbeitende Systeme. Es werden zwei Arten von Anwendungsfällen in theoretischer Form und am Beispiel praktischer Softwarelösungen untersucht:

- Ausgehend von psychologischen Modellen kognitiver Prozesse werden die Vorgänge des Wissenserwerbs und der Wissensnutzung untersucht, die bei der inhaltlichen Verarbeitung von Fachinformationen ablaufen. Es werden die Formen von Wissen charakterisiert, die - als Voraussetzung oder als Gegenstand - für diese Verarbeitungsprozesse von Belang sind. Es wird beschrieben, wie sich dieses Wissen in Form einer sogenannten Wissensbasis auf einem Computersystem darstellen läßt und welche Funktionen ein solches System bei den Vorgängen der Wissensverarbeitung übernehmen kann. Illustriert wird dies durch die Implementation des wissensbasierten Informationssystems D&I.

- Diese Ergebnisse werden ausgeweitet auf umfassendere und abstraktere Formen von Wissen. Es wird auf das Faktenwissen und das begriffliche Wissen eingegangen, das wissensverarbeitenden Systemen zugrundeliegt, und eine Systematik präsentiert, nach der sich das Wissen auf objektorientierte Wissensbasen abbilden läßt. Am vom Autor implementierten Wissenseditor ZOO wird gezeigt, wie diese Wissensbasen mit Hilfe graphischer Methoden benutzergerecht dargestellt und durch direkte Manipulation modifiziert werden können.

Im Kapitel 2 gehen wir von einer konkreten Aufgabenstellung aus, dem Arbeitsablauf eines Dokumentations- und Beratungsinstituts. Die Anforderungen an ein Computersystem zur Unterstützung derartiger Tätigkeiten werden aus der Sicht der Benutzer formuliert. Daraus werden notwendige Eigenschaften eines wissensverarbeitenden Systems abgeleitet.

Kapitel 3 befaßt sich mit Techniken der Wissensverarbeitung. Es wird eine Klassifizierung der verschiedenen Formen von Wissen vorgenommen. Unter-

schiedliche Techniken der Wissensverarbeitung und der Wissensdarstellung werden miteinander verglichen. Es wird eine objektorientierte Repräsentationstechnik für die Darstellung von Wissen und die Programmierung von Benutzerschnittstellen vorgeschlagen.

In Kapitel 4 wird die Funktionsweise des Systems D&I beim Erwerb und bei der Nutzung von Fachwissen dargestellt. Die im System darstellbaren Arten von Fakten und Begriffen werden beschrieben und erläutert. Der Schwerpunkt in diesem Kapitel liegt auf dem Vorgang der Modellierung von gebietsspezifischem Faktenwissen.

In Kapitel 5 wird anhand der Metakomponente des Systems D&I darüberhinaus gezeigt, wie ein Anwendungsexperte die Darstellungskapazität des Systems D&I erweitern kann, indem er neues begriffliches Wissen über das Anwendungsgebiet in das System einbringt. Die Interaktion mit der Metakomponente geschieht dabei auf dieselbe Weise wie die Interaktion mit dem D&I Grundsystem.

Der anwendungsneutrale Wissenseditor ZOO ist in Kapitel 6 beschrieben. Der Wissenseditor ermöglicht die Modellierung von Wissen aus einem beliebigen Problembereich in einer objektorientiert repräsentierten Wissensbasis[2]. Mittels einer objektorientierten Benutzerschnittstelle[3] können die Fakten und Konzepte eines in ObjTalk implementierten wissensbasierten Systems sichtbar gemacht und verändert werden.

---

[2]im Sinne von [Rathke C. 86]

[3]Der Begriff der objektorientierten Benutzerschnittstelle stammt aus [Smith et al. 82].

2. Problemstellung

Die eigentlich neue und für die fünfte Computergeneration typische Aufgabe sind die wissensverarbeitenden Informationssysteme. (Dr. Marx, BMFT)

Der Bezeichnung "Expertensysteme" liegt die Anschauung zugrunde, daß diese Systeme in erster Linie über Expertenwissen und erst in zweiter Linie, lediglich implementationsbedingt, über Daten verfügen. Eine Folge dieser Betrachtungsweise ist der Übergang von der Datenverarbeitung klassischer Prägung hin zur *Wissensverarbeitung.*[4]

Die Verarbeitung von Wissen erfordert stets die Lösung von drei Teilaufgaben: Wissenserwerb, Wissensrepräsentation und Wissensnutzung. *Wissenserwerb* ist der Vorgang, Informationen, die von außerhalb des wissensverarbeitenden Systems stammen, inhaltlich aufzuarbeiten und in systemeigene Strukturen zu überführen. Diese internen Strukturen machen das Wissen des Systems aus. Das gesamte strukturierte Wissen, über das ein wissensverarbeitendes System verfügt, bezeichnen wir als seinen *Wissensbestand.* Die Art der Notation dieses Wissens ist eine Frage der *Wissensrepräsentation. Wissensnutzung* schließlich ist der Vorgang, aus dem Wissensbestand wieder Informationen abzuleiten und Schlüsse zu ziehen, die in einer gegebenen Situation anwendbar sind.

2.1 Arbeitsabläufe bei der Wissensverarbeitung

In diesem Abschnitt wollen wir uns den Arbeitsabläufen zuwenden, die bei der Verarbeitung von Wissen anfallen. Deutlich kommen diese zum Ausdruck in einer Dokumentations- und Informationsstelle. Es gibt eine Vielzahl derartiger Einrichtungen, die hier im einzelnen nicht aufgezählt werden. Als Beispiele seien lediglich die an die Gesellschaft für Information und Dokumentation (GID) angeschlossenen Fachinformationszentren (FIZ), die technische Informationsbibliothek Hannover (TIB) oder das Ostasien-Institut in Bonn (OAI) genannt.

---

[4]Datenverarbeitung ist im wesentlichen die Anwendung von algebraischen Transformationen auf alphanumerische Informationen. Textverarbeitung ist dementsprechend die Anwendung von syntaktischen und graphischen Transformationen auf Textdokumente. Für den Umgang mit Sach- und Fachwissen sind nun Methoden zur Wissensverarbeitung gefordert. Es wird sich zeigen, daß es sich dabei um semantische Operationen auf Wissensstrukturen handelt.

In diesen Instituten werden zu einem bestimmten Wissensgebiet fortlaufend Informationen beschafft und gesammelt. Die Quelltexte müssen von Fachkräften gesichtet und bis zu einem gewissen Grad verstanden werden, damit sie nach bestimmten Kategorien klassifiziert und abgelegt werden können. Ausgewählte Dokumente werden von Gebietsexperten verarbeitet, indem relevante Teile ihres Inhalts in strukturierte Sachinformationen umgesetzt werden. Diese strukturierte Sachinformation wird herkömmlichen Karteien oder auch immer häufiger Datenbanken zugeführt, welche dadurch stets den aktuellen Stand des Wissens über das betreffende Gebiet dokumentieren.

Auf dem Gebiet der Information werden passive und aktive Dienste angeboten. Zum einen geben die Gebietsexperten in einem Dokumentations- und Informationszentrum Antworten auf Sachfragen, die von außen an sie gestellt werden, bieten in konkreten Fällen Entscheidungshilfen an und beschaffen die der Antwort zugrundeliegenden Quellen. Zum andern werden Zusammenfassungen, verdichtende Analysen und tabellarische Übersichten eigenverantwortlich erstellt und Informationsdiensten, Wirtschaftsunternehmen und politischen Entscheidungsgremien offeriert.

In den Dokumentations- und Informationseinrichtungen erscheinen die Vorgänge der Wissensverarbeitung sozusagen in Reinkultur. Eingangsgrößen sind Informationen in verschiedenen Erscheinungsformen. Am Ende der Wissensverarbeitung stehen wiederum Informationen in veränderter Auswahl und Anordnung.

Aber auch in anderen Tätigkeitsfeldern haben wissensverarbeitende Vorgänge einen hohen Stellenwert. Sie sind jedoch dort Teil eines umfassenderen Prozesses der Problemlösung. Die Ergebnisse der wissensverarbeitenden Vorgänge fließen unmittelbar ein in Entscheidungen und Handlungen. Beispiele für solche Tätigkeitsfelder sind:

- Wissenschaft und Forschung, beispielsweise bei der Vorbereitung empirischer Untersuchungen,
- Planungsaufgaben, wobei die Spanne von Entwicklungsprojekten in der Industrie bis beispielsweise zur Planung eines Eigenheims durch einen Privatmann reicht,
- Beratung und Interessenvertretung, etwa durch Anwälte und Steuerberater,
- Diagnose und Therapie von Krankheiten in der Medizin,
- Untersuchung und Behebung von Geräteausfällen in der Technik,
- Konstruktion, zum Beispiel der Entwurf von integrierten Schaltkreisen oder von Maschinenteilen,

- Die arbeitsteiligen Tätigkeiten, die in einem Büro anfallen und die Fachkenntnisse, Organisationswissen und Kommunikationsfähigkeit erfordern,
- Systemanalyse und Softwareentwicklung durch ein Softwarehaus.

Wir werden zunächst von den wissensverarbeitenden Vorgängen bei den Aufgaben der Dokumentation und Information ausgehen. Später werden wir die Ergebnisse verallgemeinern, so daß sie auch auf die anderen genannten wissensverarbeitenden Prozesse angewandt werden können.

2.2 Anforderungen an ein wissensverarbeitendes System

Wissensverarbeitung hat stets den Zweck, Probleme aus einem Wissensgebiet zu lösen: Im günstigen Fall sind ein Ausgangszustand, ein Ziel und eine Menge von Wissensquellen gegeben. Die Aufgabe besteht nun darin, aus diesen Wissensquellen Aktionsfolgen abzuleiten, die vom Ausgangszustand zum Zielzustand führen. Im ungünstigen Fall, der aber in der Praxis sehr häufig auftritt, ist das Problem schlecht strukturierbar [Newell 69; Simon 73] und es müssen diese Zustände erst noch definiert werden und die relevanten Wissensquellen erst noch gefunden werden. Hieraus ergeben sich die Aufgaben eines Experten, der einen Wissensbestand verwaltet. Dieser muß alle ihm zugänglichen Informationen aus seinem Fachgebiet sammeln, inhaltlich aufarbeiten, klassifizieren, umgruppieren, verdichten und selektieren, damit letztlich sachgerechte Entscheidungen getroffen werden können. Ein System zur Wissensverarbeitung muß geeignet sein, alle diese Tätigkeiten angemessen zu unterstützen. Die Anforderungen, die sich hierbei stellen, werden in den folgenden Abschnitten unter den Gesichtspunkten des Wissenserwerbs, der Wissensnutzung und der Wissensrepräsentation präzisiert.

2.2.1 Wissenserwerb

Für jedes Sachgebiet gibt es eine Vielzahl von Informationsquellen. Zunächst gibt es die verschiedenen Formen von gedruckt vorliegenden Publikationen: Zeitschriften, Fachblätter, amtliche Mitteilungen, Produktinformationen, Zusammenfassungen usw. Eine zweite Art der Informationsbeschaffung beruht auf dem persönlichen Kontakt: Recherchen, Besichtigungen, Gespräche mit Fachleuten gehören in diese Kategorie. Schließlich sind alle Medien der elektronischen Kommunikation zu nennen: Rechnernetze, Datenträgertausch, Teletext, Bildschirmtext, Bildplatte usw.

Insgesamt ergibt sich eine wahre Flut von inhomogenen Informationen aus den unterschiedlichsten Quellen. Die Informationen sind schwer überschaubar, schwer einzuordnen und schwer zu bewerten. In der Regel haben diese Informationen die folgenden Eigenschaften:

- Sie haben zumeist verschiedene Aspekte und sagen zu verschiedenen Unterthemen etwas aus. Es gibt keine allgemeingültige Regel, wie die Informationen systematisch klassifiziert werden können.
- Die eingehenden Informationen verändern den Wissensbestand inkrementell. Logisch zusammengehörige Teilinformationen können zu gänzlich unterschiedlichen Zeitpunkten eintreffen. Es ist erforderlich, daß solche Einzelinformationen zusammenfinden, damit sie sich Stück für Stück zu einem umfassenden Bild fügen.
- Nicht alle eintreffenden Informationen sind in ihrer vollen Länge relevant für die beabsichtigte Nutzung, das Lösen von Problemen im betrachteten Sachgebiet. Es ist sinnvoll, von vornherein alles Unwesentliche auszusondern, damit der Wissensbestand nicht ausufert und die Wissensnutzung nicht erschwert wird.
- Häufig enthalten mehrere Informationsquellen zum Teil dieselben Inhalte. Dies erhöht die Verläßlichkeit der Teilinformation, stellt aber auch Redundanz dar, welche die Nutzung der Information behindert. Es ist erforderlich, diese Redundanzen aufzudecken und herauszufiltern.
- Neu eintreffende Informationen können im Widerspruch zu bereits bekannten Informationen stehen. Wenn die neuen Informationen fundierter erscheinen als die bisher bekannten, muß es möglich sein, die betreffenden Teile des Wissensbestandes zu revidieren.

Wissenserwerbsvorgänge sind durchaus nicht trivial. Es genügt nicht, alles anfallende Informationsmaterial einfach additiv zusammenzutragen, da sich sonst ein unüberschaubarer Wissensbestand ergibt, der sich nur schwer nutzen läßt. Die erworbenen Informationen müssen miteinander in Verbindung gebracht werden, mögliche Zusammenhänge müssen hergestellt und Widersprüche müssen aufgespürt und aufgelöst werden.

In dieser Arbeit wird unter dem Begriff *"Wissenserwerb"* oder *"Wissensakquisition"* sowohl die *Aneignung von Wissen durch Menschen* als auch die *Übertragung von Wissen auf Computersysteme*[5] verstanden. Der Vorgang der Übertragung von Wissen auf ein Computersystem wird im folgenden auch als *Mo-*

---

[5]In Analogie müßte es eigentlich "Aneignung von Wissen durch Computersysteme" heißen. Da dieser Begriff jedoch nicht üblich ist, folgt der Autor der Bezeichnungsweise von R. Davis [Davis, Lenat 82], der zwar allgemein den Begriff "knowledge acquisition" verwendet, aber von "knowledge transfer" und "teaching" spricht, sobald er die beteiligten Instanzen - Experte und Computersystem - nennt.

dellierung von Wissen bezeichnet. Wissenserwerb ist nicht mit Lernen an sich gleichzusetzen. Lernen umfaßt außer dem Erwerb von Kenntnissen auch das Einüben von Fähigkeiten. Während das erste ein Prozeß der Bewußtmachung von Fakten und Prinzipien ist, stellt das zweite die Aneignung von eher unbewußten Handlungsabläufen dar. Dieser zweite Aspekt des Lernens, der sehr anschaulich von Fischer, Burton und Brown [Fischer et al. 78] beschrieben wird, soll im folgenden nicht im Vordergrund stehen. Ganz außer Betracht bleiben wird die maschinelle Unterstützung menschlicher Wissenserwerbsvorgänge durch rechnerunterstütztes Lernen [Gunzenhäuser 84].

In [Michalski, Carbonell, Mitchell 83] wird Wissenserwerb definiert als das Lernen neuer symbolischer Information zusammen mit der Fähigkeit, diese Information auf effektive Weise anzuwenden. Es werden abhängig vom Aufwand des Lehrers und des Lernenden verschiedene Strategien unterschieden: Auf der einen Seite eher passive, bei denen eine Lehrperson den Lehrstoff aufbereitet und dem Lernenden vermittelt und im Extremfall regelrecht "eintrichtert", auf der anderen Seite die aktiven Strategien, bei denen der Lernende mit Hilfe von Analogieschlüssen oder anderen Schlußtechniken das Wissen selbständig aus vorgelegten Beispielen oder gar eigenen Beobachtungen ableitet. Die hier vorgestellten Wissenserwerbskomponenten von Softwaresystemen haben eher passiven Charakter, da zur Wissensübertragung ein Gebietsexperte als "Lehrer" erforderlich ist; sie sind aber auch aktiv in dem Sinne, daß sie ihren menschlichen Bediener beim Einbringen des Wissens unterstützen und das erworbene Wissen selbständig in eine Wissensbasis einbetten und konsistent halten.

2.2.2 Wissensnutzung

Das Ziel jeglicher Wissensverarbeitung ist die Nutzung des erworbenen Wissens. Der typische Fall der Nutzung besteht darin, aus dem vorhandenen Wissensbestand Informationen abzuleiten. Daraus ergeben sich folgende Anforderungen:

- Auf erworbene Informationen soll irgendwann gezielt zurückgegriffen werden. Bei Kenntnis von Teilen solcher Informationen muss es möglich sein, die zugrundeliegende Gesamtinformation wieder abzurufen. Diese besteht meist aus dem Inhalt von mehreren abgelegten Quellinformationen.

- Für die Nutzung sind nur Teilaspekte der verarbeiteten Einzelinformationen interessant. Falls alle Informationsquellen mit einem Bezug zu einer bestimmten Entscheidung in voller Länge vorlägen, wäre es unmöglich, diese innerhalb vertretbarer Zeit zu berücksichtigen. Es muß möglich sein, die für eine anstehende Entscheidung relevanten Teile der Informationen zu erkennen und eine Auslese zu treffen.

- Viele der erworbenen Informationen enthalten implizites Wissen, das nur bei entsprechender Expertise erschlossen werden kann. Aus implizitem Wissen ableitbare Informationen müssen für die Nutzung ebenso verfügbar sein, wie alle anderen Informationen auch.

Beim Vorgang der Wissensnutzung werden die Informationen unter neuen Blickwinkeln gesehen. Das Wissen wird oft in einer gänzlich anderen Form verlangt als in den zugrundeliegenden Quellinformationen. Nicht die Quellinformationen in ihrer ursprünglichen *Form*, sondern ausgewählte Aspekte ihrer *Inhalte* sind von Interesse. Die Suche nach Informationen geschieht ebenfalls nach inhaltlichen Kriterien.

Wissensnutzung ist ein sehr allgemeiner Begriff, der vielfältige Formen der Verwendung von Wissen umfaßt:

- Das Wissen wird zum Lösen von Problemen gebraucht. Ein *Anwendungssystem* bedient sich des vorhandenen Wissens.
- Die Wissensnutzung besteht im unmittelbaren Abruf von Informationen durch einen Benutzer. Ein *Informationssystem* greift auf das gespeicherte Wissen zu.

Wir werden uns im folgenden vorwiegend mit der zweiten Art der Wissensnutzung beschäftigen, wobei *Navigationswerkzeuge*[6] [Fischer 83] und Systeme zur *Visualisierung*[7] [Böcker 86] von Wissensstrukturen im Vordergrund stehen. Beispiele für Navigationswerkzeuge sind etwa das Informationssystem ZOG [Robertson et al. 81] oder das Retrieval-System RABBIT [Tou, Williams 82]. Entwicklungen auf dem Gebiet der Visualisierung von Wissensstrukturen gibt es noch sehr wenige: Ansätze finden sich in KÄSTLE [Nieper 83], einem System zur Visualisierung von Lisp-Strukturen, sowie im LOOPS-Browser[8] [Bobrow, Stefik 81] und im ObjTalk-Browser [Rathke C. 86], die beide zur graphischen Darstellung einer hierarchischen Ordnung von Begriffen dienen.

---

[6]Navigationswerkzeuge ermöglichen die "Fortbewegung" in einer Wissensstruktur: Unter Navigation wird der Übergang von einer auf dem Bildschirm dargestellten Informationseinheit zu einer benachbarten Informationseinheit verstanden.

[7]Der Begriff Visualisierung wird hier und im folgenden in einem relativ engen Sinn gebraucht: Gemeint ist immer eine Darstellung in graphischer Form. In einem erweiterten Sinn kann selbstverständlich auch jede textuelle Repräsentation zur Visualisierung von Wissensstrukturen dienen.

[8]to browse (englisch) = sich umsehen. Ein "Browser" ist ein Softwarewerkzeug zur Inspektion von netzartigen Strukturen.

2.2.3 Wissensrepräsentation

Informationen treffen nicht notwendigerweise zu dem Zeitpunkt ein, zu dem sie für eine Entscheidung relevant sind. Sie müssen daher in einer geeigneten Form abgelegt werden. Da beim Vorgang der Wissensnutzung die Inhalte der Informationen wieder aufgefunden und nach verschiedenen Kriterien selektiert werden sollen, ist es erforderlich, das erworbene Wissen in eine strukturierte Darstellung umzuformen. Die Art dieser Darstellung ist gegeben durch die gewählte Technik der Wissensrepräsentation. Diese muß den Anforderungen gerecht werden, die von drei Seiten gestellt werden:

1. Beim Wissenserwerb ist eine geringe Transformationsdistanz zwischen dem Format der Quellinformationen und der Darstellungsart des archivierten Wissens wünschenswert.
2. Für die Wissensnutzung soll das Wissen in einer inhaltlich vorverarbeiteten Form vorliegen, aus der schnell und auf einfache Weise Antworten auf ein breites Spektrum von Fragen abgeleitet werden können.
3. Die gewählte Wissensrepräsentationstechnik muß auf das Datenverwaltungssystem abgestimmt sein, das für die Speicherung des Wissensbestands verwendet wird.

Es wird deutlich, welch zentrale Rolle die Wissensrepräsentation innerhalb der Wissensverarbeitung spielt. Die Darstellungsart des Wissens bei seiner Archivierung bestimmt weitgehend, welcher Arbeitsaufwand vorsorglich beim Erwerb des Wissens und welcher bedarfsweise bei dessen Nutzung anfällt. Maßgeblich hierfür sind die folgenden Kriterien für die Repräsentation des Wissens:

- *Granulierung der Information:* In welche sinnvollen elementaren Wissenseinheiten lassen sich die Informationen zerlegen?
- *Darstellung:* Welche natürlichen Schemata liegen den darzustellenden Sachverhalten zugrunde?
- *Organisation des Informationsablagesystems:* Welche Ordnungskriterien sind geeignet zur Einordnung der Informationen?
- *Sprachunabhängigkeit:* Wie lassen sich Sachinformationen unabhängig von ihrer ursprünglichen natürlichsprachlichen Form ablegen?
- *Konsistenz:* Wie wird gewährleistet, daß sich alle Teile des Archivs auf demselben aktuellen Stand befinden?

Bei der manuellen Wissensverarbeitung sind die Möglichkeiten der Wissensrepräsentation durch das verwendete Medium (Karteien und Akten) sehr einge-

schränkt. Durch den Einsatz von Rechnern bei der Wissensverarbeitung und durch die Verwendung von Datenbanken und anderen maschinellen Informationsablagesystemen ergibt sich bei der Gestaltung von Wissensrepräsentationstechniken ein weiter Freiraum, welcher derzeit noch bei weitem nicht ausreichend erforscht ist.

3. Techniken der Wissensverarbeitung

S understands knowledge K if S uses K whenever appropriate. [Moore, Newell 74]

3.1 Psychologische Aspekte der Wissensverarbeitung

Psychologische Modelle kognitiver Prozesse geben nicht nur Aufschluß über intellektuelle Leistungen des Menschen. Derartige Modelle sind auch beim Entwurf wissensverarbeitender Computersysteme von Nutzen.[9] Ein solches Modell, das sich auf die Probleme der Wissensverarbeitung anwenden läßt, ist in der *Psychologie der Intelligenz* [Piaget 72] von Piaget zu finden. Piaget unterscheidet dort zwei Arten der Auseinandersetzung von Individuen mit ihrer Umwelt: *Assimilation* und *Akkommodation.*

Im biologischen Sinn ist *Assimilation* ein Stoffwechselvorgang, bei dem körperfremde Substanzen in körpereigene umgewandelt und in den eigenen Organismus eingebaut werden. Die Assimilation von Kohlenstoff mit Hilfe der Photosynthese bei Pflanzen oder auch die Aufnahme und Verdauung von Nahrung durch Menschen und Tiere sind prototypische Beispiele für den Assimilationsvorgang. Das Individuum nimmt also Stoffe auf, die von außen stammen, verarbeitet sie und macht sie den Stoffen, aus denen es selbst besteht, der Struktur nach ähnlich. Demgegenüber bezeichnet *Akkommodation* den Vorgang der Anpassung an die Umwelt, bei dem die eigenen Assimilationsmechanismen verändert werden. Unter Akkommodation im biologischen Sinn versteht man die Anpassung des Auges an wechselnde Gegenstandsweiten durch Kontraktion der Linse. Bei der Akkommodation verändert also das Individuum seine eigenen Stoffe und Strukturen, also Bestandteile seiner selbst. Ziel der Akkommodation ist es, mit der Umwelt besser zurechtzukommen.

Piaget wendet diese Begriffe auf die geistige Auseinandersetzung des Menschen mit seiner Umwelt an. Für J. Moore und A. Newell als typische Repräsentanten der Künstlichen-Intelligenz-Forschung ist die Fähigkeit zu Akkommodation und Assimilation eine notwendige Eigenschaft eines Programmes, das Probleme verstehen können soll [Moore, Newell 74].

---

[9] Umgekehrt ergibt sich für Psychologen die Möglichkeit, Modelle menschlicher wissensverarbeitender Prozesse in der Computersimulation auf ihre Schlüssigkeit zu überprüfen, sobald maschinelle wissensverarbeitende Systeme verfügbar sind.

Auf die Vorgänge der Wissensverarbeitung bezogen ist Assimilation ein Wissenserwerbsvorgang, bei dem Informationen, die von außen stammen, in eigene Wissensstrukturen eingepaßt werden. Akkommodation hingegen ist die Anpassung der eigenen Wissenserwerbsmechanismen und Wissensstrukturen an neue Erfordernisse. Bei beiden Vorgängen spielt Wissen eine zweifache Rolle; zum einen haben sie Wissen zum Gegenstand, zum andern laufen sie wissensgesteuert ab. Es ergibt sich die Möglichkeit, Wissen danach zu klassifizieren, welche Rolle es bei Assimilation und Akkommodation spielt. Dabei zeigt es sich, daß drei verschiedene Arten von Wissen unterschieden werden können: Sachwissen, konzeptuelles Wissen und Metawissen.[10] In den folgenden Abschnitten werden diese drei Arten von Wissen in ihrem Verhältnis zu den Vorgängen der Assimilation und der Akkommodation charakterisiert.

3.1.1 Assimilation von Sachwissen

Sachwissen oder Faktenwissen ist Wissen um konkrete Sachverhalte aus einem Problemraum. Es zeigt sich in der Kenntnis von Objekten, deren Eigenschaften und wechselseitigen Beziehungen. Sachwissen hat elementare Aussagen zum Gegenstand, z.B. "Mein Auto steht in der Garage" oder "Stuttgart ist eine Großstadt". Sachwissen ist eine Art von Wissen, die sich sehr schnell verändern kann und häufig nur befristete Gültigkeit besitzt.

Sachwissen in großem Umfang fällt beispielsweise in einer Versicherungsgesellschaft an. Die Kundendaten, die individuellen Bedingungen der abgeschlossenen Versicherungsverträge und die Verläufe von Schadensfällen sind Gegenstand dieses Sachwissens. Da die Fülle dieses Sachwissens das Gedächtnis eines Menschen überfordert, werden die entsprechenden Informationen in Form von Akten bzw. von Computerdateien abgelegt.

Der Erwerb von Sachwissen ist ein Assimilationsvorgang. Man versucht Informationen, die von außen stammen, zu begreifen und den eigenen Wissensstrukturen einzupassen. Gelingt einem dies, so hat man den zugrundeliegende Sachverhalt verstanden. Gelingt dies nicht, so kann es hierfür zwei Ursachen geben:

---

[10] R. Davis spricht in diesem Zusammenhang von sogenannten "Levels of Knowledge" und bezeichnet die drei angesprochenen Arten von Wissen als "object-level knowledge", als "meta-level knowledge" und als "second order meta-level knowledge" [Davis, Lenat 82, Seite 399ff.].

1. Die bisher vorhandenen eigenen Wissensstrukturen sind nicht dafür geeignet, das Sachwissen aufzunehmen. Die Sachinformation konnte nicht "begriffen" werden, weil die Begriffe fehlten, nach denen das Wissen eingeordnet werden kann.
2. Die Begriffe sind zwar vorhanden, aber es fehlt an der Fähigkeit, diese Begriffe auf den aktuellen Fall anzuwenden.

Der Erwerb von Sachwissen ist also nur möglich, wenn der Lernende über Wissen einer höheren Stufe verfügt. Es ist dies das sogenannte begriffliche oder konzeptuelle Wissen:

Konzeptuelles Wissen ist Wissen über den Umgang mit Sachwissen. Im konzeptuellen Wissen liegen die Schemata, die erforderlich sind, um Sachwissen zu erschließen, einzuordnen, zu memorieren und zu nutzen. Nach außen hin ist diese Art von Wissen erkennbar als die Fähigkeit, Problemstellungen aus einem Sachgebiet aufzunehmen und gegebenenfalls zu lösen.

Der Lernende erkennt also in den Erscheinungen der Außenwelt Ausprägungen ihm bekannter begrifflicher Schemata und kann daher das Ergebnis seiner Wahrnehmungen in die interne Struktur seines Sachwissens einordnen. Dieses Sachwissen kann beim Lösen von Problemen genutzt werden und bestimmt so das künftige Verhalten des Wissensträgers.

Ein Beispiel hierfür ist die Tätigkeit eines Sachbearbeiters in einer Versicherung. Das konzeptuelle Wissen des Sachbearbeiters besteht in der Kenntnis der verschiedenen Typen von Geschäftsvorgängen. Jeder Vorfall, beispielsweise die Schadensmeldung eines Kunden, wird in ein geeignetes Schema eingepaßt, in diesem Fall vielleicht in den "Geschäftsvorfall KfZ-Schadenregulierung". Damit lassen sich die Angaben des Kunden in die gebräuchliche Vorgangsstruktur einordnen und in einer Akte ablegen, und die Bearbeitung des Falls kann nach Sachlage erfolgen.

Beim Assimilationsvorgang werden also von außen kommende Elemente für eine bestehende interne Struktur passend gemacht und in diese eingebaut. Das Schema, nach dem die interne Struktur aufgebaut ist, bleibt unverändert. Auf Softwaresysteme übertragen bedeutet dies, daß neue Datenobjekte erzeugt werden, welche aber bekannten abstrakten Datentypen angehören. Die abstrakten Datentypen selbst und die Methoden für deren Verarbeitung bleiben unberührt.

3.1.2 Repräsentation von Wissen in Form von Frames

Eine Erklärung der Abläufe bei der Assimilation von Wissen versucht Marvin Minsky in *A Framework for Representing Knowledge* [Minsky 75]. Ausgehend von klassischen und modernen Anschauungsweisen der Psychologie, der Linguistik und der Künstlichen Intelligenz entwickelt Minsky eine Theorie darüber, wie Wahrnehmungen der äußeren Welt mit den internen Gedächtnisstrukturen in Kontakt gebracht werden.

Der Mensch nimmt seine Umgebung wahr als eine Folge von *Situationen*: Eine Situation ist bestimmt durch den Zustand der äußeren Welt, der unter dem Blickwinkel des Betrachters erkennbar ist. Eine neue Situation tritt ein, wenn sich ein beobachteter Zustand ändert oder wenn sich der Blickwinkel des Beobachters verschiebt. Nach Minskys Vorstellung werden Situationen in der Form von sogenannten *Frames*[11] wahrgenommen und verstanden.

Frames sind Datenstrukturen, mit denen *stereotypische* Situationen repräsentiert werden, beispielsweise die perspektivische Erscheinung eines Zimmers für einen Betrachter oder der Besuch eines Kindergeburtstags. Frames können Beschreibungen derartiger Sachverhalte speichern; darüber hinaus enthalten sie Informationen, wie solche Beschreibungen zu interpretieren sind.

Minsky beschreibt Frames als Netze aus semantischen Symbolen und Relationen. Auf oberster Ebene ist ein solches Netz fest vorgegeben und bezeichnet Sachverhalte, die in einer entsprechenden Situation stets zutreffen. An den Enden des Netzes befinden sich Leerstellen, sogenannte *Slots*[12], die noch mit Daten gefüllt werden können. Diese Slots können mit Bedingungen versehen sein, die beim Füllen der Slots eingehalten werden sollen, oder sie können mit sogenannten *Defaults*[13] vorbesetzt sein; dies sind Voreinstellungen, welche gelten, bis der Slot mit einem abweichenden Wert gefüllt wird.

Das Wahrnehmen und Verstehen von Situationen beruht auf folgenden Abläufen: Wenn eine neue *konkrete* Situation eintritt, muß ein Frame gesucht werden, der eine *stereotypische* Situation beschreibt, die der gegebenen Situation möglichst ähnlich ist. Dann muß man versuchen, die Slots des Frames mit

---

[11] frame (englisch) = Rahmen

[12] slot (englisch) = Schlitz

[13] default (englisch) = Nichterscheinen

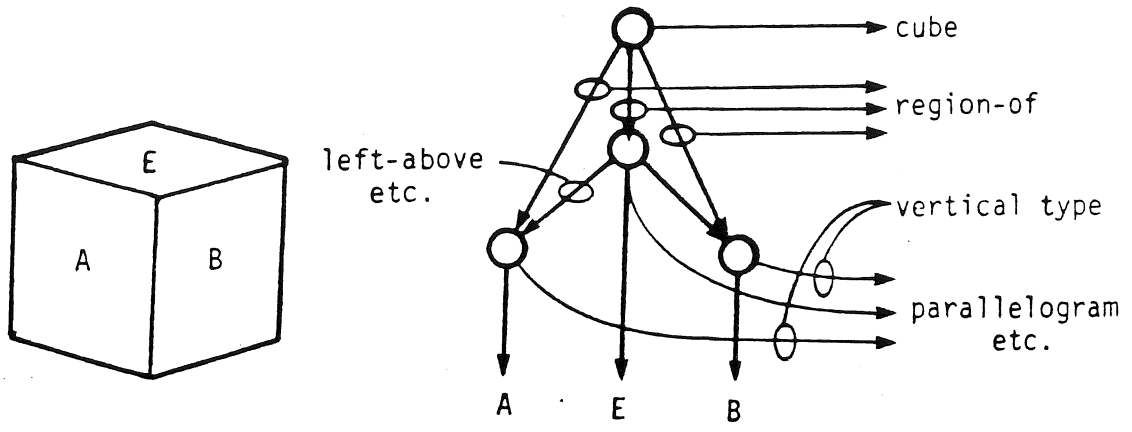


Abbildung 3-1: Die perspektivische Erscheinung eines Würfels und deren Darstellung in Form einer Framestruktur[14]

den aktuell gültigen Informationen zu füllen. Häufig sind dies andere Frames, deren Slots ebenfalls gefüllt werden müssen. Beim Füllen der Slots müssen die bestehenden Bedingungen überprüft werden; falls die Bedingungen unerfüllbar sind, muß der Vorgang mit einem anderen Frame wiederholt werden.

Am Beispiel der im letzten Abschnitt angeführten Schadensmeldung bei einer Versicherung läßt sich der Vorgang der Frameauswahl und des Füllens von Slots zeigen. Bei einer Schadensmeldung muß üblicherweise ein Formular ausgefüllt werden. Dieses Formular spielt die Rolle eines Frames. Welches Formular gebraucht wird, läßt sich meist anhand einiger weniger entscheidender Schlüsselinformationen feststellen. Beispielsweise stellen die Stichwörter *Kraftfahrzeug* und *Schadensmeldung* Angaben dar, die zur Bestimmung eines passenden Formulars ausreichen. Die auszufüllenden Eintragungsfelder des Formulars entsprechen den Slots eines Frames. Auf manchen Feldern liegen Bedingungen (beispielweise muß in einem Feld ein existierender Fahrzeugtyp eingetragen werden). Für manche Felder gelten Defaults (z.B. die Kontonummer für Erstattungen ist diejenige, die bereits früher bei der Erteilung der Abbuchungserlaubnis angegeben wurde). Eventuell müssen weitere Informationen auf Zusatzblättern gemacht werden; das heißt, ein zusätzliches Frame ist erforderlich, um den Sachverhalt ausreichend zu beschreiben.

[14]Abbildung entnommen aus [Minsky 75]

Das Auswählen und Füllen von Frames wird als *Instantiierung* bezeichnet, die gefüllten Frame nennt man auch Instanzen der ungefüllten Frames. Mit den im vorigen Abschnitt eingeführten Bezeichnungen läßt sich ein wesentlicher Unterschied zwischen ungefüllten und instantiierten Frames feststellen. Die instantiierten Frames beschreiben konkrete Sachverhalte, sie dienen also zur Repräsentation von Sachwissen. Die ungefüllten Frames enthalten Wissen über die Struktur und die stereotypischen Eigenschaften einer ganzen Klasse von Situationen, sie werden also verwendet, um konzeptuelles Wissen darzustellen.

Erinnerung, d.h. der Abruf von Informationen erfolgt auf ähnliche Weise wie das Verstehen von Informationen: Es muß ein Frame gefunden werden (jetzt allerdings ein gefülltes), das auf die Fragestellung paßt. Auch hier dienen Schlüsselinformationen zur Auffindung eines passenden Frames. Verwandte Frames können durch ein sogenanntes *Ähnlichkeitsnetz*[15] miteinander verbunden sein. Wenn das erste gefundene Frame die Suchbeschreibung nicht erfüllt, kann die Suche bei einem benachbarten Frame im Ähnlichkeitsnetz fortgesetzt werden.

3.1.3 Akkommodation von konzeptuellem Wissen

Die stereotypischen Situationen, die ein Mensch erwartet, sind in seinem konzeptuellen Wissen begründet. Wenn neue, bislang unbekannte Situationen auf den Menschen zukommen, reichen die geläufigen Assimilationsmechanismen nicht mehr aus, um die Erscheinungen der Umwelt zu erfassen. Der Mensch muß neue Konzepte erlernen, um diesen Situationen gerecht zu werden. Der Vorrat der abrufbaren Stereotypen muß erweitert werden. Im Gegensatz zur Assimilation, bei der das Individuum Wahrnehmungen seiner Umgebung verarbeitet, indem es sie den eigenen gedanklichen Strukturen anpaßt, erfordert der Erwerb neuer Konzepte die Anpassung der eigenen gedanklichen Strukturen an die Anforderungen von außen.

Beim Erwerb neuer Konzepte handelt es sich um den Vorgang der Akkommodation. Akkommodation wird von Piaget als ein zur Assimilation spiegelbildlicher Vorgang angesehen, nämlich als Wirkung der Umwelt auf den Organismus. Das bedeutet, daß ein Individuum die Folgen der es umgebenden Reize niemals bloß erduldet. Vielmehr paßt sich das Individuum an, indem es seinen Assimilationsfähigkeiten verändert. Akkommodation ist erforderlich, wenn die Assimi-

---

[15] Minsky verwendet den Begriff "similarity-network"

lationsmechanismen nicht mehr ausreichen, um die Erscheinungen der Umwelt zu erfassen. Es werden die Schemata des konzeptuellen Wissens verändert und bereichert, damit der Umgang mit neuen Formen von Sachwissen möglich wird. Die menschlichen Fähigkeiten zur Akkommodation kommen zum Ausdruck, wenn es darum geht, neue Aufgabenfelder zu beherrschen.

Beim Akkommodationsvorgang wird also der Aufbau der internen Strukturen verändert, damit die Außenwelt besser abgebildet werden kann. Akkomodation ist eine weit schwierigere Aufgabe als die Assimilation; es müssen neue Schemata aufgebaut werden und bestehende Strukturen umgeformt werden, damit sie den neuen Schemata entsprechen. Daher gehen Revisionen des konzeptuellen Wissens meist sehr langsam vonstatten.

In Softwaresystemen bedeutet Akkommodation, daß sich das Verhalten des Programmes beim Lösen von Problemen qualitativ verändert. Auf die Implementierung bezogen heißt dies, daß neue Datentypen und neue Verarbeitungsmethoden geschaffen werden müssen. Dazu sind herkömmliche Softwaresysteme selbständig nicht in der Lage. Die Akkommodationsaufgaben müssen von Systemanalytikern und Programmierern vorgenommen werden.

Am Beispiel der Versicherungsgesellschaft läßt sich die Akkommodation konzeptuellen Wissens ebenfalls deutlich machen: Wenn eine Versicherung neue Dienste anbieten will, zum Beispiel eine Haftpflichtversicherung für Privatflugzeuge, so reichen die bisher vorhandenen Schemata nicht aus, um die neu auftretenden Geschäftsvorfälle zu erfassen und zu bearbeiten. Es wird neues konzeptuelles Wissen benötigt; im konkreten Fall ist es erforderlich, neue Schemata für Haftpflichtfälle bei Privatflugzeugen zu entwickeln. Es müssen Verfahren gefunden werden, um die in Zukunft auftretenden Vorgänge zu assimilieren, mithin liegt ein Akkommodationsproblem vor.

Auch für den Akkommodationsvorgang wird konzeptuelles Wissen einer höheren Stufe benötigt: der Erwerb von konzeptuellem Wissen wird gesteuert durch *Metawissen.* Metawissen ist Wissen über den Umgang mit konzeptuellem Wissen, das heißt, im Metawissen liegen die Schemata, nach denen neue Schemata erworben werden können.

Am Beispiel der Versicherungsgesellschaft läßt sich dies wiederum verdeutlichen. Die Versicherung muß bei der Einführung der neuen Dienstleistung nicht völlig bei Null anfangen. Die in der Versicherung arbeitenden Versicherungsexperten

verfügen über Wissen, das sich auf alle Versicherungsformen anwenden läßt, also selbst auf solche, die es bisher nicht gab; dieses Wissen ist das Metawissen über Versicherungskonzepte im allgemeinen. In diesem Metawissen ist das Meta-Schema begründet, nach dem sich das Konzept der neuen Versicherungsform klassifizieren und einordnen läßt.

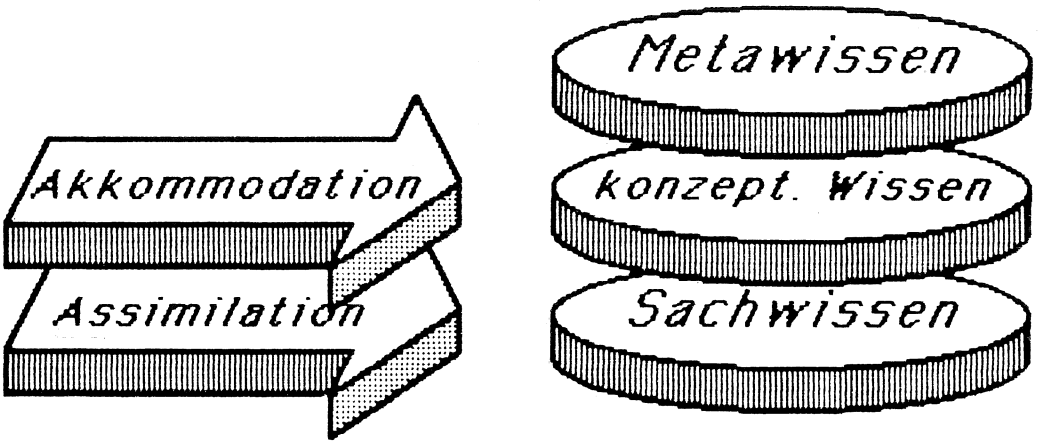


Abbildung 3-2: Akkomodation als Meta-Ausprägung der Assimilation

Hier zeigt sich einerseits eine Parallelität, andererseits eine Schichtung zwischen Assimilation und Akkommodation von Wissen. Assimilation hat Sachwissen zum Gegenstand und wird gesteuert von konzeptuellem Wissen. Akkommodation als Wissenserwerbsvorgang hat konzeptuelles Wissen zum Gegenstand und wird gesteuert von Metawissen. Das Metawissen spielt also für die Akkommodation dieselbe Rolle wie das konzeptuelle Wissen für die Assimilation. Im Metawissen liegt der Formalismus, mit dem konzeptuelles Wissen ausgedrückt werden kann, ähnlich wie konzeptuelles Wissen die Begriffe liefert, mit denen sich Sachwissen ausgedrücken läßt. Wenn wir über das notwendige Metawissen verfügen, wird konzeptuelles Wissen ähnlich handhabbar wie sonst das Sachwissen. Akkommodation ist in dieser Sichtweise nur noch die "Meta-Ausprägung" der Assimilation (siehe Abbildung 3-2).

Wie wird nun Metawissen erworben? Benötigt man zum Erwerb von Metawissen Wissen einer noch höheren Stufe, Meta-Meta-Wissen sozusagen, und zu dessen Erwerb wiederum Meta-Meta-Meta-Wissen? Gibt es eine Reihe von immer

abstrakteren Wissensqualitäten, welche niemals abbricht? Läßt sich in diesem Modell überhaupt der Vorgang des Erwerbs von konzeptuellem Wissen erklären? Oder wird bei den entscheidenden Fragen lediglich auf die nächsthöhere Meta-Ebene verwiesen? Wir werden darauf in den Abschnitten 3.4.3 und 5.4 noch eingehen.

Eines läßt sich bereits an dieser Stelle feststellen: Die besondere Eigenschaft des Metawissens besteht im Grunde nur darin, daß es eine Abstraktion konkreteren konzeptuellen Wissens darstellt. Dabei ist Metawissen auch nur eine Art konzeptuellen Wissens: wie dieses ist es Wissen über Wissen. Das heißt, alles über das konzeptuelle Wissen Gesagte läßt sich auch auf das Metawissen anwenden. Erwerb und Nutzung von Metawissen im besonderen brauchen also nicht anders vonstatten zu gehen als Erwerb und Nutzung von konzeptuellem Wissen im allgemeinen.

3.1.4 Kenntnisse und Fertigkeiten

Wissen tritt auf zwei unterschiedliche Weisen in Erscheinung: zum einen als *Kenntnis* von Informationen irgendwelcher Art, zum andern als *Fertigkeit*, d.h. als die Fähigkeit, in einer Situation richtig zu handeln.[16]

- *Kenntnisse* sind eine Sache des Kennens. Kenntnisse stellen diejenige Form von Wissen dar, mit welcher *WAS*-Fragen beantwortet werden. Dies ist Wissen um die Eigenschaften und Merkmale konkreter und abstrakter Dinge und Erscheinungen unserer Umwelt. Dieses Wissen ist beschreibender Art. Deshalb soll diese Erscheinungsform des Wissens im folgenden als *deskriptiver Aspekt* des Wissens bezeichnet werden.
- *Fertigkeiten* sind eine Sache des Könnens. Fertigkeiten beruhen auf dem Wissen, *WIE* man bestimmte Aufgaben löst, indem man problemangemessen handelt. Da sich Fertigkeiten in der Fähigkeit zeigen, bestimmte Tätigkeiten zu verrichten, soll diese Erscheinungsform des Wissens im folgenden als *prozeduraler Aspekt* des Wissens bezeichnet werden.

Deutlich wird diese Doppelnatur des Wissens, wenn man untersucht, welche Rolle Wissen beim Lösen von Problemen spielt. An einem Beispiel, der Zubereitung eine Mahlzeit, soll dies im folgenden gezeigt werden. Das Lösen dieses Problems läßt sich grob in zwei Tätigkeiten zerlegen: das Erkennen einer Anfangssituation und das Handeln zum Erreichen eines Zielzustands.

---

[16] Im englischen Sprachgebrauch wird für Kenntnis das Wort knowledge, für Fertigkeit das Wort skill verwandt

Das Erkennen einer Problemsituation ist der bekannte Vorgang der Assimilation von Faktenwissen, also ein Wissenserwerbsvorgang. Ziel dieses Wissenserwerbsvorgangs ist es, Kenntnisse über die Ausgangssituation zu erhalten, das heißt, zu wissen, *WAS* die Eigenschaften und Merkmale der gegebenen Situation sind. Die erforderlichen Kenntnisse sind in unserem Beispiel: *Für wieviel Personen soll gekocht werden? Wann muß die Mahlzeit fertig sein? Welches Gericht soll gekocht werden? Welche Zutaten werden benötigt? Welche Zutaten sind vorhanden?*. Das beim Erkennen einer Problemsituation erworbene Wissen ist also Faktenwissen mit deskriptivem Charakter.

Selbstverständlich wird mehr als nur dieses Faktenwissen benötigt, um das Problem zu lösen. Es wird vor allem konzeptuelles Wissen benötigt, dieses ist aber in der Regel zu einem früheren Zeitpunkt erworben worden, in unserem Beispiel vielleicht in einem Kochkurs. Das konzeptuelle Wissen zeigt sich in der Kenntnis von Gerichten und Kochrezepten und in der Fähigkeit, nach diesen Kochrezepten ein bestimmtes Gericht zuzubereiten. Konzeptuelles Wissen kann also sowohl deskriptiv *("Was sind die Eigenschaften der Gerichte und Zutaten?")*, als auch prozedural *("Wie wird Feldsalat verlesen?")* in Erscheinung treten.

Wenn allerdings konzeptuelles Wissen angewandt werden soll, ist der prozedurale Aspekt des Wissens entscheidend. Das Wissen soll dann in Handlungen umgesetzt werden, es sind also Fertigkeiten gefragt. Im Beispiel des Kochens ist nicht entscheidend zu wissen, was die exakte Definition von Fadennudeln ist, sondern wie man sie kocht.

Der Vorteil prozeduralen Wissens besteht also in seiner unmittelbaren Anwendbarkeit. Die Bedeutung deskriptiven Wissens tritt deshalb in vielen praktischen Anwendungsfällen zurück. Im Extremfall sind die deskriptiven Anteile des Wissens nicht mehr rekonstruierbar. In diesem Fall reden wir auch von einer "compilierten Darstellung" des Wissens: Dabei besteht das gesamte Wissen aus vorgefertigten Anweisungsfolgen, die im gegebenen Fall nur noch befolgt werden müssen. Derart kodiertes Wissen läßt sich auf einfache Weise praktisch umsetzen; es ist daher besonders geeignet für eine maschinelle Verarbeitung. Compilierte Wissensdarstellung ermöglicht schnelle und effiziente Problemlösungen, sie ist für Standardaufgaben gut geeignet, sie wird aber in den von der Regel abweichenden Fällen versagen, da es nicht möglich ist, alle Situationen vorherzusehen.

Häufig kann prozedurales Wissen aus deskriptivem Wissen erschlossen werden: Wenn wir voraussetzen, daß bereits Wissen über den Kochen von Teigwaren vorhanden ist, kann aus dem deskriptiven Wissen *"Fadennudeln sind Teigwaren mit vergleichsweise kleinem Querschnitt"* der Zubereitungsvorgang für Fadennudeln abgeleitet werden. Dieser Vorgang unterscheidet sich von der Zubereitung von gewöhnlichen Nudeln durch die kürzere Kochzeit.

Wenn Wissen nur in prozeduraler Form vorliegt, ist es nicht mehr möglich, Handlungen zu rechtfertigen und zu erklären. Es werden nur auswendig gelernte Handlungsanweisungen befolgt, es ist aber kein Wissen darüber vorhanden, warum die Handlungsanweisungen so und nicht anders lauten. Auch beim Erwerb von konzeptuellem Wissen ist eine nurprozedurale Betrachtungsweise nachteilig: Wissensvermittlung, die sich auf die Weitergabe von bloßen Rezepten beschränkt, kann zu unbefriedigenden Ergebnissen führen, wenn dabei kein tieferes Verständnis der zugrundeliegenden Sachverhalte entsteht. Oftmals ist es deshalb einfacher, präziser und wirkungsvoller, nicht Lösungsverfahren zu beschreiben, sondern Eigenschaften und Gesetzmäßigkeiten des Problemgebiets. Es wird dann dem Lernenden bewußt, *WAS* die Eigenheiten des Problemraums sind; denn es liegt Wissen darüber in deskriptiver Form vor. Bei der Lösung eines konkreten Problems besteht aber dann die Notwendigkeit, das deskriptive Wissen zu interpretieren und in Methoden, Pläne und Aktionen umzusetzen. Durch häufige Anwendung des Wissens, durch Übung, können sich dann aus den Kenntnissen des Problemraums nach und nach Fertigkeiten, also compiliertes Wissen bilden.

Der deskriptive und der prozedurale Aspekt des Wissens sind also beide entscheidend, wenn wirklich von Wissen und Wissensverarbeitung die Rede sein soll. Ist nur der deskriptive Anteil des Wissens vorhanden, so ist nicht klar, wie das Wissen anzuwenden ist. Ist nur der prozedurale Anteil des Wissens vorhanden, so fehlt das wirkliche Verständnis des Problems.

3.2 Klassische Techniken der Wissensdarstellung

Welche klassischen Methoden der Wissensdarstellung gibt es? Eine musterhafte Art der Wissensdarstellung liegt der Erinnerungsfähigkeit des Menschen zugrunde. Es ist uns Menschen aber von Natur aus nicht bewußt, auf welchen Prinzipien diese beruht. Auch die biochemische und die neurologische Forschung konnte die natürlichen Wissensrepräsentationstechniken des Menschen bisher nicht enträtseln. Es scheint, daß ebensowenig wie durch das Zerlegen eines Computers in seine Einzelteile erschlossen werden kann, wie dessen Programme implementiert sind, sich durch Untersuchungen des menschlichen Gehirns die Frage lösen läßt, wie das Wissen in den Köpfen von Menschen dargestellt ist.

Ein praktikables Vorgehen besteht jedoch darin, die Situationen zu untersuchen, in denen sich das menschliche Wissen veräußerlicht. Derartige Untersuchungen lassen Erkenntnisse auf dem Gebiet der Wissensrepräsentation und Einsichten in Techniken der Wissensübertragung erwarten. Es lassen sich so Vorbilder für die Gestaltung wissensverarbeitender Computersysteme gewinnen.

Wissen veräußerlicht sich bereits, wenn Menschen Wissen praktisch anwenden. Am deutlichsten wird aber menschliches Wissen veräußerlicht, wenn es an andere Menschen weitergegeben wird. Die Übertragung von Wissen an andere Menschen ist ein Kommunikationsvorgang. Zur Kommunikation wird eine Sprache oder wenigstens ein Informationscode benötigt. Die Codes und Sprachen, die der Mensch zur Kommunikation verwendet, stellen also die klassischen Hilfsmittel zur Darstellung von Wissen in einer von seinem ursprünglichen Träger losgelösten Form dar.

In den folgenden Abschnitten werden die die Techniken zur Übertragung und externen Darstellung von Wissen historisch betrachtet. Dabei läßt sich ein Übergang von eher unbewußten und informellen zu immer mehr expliziten und formalisierten Techniken feststellen. Diese Entwicklung hat letztlich den Einsatz von Computern möglich gemacht. Mit der Benutzung von Computern kehrt sich diese Entwicklung jedoch teilweise wieder um, was die Formalität der Interaktion angeht. Bei der Benutzung moderner Softwaresysteme sind immer weniger formale Protokolle einzuhalten.

3.2.1 Biologische Vererbung

Die *biologische Vererbung* von Verhaltensweisen und Merkmalen der äußeren Gestalt ist die älteste und am wenigsten bewußte Art der Übertragung von Wissen. Der genetische Code dient hierbei zur Repräsentation des Wissens. Die Nutzanwendung dieses Wissens liegt in der Fortpflanzung der Art. Dem Lebewesen (Pflanze, Tier oder Mensch) ist dieses Wissen jedoch nicht bewußt; es bestimmt die Zugehörigkeit zu einer Art und äußert sich in Reflexen und unbewußten Instinkten. Eine Darstellung der biologischen Reproduktionsmechanismen als Vorgänge genetischer Informationsverarbeitung findet sich in [Häfner 86].

3.2.2 Nachahmung

Das Prinzip von *Vorbild und Nachahmung* ist die vorherrschende Art, auf der die Wissensvermittlung bei kleinen Kindern beruht. Nachahmung ist nicht auf Verbalisierung angewiesen; die Kommunikation beruht auf der Ausübung und Wahrnehmung von Körpergesten. Das erworbene Wissen besteht in der Kenntnis, *wie* eine Tätigkeit ausgeübt wird, die logische Rechtfertigung, ja selbst der Zweck der erworbenen Handlungsweise bleibt hierbei noch unklar. Dennoch ist das Prinzip von Vorbild und Nachahmung sehr wirkungsvoll. Nachahmung ist ein Beispiel dafür, wie der Mensch Wissen im Handeln erwirbt.

F. Nake vertritt in [Nake 85, Seite 66ff] die Ansicht, daß alles Wissen letztlich durch Handeln erworben sei: *"Wissen ohne das Handeln des Menschen, 'unmittelbares Wissen', gibt es nicht... Einwenden wird man - und zwar gerade dann, wenn es um die Vermittlung von Tatsachen und Regeln am Computer geht -, daß Wissen doch aber auch durch Kommunikation entstehen kann, daß es mitgeteilt und nicht nur handelnd erworben werden könne... Was jedoch dabei vermittelt wird, ist bereits Vermitteltes, muß Wissen schon vorher geworden sein..."* Für Dröge [Dröge 72], auf den sich Nake beruft, ist menschliches Handeln sogar eine definierende Eigenschaft des Wissensbegriffs: *"Wissen ist die im Handeln vermittelte und dieses vermittelnde Existenzform individueller und gesellschaftlicher Erfahrung."*

Das Prinzip von Vorbild und Nachahmung zur Wissensübertragung hat auch Eingang in das Schnittstellendesign für Computersysteme gefunden:

- *Query-by-Example* [Zloof 75] ist eine Dialogsprache für relationale Datenbanken, die darauf beruht, daß der Benutzer seine Anfrage durch Eingabe einer möglichen Lösung angibt. Query-by-Example erschließt die beabsichtigte Anfrage des Benutzers, indem es die vorgegebene Antwort abstrahiert, und gibt die Ergebnisse in Tabellenform aus.

- *Tinker* [Liebermann, Hewitt 80] ist eine Programmierumgebung für Lisp, in der das Programmieren durch Vorgabe von Beispielen erfolgt. Der Benutzer spezifiziert Programmschritte für eine konkrete Situation. Tinker erzeugt daraus eine Lisp-Prozedur, die den allgemeinen Fall abdeckt. Durch Vorgabe zusätzlicher Beispiele kann das Verhalten der Prozedur getestet und verbessert werden.

3.2.3 Sprache

Die *natürliche Sprache* ist für uns Menschen das wichtigste Medium zur Wissensvermittlung. Die natürliche Sprache kann ebenso wie Gestik und Nachahmung dazu dienen, Handlungsweisen zu vermitteln; also auch mit Sprache kann das *WIE* beschrieben werden. Gleichzeitig ist es möglich, mit Hilfe der natürlichen Sprache Gegenstände und Begriffe zu benennen und zu beschreiben; d.h. es kann ausgedrückt werden, *WAS* für Sachverhalte einer Handlung zugrundeliegen. Dies gilt für gesprochene wie für geschriebene Sprache gleichermaßen.

Die Verwendung von Sprache bewirkt eine Trennung zwischen den Sachverhalten und Objekten der wirklichen Welt und ihren sprachlichen Beschreibungen, eine Trennung zwischen den Dingen und ihren Namen, eine Trennung zwischen Original und Abbild. Die Beschreibung eines Objekts der realen Welt bezeichnet dieses Objekt, ist aber nicht das Objekt selbst.

Die Zuordnung zwischen sprachlichen Beschreibungen und Zuständen der Welt ist durch die Semantik der Sprache bestimmt. Die Festlegung der Semantik ist ein besonderer Vorgang der Kommunikation, der als *Metakommunikation*[17] bezeichnet wird. Hierfür wird eine weitere Art von Sprache, eine *Metasprache*, erforderlich, mit der die Phänomene der Welt und die Ausdrucksmittel der verwendeten Sprache bezeichnet und zueinander in Bezug gesetzt werden können. Als Metasprache kann wiederum die natürliche Sprache dienen. Dies ist in der Umgangssprache so üblich. Auch im vorliegenden Dokument geschieht dies so, wenn die Eigenschaften von Sprachen zur Vermittlung und Repräsentation von Wissen erörtert werden. Dies hat den Vorteil, daß an den Leser keine zusätzli-

---

[17] siehe hierzu z.B. [Oberquelle et al. 83]

chen Voraussetzungen gestellt werden. Der Nachteil besteht darin, daß es dann mitunter schwierig wird, präzise Formulierungen zu finden. Wir müssen daher manchmal scheinbar umständliche Ausdrucksweisen in Kauf nehmen, wenn wir mit ebenso verständlichen wie zweifelsfreien Worten zwischen Objekten und ihren Namen, zwischen Eigenschaften von Objekten und ihren Beschreibungen unterscheiden wollen.

Für den Spezialfall *schriftlich* fixierter Sprache gilt alles bisher gesagte ebenso. Die geschriebene Sprache ist aber formaler und durch Satzzeichen, Absätze, Überschriften usw. stärker strukturiert als die gesprochene Sprache.

Der Zugriff auf schriftliche Informationen ist grundsätzlich wahlfrei: Schriftlich niedergelegtes Wissen läßt sich nach Ordnungskriterien organisieren. Hierarchische und lexikalische Ordnungsprinzipien sind möglich. Durch Querverweise läßt sich die sequentielle Ordnung durchbrechen. Große Mengen von Informationen können in Lexika, Registraturen und Bibliotheken verwaltet werden.

Bei schriftlicher Kommunikation ist die Einheit der Zeit zwischen Wissensvermittler und Wissenserwerber durchbrochen, der erstere kennt den letzteren meist auch gar nicht. Es können weniger Annahmen über einen Leser getroffen werden als über einen Hörer; Rückfragen des Lesers sind nicht ohne weiteres möglich. Daher wird Wissen in schriftlichen Darstellungen meist viel expliziter vermittelt als mit gesprochener Sprache.

Für spezielle Erfordernisse wurden vom Menschen *künstliche, formale Sprachen* geschaffen. Beispiele hierfür sind:

- der algebraische Kalkül der Mathematik, wie er von Vieta Ende des 16. Jahrhunderts eingeführt wurde [ABC Mathematik 78];
- die logische Formelschrift [Frege 79];
- chemische Strukturformeln;
- Gleichungen zwischen physikalischen Größen;
- algorithmische Sprachen, wie z.B. Algol [Backus et al. 63].

Für formale Sprachen gibt es eindeutige Definitionen, die festlegen, wie die Ausdrücke der Sprache zu interpretieren sind. Die Bedeutung einer Formulierung ist unabhängig vom Sprecher, vom Hörer und von der Situation. Diese Tatsache begünstigt den Einsatz von formalen Sprachen als Computersprachen.

Im Kontrast zur Informationsübermittlung mit Hilfe einer formalen Sprache steht die Kommunikation zwischen Menschen. In der zwischenmenschlichen Kommunikation wird abhängig von den Wissensständen der Gesprächspartner und dem Zustand der gemeinsamen Umwelt nur ein Teil einer Gesamtinformation expressis verbis ausgetauscht. Der Rest wird nur indirekt vermittelt, indem er von den Kommunikationspartnern aus anderen Quellen erschlossen wird. Nach dem Kommunikationsschema von Kupka, Maaß und Oberquelle (Abbildung 3-3) ist die zwischenmenschliche Kommunikation als Aktivität zu sehen, die von einem äußeren Faktor und fünf inneren Faktoren abhängig ist: dem Einfluß der Umwelt sowie den Konventionen, dem Selbstbild, den Intentionen, dem Partnerbild und dem Allgemeinwissen, über das der jeweilige Kommunikationspartner verfügt. Alle sechs Faktoren sind aber nicht statisch, sondern werden wiederum durch den Kommunikationsvorgang beeinflußt ÄOberquelle et al. 83].

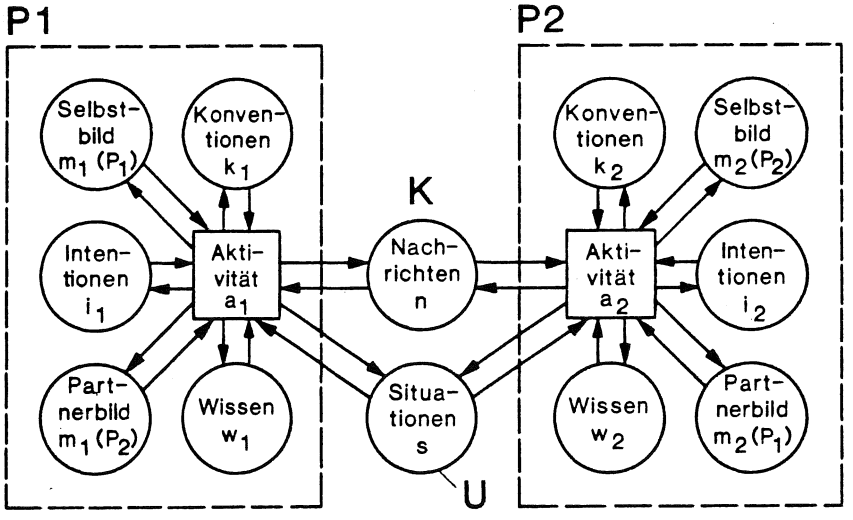


Abbildung 3-3: Kommunikationsschema von Kupka, Maaß und Oberquelle

Anhand dieses Modells wird deutlich, daß stark formalisierte, nicht kontextbezogene Kommunikationsformen für den menschlichen Gebrauch in breitem Umfang nicht angemessen sind. Die große Herausforderung bei der Gestaltung benutzergerechter Computersysteme besteht deshalb darin, das Kommunikationsverhalten zwischen Menschen auf die Interaktion mit dem Computer zu übertragen. Fischer [Fischer 83] schlägt zu diesem Zweck eine wissensbasierte Ar-

chitektur für die *Mensch-Computer-Kommunikation* vor. Er sieht die von Oberquelle et. al. beschriebenen inneren Faktoren der Kommunikation als Inhalt einer sogenannten *Wissensbasis* an. Eine Wissensbasis kann nach Fischer folgende Arten von Wissen enthalten:

- Modelle der Kommunikationspartner
- Wissen über Problemlösen
- Wissen über spezielle Problembereiche
- Wissen über Kommunikationsprozesse

Menschliche Kommunikationspartner verfügen über eine derartige Wissensbasis. Erforderlich ist es, auch im Computersystem eine derartige Wissensbasis aufzubauen. Wenn nun in beiden Wissensbasen, der des Menschen und der des Computers, gemeinsames Wissen vorhanden ist, wirkt dies so, als wäre neben dem expliziten Kommunikationskanal (gegeben durch das Ein/Ausgabegerät des Computersystems und das sensomotorische System des Menschen) noch ein weiterer impliziter Kommunikationskanal vorhanden, und es ist nicht mehr erforderlich, alle Information explizit mit dem Computer auszutauschen (Abbildung 3-4). In Abschnitt 3.3.3 werden wir noch auf die konstruktiven Gesichtspunkte beim Bau wissensbasierter Systeme zu sprechen kommen.

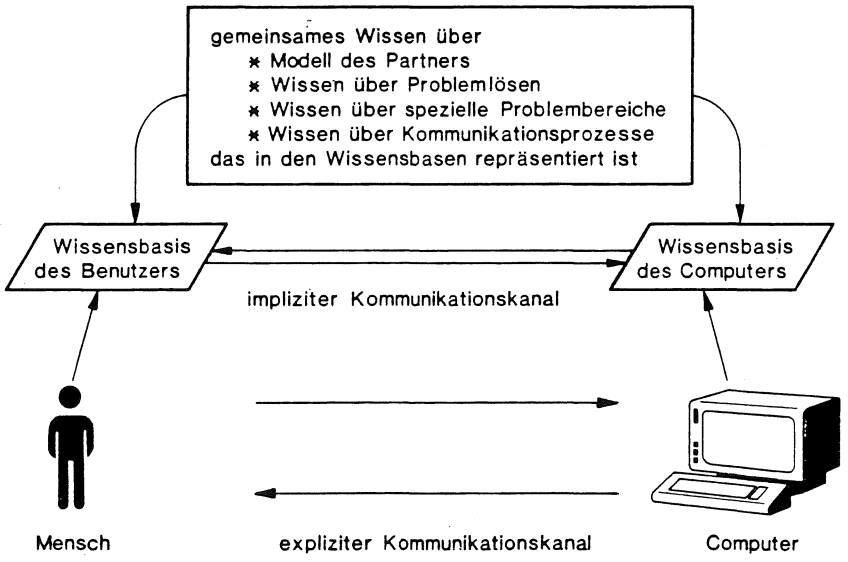


Abbildung 3-4: Architektur eines wissensbasierten Systems für die Mensch-Computer-Kommunikation nach Fischer

3.2.4 Bild

Noch älter als die Verwendung von Schrift ist die Technik der Aufzeichnung von Informationen mit Hilfe von bildlichen Darstellungen. Bilder eignen sich gut, um konkrete Situationen und Gegenstände abzubilden. Die Bedeutung von Abbildungen ist meist unmittelbar "einsichtig". Die Transformationsdistanz zur Realität ist geringer als etwa bei der Beschreibung von Situationen mit sprachlichen Mitteln. Für die Darstellung von Abläufen eignen sich Bilderfolgen oder Filme. Schwieriger ist die Aufgabe, abstrakte Begriffe in bildlicher Form darzustellen: Geeignete Darstellungsweisen sind dann die symbolische Darstellung von Begriffen (beispielsweise ein Herz für den Begriff Liebe) oder die exemplarische Darstellung (etwa die Abbildung des Sandstrands für den Begriff Ferien). Bilder sind unabhängig von der natürlichen Sprache. Sie sind aber dennoch nicht kulturunabhängig, vor allem wenn sie Abstrakta bezeichnen sollen.

Die Elemente eines Bildes verteilen sich auf einen zweidimensionalen Raum, ihre Beziehung untereinander ist nicht durch eine lineare Ordnung eingeschränkt. Daher sind Bilder insbesondere geeignet, um Konfigurationen von Objekten darzustellen, die in vielfältigen Relationen zueinander stehen. Es gibt bildliche Darstellungsformen, für die eine definierte Lesart existiert, Beispiele hierfür sind:

- Funktionsgraphen in der Mathematik,
- Konstruktionszeichnungen in der Architektur und im Maschinenbau,
- Schaltpläne in der Elektronik.

Derartige Graphiken können in einer ähnlichen Weise formal sein wie geschriebene formale Sprachen. Es gibt eine Syntax und eine Semantik, die durch Normen und Konventionen gegeben sind. Auch Daten- und Programmstrukturen lassen sich durch solche Graphiken darstellen, etwa durch Nassi-Shneiderman-Diagramme [Nassi, Shneiderman 73], durch die HIPO-Technik [IBM 74], durch ein STRUPPI-Diagramm [Bauer D. 84] oder durch eine der vielfältigen Darstellungsarten von Funktionen und Daten, die in [Böcker, Fischer, Nieper 86] beschrieben sind. Der in einem späteren Kapitel beschriebene Wissenseditor ZOO verwendet ebenfalls eine derartige graphische Darstellungsform.

Graphische Darstellungen haben aber auch einen nicht-formalen Charakter: Die zugrundeliegende Syntax wird als nicht so fremdartig empfunden, wie die Syn-

tax einer formalen Sprache. Oft ist die Syntax bereits durch die Gesetze der Geometrie gegeben; beispielsweise ist es unmöglich, ein kleines Objekt darzustellen, daß ein größeres enthält. Andere syntaktische Regeln treten visuell als ästhetische Kriterien in Erscheinung, wie etwa die Geschlossenheit oder Parallelität von Linien.

Graphische Darstellungen werden mit zunehmendem Erfolg für die Gestaltung von Benutzerschnittstellen von Computersystemen eingesetzt. Diese Benutzerschnittstellen stützen sich meist auf eine sogenannte *Metapher*, das heißt, es wird versucht, den Problemraum in der Gestalt einer gegenständlichen Welt auf dem Bildschirm zu zeigen. Beispiele für derartige Metaphern sind:

- Die Schreibtischmetapher bei Fenstersystemen [Fabian 86]. Unterschiedliche Anwendungen leben in rechteckigen Bildschirmbereichen, die sich wie papierene Dokumente auf einem Schreibtisch gegenseitig überdecken können.
- Die Bürometapher bei Büroinformationssystemen, z.B. im Xerox Star [Smith et al. 82]. Textdateien, Dateiverzeichnisse und Softwarewerkzeuge werden als Papierblätter, Aktenordner und Bürogeräte visualisiert.

3.3 Maschinelle Wissenstechniken

Ist es überhaupt möglich, Wissen in Form von Computersoftware darzustellen? Sicherlich ja. In jedem Programm steckt Wissen seines Programmierers, das bei der Benutzung des Programmes zur Anwendung kommt. Dieses Wissen betrifft zum einen das Rechnersystem und dessen Möglichkeiten, zum andern das Anwendungsgebiet.

Entscheidend für die Verwertbarkeit dieses Wissens ist allerdings, in welcher Gestalt es in dem Computerprogramm dargestellt ist. Häufig lassen sich die natürlichen Bestandteile dieses Wissens nicht mehr eindeutig an einer bestimmten Stelle des Programmes lokalisieren; denn der Programmcode ist eher nach technischen Gesichtspunkten strukturiert als nach den verschiedenen Arten und Zusammengehörigkeitsbeziehungen des zugrundeliegenden Wissens. Außerdem ist ein großer Teil des Wissens des Programmierers, beispielsweise über die Problemstellung und die Ziele der Softwarelösung, für den Programmablauf nicht wesentlich und ist infolgedessen im Programmcode nicht mehr notwendigerweise repräsentiert. In einem derartigen Fall sprechen wir von einer *impliziten* Wissensdarstellung.

Ein typisches Merkmal der impliziten Wissensdarstellung ist ihre einseitige Ausrichtung an einem Verwendungszweck, nämlich der Steuerung des Ablaufs eines Anwendungsprogramms. Es handelt sich hierbei zumeist um compiliertes Wissen (siehe Abschnitt 3.1.4), also Wissen, das in laufzeiteffizienter, prozeduraler Form dargestellt ist. Wenn es nur darum geht, eine Maschine zu programmieren, kann diese Darstellungsform durchaus zweckmäßig sein. Eine für den Menschen verständliche Wissensdarstellung muß aber auf vielseitige Weise interpretierbar sein. Für den Menschen wesentlich ist hierbei der deskriptive Aspekt des Wissens, der von Bedeutung ist, wenn das Programm dokumentiert, erklärt oder verändert werden soll. Deskriptiv formuliertes Wissen ist aber in einem herkömmlichen Computerprogramm nur implizit enthalten; ohne eine zusätzliche Dokumentation läßt es sich aus dem Code des Programms nicht mehr ablesen.

Im Gegensatz dazu steht die Idealvorstellung einer *expliziten* Darstellung des relevanten Wissens in einem Rechnersystem. Explizit repräsentiertes Wissen ist aus dem System herauslösbar, es existiert unabhängig von dem Verarbeitungsprogramm, das es aufbaut, benutzt und interpretiert. Unterschiedliche Arten von Wissen sind voneinander getrennt, verwandte Arten von Wissen sind an einem gemeinsamen Ort repräsentiert. Das Wissen einer bestimmten Art wiederum ist aufgeteilt in isolierbare Wissenseinheiten. Diese explizite Darstellung von Wissen rechtfertigt erst die Verwendung des Begriffs *Wissensrepräsentation,* weil sie sich an der Struktur des betreffenden Wissens orientiert und nicht an irgendwelchen anderen Gesichtspunkten.

Der Nutzen einer expliziten Wissensdarstellung besteht darin, daß derart repräsentiertes Wissen auf viele Weisen interpretierbar ist. Explizit vorhandenes Wissen läßt sich in eine menschenangemessene Darbietungsform ebenso übersetzen wie in einen maschineneffizienten Code.[18] Prozeduraler und deskriptiver Aspekt des Wissens sind gleichermaßen berücksichtigt.

3.3.1 Datenbanksysteme

Unter *Daten* versteht man zunächst einfache, symbolische Elemente wie Zahlenwerte oder Zeichenketten. Durch Kombinationen elementarer Daten erhält man darüberhinaus Daten mit komplexerer Struktur, wie etwa Datensätze, Datenbäu-

---

[18]Siehe hierzu auch "Knowledge-based Communication Processes in Software Engineering" [Fischer, Schneider 84a].

me oder Netze. Unter einer *Datenbank* versteht man eine Kollektion von Daten, die einem eingrenzbaren Zweck dienen. Das der Datenbank zugrundeliegende *Datenbanksystem* schließlich legt die Methodik fest, nach der Daten zur Verfügung gestellt werden [Date 75]. Die Daten einer Datenbank reflektieren das Faktenwissen, das von den angeschlossenen Anwenderprogrammen mit Hilfe einer *Datenmanipulationssprache* aufgebaut und genutzt wird.

Sogenannte *Datenmodelle* oder *konzeptuelle Modelle* definieren die Strukturen, nach denen eine Datenbasis aufgebaut ist. Beispielsweise in relationalen Datenbanksystemen [Codd 70] handelt es sich bei diesen Strukturen um Relationen im mathematischen Sinn, die extensional als Tabellen elementarer Daten definiert sind. Konzeptuelle Modelle werden vom Datenbankverwalter mit Hilfe einer *Datendefinitionssprache* beschrieben. Sie spiegeln das konzeptuelle Wissen wider, das einer Datenbank zugrundeliegt.

Datenbanksysteme sind sehr flexibel und können auch sehr komplexe Sachzusammenhänge repräsentieren. Schwierigkeiten entstehen jedoch, wenn es darum geht, aus Daten das zugrundeliegende Wissen zu rekonstruieren. Die Freiheit bei der Interpretation der Daten ist nämlich groß, da die Semantik der Daten in der Datenbank selbst nicht repräsentiert ist. Die konzeptuellen Modelle, die eventuell über die Bedeutung der Daten Aufschluß geben könnten, sind selbst keine Daten, sondern Ausdrücke einer von der Datenbank abgekoppelten Datendefinitionssprache. Sie können daher nicht einfach über die Datenmanipulationssprache erreicht werden.

Den an eine Datenbank angeschlossenen Anwenderprogrammen bleibt es somit selbst überlassen, die erhaltenen Daten zu interpretieren. Die Bedeutung der Daten ist für alle praktischen Belange implizit in den Prozeduren der Anwendungsprogramme festgelegt. Wenn auch versucht wird, Programmbausteine für Anwendungen in sogenannten Methodenbanken [Dittrich et al. 79] zu verwalten, bleibt dennoch das Problem bestehen, daß das Wissen um die Bedeutung der Daten nirgendwo explizit und für alle Anwendungen bindend dargestellt ist.

Ähnliche Konsequenzen ergeben sich auch für die Aufgaben des Wissenserwerbs: Die zu assimilierenden Sachinformationen müssen vom Anwendungssystem in eine Datenform überführt werden. Das Wissen darüber, durch welche Arten von Daten die erhaltenen Informationen repräsentiert werden können, liegt beim Anwendungsprogramm selbst und kann nicht aus dem konzeptuellen Modell der Datenbank abgeleitet werden. Der Erwerb konzeptuellen Wissens, der eine Ak-

kommodation des konzeptuellen Modells erfordert, kann schon gar nicht von einem Anwendungsprogramm aus erfolgen, sie bleibt einzig und allein dem Systemverwalter vorbehalten und erfordert Kenntnisse der Datendefinitionssprache.

3.3.2 Informationssysteme

Informationssysteme gewinnen Informationen aus Daten oder führen Informationen in eine Datendarstellung über. Dabei sind Informationen mehr als Daten. Eine Information ist der abstrakte Gehalt einer Auskunft oder einer Nachricht, die für einen Adressaten bestimmt ist und für diesen einen Aspekt der Nutzung besitzt. In [Bauer, Goos 71] wird der Begriff der Information definiert als das Ergebnis der Interpretation einer Nachricht. In diesem Sinne kann man Informationen verstehen als interpretierte Daten [Neuhold 77].

Die meisten bisherigen Informationssysteme haben die *Produktion* von Informationen zum Ziel. Managementinformationssysteme, Personalinformationssysteme, elektronische Kommunikationssysteme gewinnen Informationen aus Daten, erzeugen Informationen oder verbreiten Informationen. Vernachlässigt wurden bisher Systeme, die dem Menschen helfen, die Flut von Informationen wieder zu verarbeiten. Notwendig sind Werkzeuge und Systeme zur inhaltlichen *Aufarbeitung* von Informationen.

Die einzigen derartigen Systeme, die derzeit zur Verfügung stehen, sind Dokumentationssysteme oder Dokumentenablagesysteme. Diese Systeme verarbeiten die eingegebenen Dokumente jedoch nicht inhaltlich. Gespeichert werden die Textquellen sowie Schlüsselinformationen, nicht aber die Bedeutungen der Texte. Dies hat Auswirkungen auf die Art der Recherche: Informationen können nicht nach inhaltlichen Gesichtspunkten abgerufen werden, sondern nur nach formalen Kriterien. Es ist beispielsweise leicht möglich, alle Textdokumente abzurufen, die einen bestimmten Textstring enthalten (z.B. "Lisp"), man hat aber bereits Schwierigkeiten, wenn man ein Dokument sucht, das sich auf ein bestimmtes Thema (z.B. "Symbolmanipulation") bezieht, wenn der betreffende Begriff weder im Text enthalten ist, noch beim Vorgang der Ablage dem Dokument explizit zugeordnet worden ist. Ein erster Schritt zu einer befriedigenderen Lösung besteht beispielsweise darin, die Relationen zwischen Schlüsselwörtern in der Datenbasis zu repräsentieren (wie dies beispielsweise in [Arning 86] geschieht). Das Ziel, das auch mit dem im Kapitel 4 beschriebenen D&I Anwendungssystem verfolgt wird, ist aber die Repräsentation der Bedeutung der zu verarbeitenden Texte.

3.3.3 Wissensbasierte Systeme

> Knowledge = data + interpretation, where *data* can be anything and *interpretation* involves decoding and inferencing. (Freksa, Furbach und Dirlich)

Während *Information* als das *Ergebnis* der Interpretation von Daten gesehen werden kann, impliziert *Wissen* die *Fähigkeit* zu einem solchen Interpretationsvorgang. Der aus [Freksa et al. 84] zitierte Satz muß deshalb folgendermaßen übersetzt werden: *Wissen = Daten + die Fähigkeit, diese Daten zu interpretieren und Informationen daraus abzuleiten.* Wenn man Daten als ein Mittel betrachtet, um Fakten zu repräsentieren, und die Fähigkeit, mit diesen Daten umzugehen, dem konzeptuellen Bereich zuordnet, kommt auch hier wieder die in Kapitel 3.1 beschriebene Einteilung von Wissen in Faktenwissen und konzeptuelles Wissen zum Ausdruck.

Konzeptuelles Wissen ist in Datenbanken und herkömmlichen Informationssystemen ungenügend repräsentiert. Die möglichen Verwendungsarten und Interpretationen des in der Datenbank repräsentierten Faktenwissens sind durch die Gesamtheit der angeschlossenen Informations- und Anwendungssysteme gegeben. Diese Verwendungsarten können aber von diesen Programmsystemen nicht aus der Datenbank abgeleitet werden. In den Anwendungsprogrammen ist also implizit und vermischt mit der Steuerung des Programmablaufs auch ein Teil des konzeptuellen Wissens repräsentiert.

Der wissensbasierte Ansatz hingegen hat die Trennung von Wissen und Ablaufsteuerung zum Ziel. Die Ablaufsteuerung eines wissensbasierten Systems übernimmt eine anwendungsunabhängige Systemkomponente, etwa ein Interpreter. Faktenwissen und konzeptuelles Wissen eines wissensbasierten Systems sind in Form einer oder mehrerer Wissensbasen repräsentiert. Diese Wissensbasen können z.B. dazu dienen, um die im Architekturmodell von Fischer[19] genannten Wissensbereiche darzustellen, die für die Mensch-Computer-Kommunikation relevant sind. Das Wissen in den Wissensbasen ist auf vielfältige Weise verwendbar. Daher ist es möglich, daß viele unterschiedliche Systemkomponenten auf eine gemeinsame Wissensbasis zugreifen:

---

[19] siehe Seite 35

- Wenn der vom Rechner eingeschlagene Lösungsweg bei der Erschließung von Fakten in einer Wissensbasis repräsentiert ist, werden *Erklärungskomponenten* möglich, die eine Begründung für die vom System gefundenen Ergebnisse liefern können. Das regelbasierte System MYCIN [Buchanan, Shortliffe 84] und das auf ObjTalk-Constraints beruhende System FINANZ [Rathke C. 86] sehen beispielsweise derartige Erklärungskomponenten vor.
- *Dokumentationssysteme* [Fischer, Schneider 84b] erzeugen aus den in einer Wissensbasis explizit repräsentierten Konzepten und Fakten eine situations- und benutzerangepaßte Softwaredokumentation.
- Aktive und passive *Hilfesysteme* [Fischer, Lemke, Schwab 85] nutzen die Wissensbasis zur Generierung von Hilfeinformation. Es werden Hilfeinformationen über Konzepte des Anwendungsgebiets und der Interaktion mit dem System abgeleitet und dem Benutzer angeboten.
- *Metasysteme*, wie zum Beispiel das System TEIRESIAS [Davis, Lenat 82] oder das in Kapitel 5 beschriebene D&I Metasystem, und Wissenseditoren, wie das in Kapitel 6 beschriebene System ZOO, ermöglichen es, neues Wissen in eine Wissensbasis einzubringen und das bereits repräsentierte Wissen zu inspizieren.

Aus technischer Sicht besteht eine Wissensbasis aus einer Kollektion von Wissenseinheiten und einer Verwaltungskomponente für diese Wissenseinheiten [Bauer D. 86]. Die Zerlegung des gesamten Wissens in isolierbare und weitgehend deskriptive Wissenseinheiten ist von entscheidender Bedeutung für die Verständlichkeit und Zugänglichkeit des dargestellten Wissens. Es gibt eine Reihe von Wissensrepräsentationsmechanismen, die dazu dienen, derartige Wissenseinheiten darzustellen:

Produktionssysteme

Mit Produktionssystemen wird versucht, die Komplexität herkömmlicher Programmsysteme zu vermindern, indem das prozedurale Wissen in kleinere, voneinander unabhängige Einheiten, die Regeln, zerlegt wird. Regeln besitzen ein Muster (englisch *pattern*), das beschreibt, wann die Regel angewandt werden kann. Die Auswahl der anwendbaren Regeln wird durch eine Inferenzmaschine vorgenommen, die vom Anwendungsgebiet unabhängig ist. Die Anwendung einer Regel bewirkt, daß neue Fakten erschlossen werden und daß Aktionen ausgeführt werden. Mit Regeln läßt sich sehr gut inexaktes, heuristisches Wissen darstellen. Der Vorteil von Regeln besteht darin, daß sie auch für Wissensgebiete anwendbar sind, für die es keine geschlossene Theorie gibt. Regeln sind aber keine völlig deskriptiven Wissenseinheiten. Auch Regeln können letztlich nur beschreiben, *WIE* ein Problem zu lösen ist, nicht aber *WAS* die zugrundeliegenden Eigenschaften des Problemraums sind [Davis 82].

Logik

Die Verwendung von Logik ist ein Mittel, um die Beschränkungen eines Datenbanksystems zu überwinden. Das japanische Fifth-Generation-Computer-Projekt setzt auf den Einsatz der Logik-Sprache PROLOG [Clocksin, Mellish 84] in der Kombination mit dem relationalen Datenbankmodell, wobei mit Hilfe logischer Prädikate und Formeln Faktenwissen und konzeptuelles Wissen ausgedrückt werden kann und in der Datenbank gespeichert werden kann. Wissenseinheiten sind hier logische Formeln.

Frames

Ausgehend von den Vorstellungen Minskys, die in Abschnitt 3.1.2 beschrieben sind, repräsentieren framebasierte Sprachen wie FRL [Roberts, Goldstein 77] oder KRL [Bobrow, Winograd 77] Wissen auf deskriptive Weise in Form von Frames. Faktenwissen ist durch gefüllte Frames, konzeptuelles Wissen ist durch ungefüllte Frames repräsentiert. Es gibt eine Reihe von vordefinierten Operationen, um auf Frames zuzugreifen. Diese Operationen sind nicht durch Frames dargestellt, sie können aber Prozeduren auslösen, die in den Frames deklariert sind.

Objekte

Objektorientierte Sprachen stellen eine Erweiterung der Framesprachen um das Actor-Konzept von Hewitt dar [Hewitt 77]. Objekte sind (im Sinne von Actors) aktive Softwarestrukturen, die untereinander Botschaften austauschen. Zugleich haben Objekte aber die Funktion von Frames, d.h. sie sind imstande, Zustände zu speichern. Beispiele für derartige Sprachen sind Smalltalk [Ingalls 81], KL-ONE [Brachmann 78] oder ObjTalk [Rathke C. 86]. In Form von Objekten läßt sich Wissen repräsentieren. Dabei sind die deskriptiven und die prozeduralen Aspekte von Wissen berücksichtigt:

- Objekte (als Frames) besitzen *Merkmale* und Merkmalswerte, die in Form von sogenannten *Slots* dargestellt sind. Da Slots Referenzen auf andere Objekte enthalten können, läßt sich Wissen in Form von semantischen Netzen [Quillian 68] darstellen. Merkmale lassen sich beschreiben und lesen, Objekte lassen sich erzeugen und löschen. Dadurch kann das Wissen der Wissensbasis zu jedem Zeitpunkt manipuliert werden. Die Belegung von Slots mit Merkmalswerten berücksichtigt den deskriptiven Aspekt der Wissensdarstellung.

- Objekte (als Actors) verfügen über *Methoden*. Durch Botschaftenaustausch zwischen Objekten werden diese Methoden aktiviert. In den Methoden ist festgelegt, wie ein Objekt auf eine eintreffende Botschaft reagiert. Methoden lassen sich ansehen als Programmstücke, bei deren Ausführung entweder

Merkmale des betreffenden Objektes verändert werden oder aber neuerlich Botschaften an andere Objekte versandt werden. Daher kommt in den Methoden der prozedurale Aspekt des Wissens zum Ausdruck.

Hybride Repräsentationsmechanismen

In neuerer Zeit verwischen sich die Unterschiede zwischen den verschiedenen Wissensrepräsentationssystemen. Es gibt eine Tendenz hin zu sogenannten hybriden Systemen, die mehrere der beschriebenen Formalismen unterstützen. Beispiele für derartige Systeme sind LOOPS [Bobrow, Stefik 81], das den prozeduralen, den objektorientierten, den zugriffsorientierten und den regelbasierten Programmierstil unterstützt, und KEE [KEE 85], das auf der objektorientierten Repräsentation der UNITS [Stefik 79] aufbaut und in diesem Formalismus weitere Konzepte wie Regeln und logische Formeln zur Verfügung stellt. Einen ähnlichen Weg geht auch ObjTalk, das die objektorientierte Darstellung von Regeln und Constraints erlaubt und zur Darstellung zusätzlicher Konzepte erweiterbar ist.

3.4 Repräsentation von Wissen mittels ObjTalk

In unseren Systemen wird die Sprache ObjTalk zur Repräsentation von Wissen verwendet. Wie in Abschnitt 3.1 ausgeführt, lassen sich abhängig vom Abstraktionsgrad der zugrundeliegenden Informationen drei fundamentale Erscheinungsformen von Wissen, nämlich *Faktenwissen* und *konzeptuelles Wissen* und *Metawissen* voneinander unterscheiden. Alle drei Arten von Wissen lassen sich mit Hilfe von ObjTalk darstellen.

3.4.1 Faktenwissen

Zur Darstellung von Faktenwissen stehen in der Wissensrepräsentationssprache ObjTalk folgende Konzepte zur Verfügung:

- Objekte der realen Welt und Sachverhalte, in denen solche Objekte eine Rolle spielen, lassen sich durch *ObjTalk-Objekte* darstellen.
- Merkmale von Objekten lassen sich mit Hilfe der *Slots* darstellen, über die jedes ObjTalk-Objekt verfügt. Gleiches gilt für die Merkmale von Sachverhalten und für die Rollen, die Objekte in einem Sachverhalt spielen können. Der Wert eines Merkmals bzw. die Belegung einer Rolle mit einem Objekt läßt sich als sogenannter *Filler* des Slots darstellen. Die mit Hilfe von Slots dargestellten Eigenschaften können den Charakter von *Attributen* oder *Relationen* haben. Bei Attributen besteht der Filler des entsprechenden Slots

Objekt	Klasse	Slot	Filler
Brief-1	Brief	Absender	Wolf
		Empfänger	Michael
		Datum	15.02.86
		Betrifft	ZOO

Abbildung 3-5: Repräsentation eines Objekts in ObjTalk

Objekt	Klasse	Slot	Filler
Versenden-1	Versenden	Handelnder	Wolf
		Nutznießer	Michael
		Zeitpunkt	15.02.86
		Gegenstand	Brief-1

Abbildung 3-6: Repräsentation eines Sachverhalts in ObjTalk

aus einem elementaren Lisp-Ausdruck, z.B. einem Textsymbol oder einer Zahl. Im Fall von Relationen besteht der Filler des betreffenden Slots aus einem ObjTalk-Objekt oder einer (eventuell leeren) Liste von ObjTalk-Objekten.

- ObjTalk ermöglicht eine Klassifizierung von Objekten. ObjTalk-Objekte lassen sich in verschiedene *Klassen* einteilen. Gleichartige ObjTalk-Objekte gehören derselben Klasse an. Aus der Klassenzugehörigkeit leitet sich unter anderem ab, welche Slots ein ObjTalk-Objekt besitzen kann.

Die Abbildungen 3-5 und 3-6 zeigen Beispiele für die Repräsentation von Objekten und Sachverhalten mittels ObjTalk.

3.4.2 Konzeptuelles Wissen

Konzeptuelles Wissen wird in ObjTalk ebenfalls in objektorientierter Form dargestellt, und zwar mit Hilfe sogenannter konzeptueller Objekte.

Die wichtigsten konzeptuellen Objekte sind die *Klassen.* Klassen beschreiben die gemeinsamen Eigenschaften gleichartiger Objekte, die als *Instanzen* der

Objekt	Klasse	Slot	Filler
Brief	class	superc	Dokument
		slots	Absender Empfänger Betrifft Datum Inhalt
		methods	Versenden Löschen Editieren Beantworten
		rules	Automatische_Ablage

Abbildung 3-7: Repräsentation einer Klasse in ObjTalk

Klasse bezeichnet werden.[20] Klassen sind von ihrem Aufbau her ganz normale ObjTalk-Objekte. Ein Unterschied besteht jedoch darin, daß sich aus den Merkmalen einer Klasse die Eigenheiten ihrer Instanzen ableiten. Klassen stellen also Abstraktionen ihrer Instanzen dar.

Klassen stehen untereinander in einer Spezialisierungshierarchie.[21] In ObjTalk ist diese Hierarchie durch die Superklassenrelation *superc* definiert. Diese Relation ist mit Hilfe ganz normaler Slots repräsentiert, für ihre Darstellung ist also kein besonderes ObjTalk-Konzept nötig. Aus dem beschreibenden Wissen über die Spezialisierungshierarchie leitet sich das prozedurale Wissen der Vererbung ab. Die Instanzen einer spezialisierten Klasse haben in der Regel "mehr" Eigenschaften als die Instanzen einer weniger spezialisierten Superklasse; Eigenschaften werden also von den Superklassen ererbt.

Als weitere wichtige Sprachelemente von ObjTalk sind die *Slotbeschreibungen* zu nennen.[22] Slotbeschreibungen sind Objekte, welche die charakteristischen Eigenschaften von Slots festlegen, die für eine ganze Klasse von Instanzen definiert sind. Die Relation *slots* ordnet einer Klasse ihre Slotbeschreibungen zu.

---

[20]Der Begriff "Instanz" ohne weitere Angabe der zugehörigen Klasse wird üblicherweise in einem hierzu unterschiedlichen Sinn verwendet: Die Feststellung "I ist eine Instanz" gilt als gleichbedeutend mit der Aussage "I ist keine Klasse", obwohl jede Klasse Instanz einer anderen Klasse, nämlich der Klasse class ist [Stefik, Bobrow 85].

[21]Genauer gesagt handelt es sich um eine multiple Hierarchie oder Heterarchie, da eine Klasse mehr als eine Klasse besitzen kann, die sich direkt über ihr in der Hierarchie befindet.

[22]Im ObjTalk-Grundsystem sind Slotbeschreibungen derzeit als Assoziationslisten repräsentiert. In den im folgenden beschriebenen Systemen D&I und ZOO war es erforderlich, auch Slotbeschreibungen als Objekte anzusprechen. Daher wurde für diese Systeme vom Autor eine kompatible Erweiterung vorgenommen, die dies ermöglicht. Die im folgenden beschriebene ObjTalk-Version ist die neuere, die im System ZOO Verwendung findet. Die entsprechende Erweiterung im System D&I war auf den Anwendungsfall bezogen und mit dem D&I-spezifischen Konzept des Merkmals verknüpft, das noch in den nächsten zwei Kapiteln beschrieben wird.

Objekt	Klasse	Slot	Filler
Absender	slot	type	Person
		format	single
		inverse	
		defaultvalue	Wolf

Abbildung 3-8: Repräsentation einer Slotbeschreibung in ObjTalk

Jede Slotbeschreibung besteht aus einer Reihe von Beschreibungselementen. Diese sind repräsentiert durch die Slots der Slotbeschreibung. Es gibt eine Vielzahl von fundamentalen Beschreibungselementen, die z.B. in [Lemke 85] dargestellt sind und hier nicht aufgeführt werden sollen. Darüberhinaus gibt es höhere Beschreibungselemente wie das Attribut *format* und die Relationen *type*, *inverse* und *defaultvalue*: Das Attribut *format* legt fest, ob der Slot singuläre oder multiple Werte (Listen) annimmt. Die Relation *type* verweist auf eine Klasse, deren Instanzen als Filler des Slots in Frage kommen. Die Relation *inverse* kann auf eine andere Slotbeschreibung zeigen, die eine Umkehrrelation definiert. Der *defaultvalue* kann eine Instanz enthalten, die als Standardbelegung für den Filler des Slots dient (Abbildung 3-8).

In ähnlicher Weise wie die Slotbeschreibungen sind auch die *Methoden* einer Klasse definiert. Methoden sind über die Relation *methods* ihren Klassen zugeordnet. In den Methoden ist das prozedurale konzeptuelle Wissen eines wissensbasierten Systems repräsentiert. In den Methoden ist die Reaktion der Instanzen einer Klasse beim Eintreffen von Botschaften definiert. Methoden sind beschrieben durch einen *filter:*, der die Aktivierung regelt, sowie durch einen *body:*, der den Programmcode enthält, der die Reaktion des Objekts beschreibt. Ähnliches wie für die Methoden gilt auch für die *Regeln* in Objtalk. Regeln sind über die Relation *rules* ihrer Klasse zugeordnet. Regeln besitzen anstelle eines Filters eine Triggerbedingung, welche die Ausführbarkeit der Regel bestimmt.

Zuletzt sollen noch die *Constraints* erwähnt werden, die als eine Art Spezialfall der Klassen ähnlich wie normale Klassen aufgebaut und definiert sind. Constraints ermöglichen die Beschreibung und Aufrechterhaltung von Konsistenzbedingungen zwischen Slots mit Hilfe von automatisch ablaufenden ObjTalk-Regeln.

3.4.3 Metawissen

Metawissen läßt sich in ObjTalk durch vordefinierte konzeptuelle Objekte, die sogenannten Metaobjekte *object*, *class*, *Action*, *Method*, *Extended-Method*, *Rule*, *slot* und *constraint* darstellen. Diese Objekte sind von ihrem Aufbau her normale Klassen, sie haben aber für das ObjTalk-System fundamentale Bedeutung.

Die Klasse *object* ist die Wurzel der Spezialisierungshierarchie und beschreibt die gemeinsamen Eigenschaften aller Objekte in einem ObjTalk-System. Eine derartige Eigenschaft besteht z.B. darin, daß jedes Objekt einen Namen besitzen kann.

Die Metaklasse[23] *class* ist die Klasse aller Klassen, sie beschreibt die gemeinsamen Eigenschaften aller Klassen in einem ObjTalk-System. Eine derartige Eigenschaft besteht z.B. darin, daß jede Klasse die Slots *superc*, *slots*, *methods*, *rules* und *constraints* besitzt.

Die Klasse *slot* ist die Klasse aller Slotbeschreibungen.

Die Klassen *Action*, *Extended-Method*, *Method* und *Rule* beschreiben die gemeinsamen Eigenschaften bestimmter Mengen von Methoden und Regeln.

Die Metaklasse *constraint* ist die Klasse aller Constraints. Da Constraints eine besondere Art von Klassen sind, ist *constraint* eine Subklasse von *class*.

3.5 Externe Darstellung von Wissen

Jedes ObjTalk-Objekt ist systemintern durch eine Datenstruktur der Implementationssprache LISP repräsentiert. Der innere Aufbau dieser Datenstruktur ist eine Frage der Implementierung von ObjTalk, er soll dem Benutzer von ObjTalk verborgen bleiben. Entsprechend der objektorientierten Philosophie von ObjTalk können aber die Eigenschaften eines Objekts durch das Versenden einer Botschaft erfragt oder verändert werden.

Die interne Repräsentation eines Objektes unterliegt den räumlichen und zeitlichen Beschränkungen eines Betriebssystem-Prozesses, sie existiert nur innerhalb und während der Dauer des sie tragenden LISP-Prozesses. Wenn man Objekte

---

[23]Klassen, deren Instanzen wiederum Klassen sind, werden als Metaklassen bezeichnet.

permanent speichern, an andere LISP-Prozesse verschicken oder benutzergerecht auf einem Bildschirmgerät darstellen möchte, benötigt man eine für den jeweiligen Fall geeignete *externe Repräsentation* von Objekten.

Die knappste externe Darstellung eines Objekts ist dessen Name. Im Namen zeigen sich nach außen hin die Existenz und (sofern der Name eindeutig ist) die Identität des Objekts. Mit Hilfe eines Namens ist es möglich, von außerhalb des ObjTalk-Systems auf ein Objekt Bezug zu nehmen. Der Name gibt jedoch wenig Aufschluß über die Eigenschaften eines Objekts.

```
(ask Brief make: Brief-1 with:
     (Absender = Wolf)
     (Empfaenger = Michael)
     (Datum = 15.02.86)
     (Betrifft = ZOO))
```

Abbildung 3-9: Definition eines ObjTalk-Objekts

Eine vollständige externe Beschreibung eines Objekts stellt dessen Definition dar. In ObjTalk ist die Definition eines Objekts gegeben durch eine Botschaft, die, wenn sie an ihren Adressaten abgesandt wird, das Objekt von neuem erzeugen würde (Abbildung 3-9). Mit Hilfe der Definition ist es möglich, das Objekt vollständig zu rekonstruieren. In ObjTalk erhält man die Definition eines Objekts, indem man eine entsprechende Anforderung an das betreffende Objekt schickt. Das externe Format der Definition ist auf den Programmierer zugeschnitten, der Objekte definiert, auf Dateien speichert oder sich ausdrucken läßt. Für eine benutzergerechte externe Darstellung von Objekten ist das Definitionsformat jedoch weniger geeignet.

3.5.1 Benutzerschnittstellen zur Wissensübertragung

Die Darstellung von Wissen ist von entscheidender Bedeutung für die Benutzerschnittstelle von Softwaresystemen. Bei der Interaktion zwischen Mensch und Computer ist auf beiden Seiten Wissen repräsentiert: in der Vorstellung des *Menschen* und im Speicher des *Computers*. Das zu lösende *Problem* selbst ist Gegenstand dieses Wissens, es ist gegeben durch einen Ausgangszustand und einen Zielzustand von Objekten der realen Welt. In der Vorstellung des Menschen und in den Softwareobjekten des Computers ist Wissen vorhanden über diese Zustände im speziellen und über den Problembereich im allgemeinen.

Interaktive Computerbenutzung ist Wissensaustausch. Der Mensch versorgt den Computer mit Wissen, das dieser nutzt, um zusammen mit bereits vorhandenem Wissen neues Wissen zu inferieren. Der Computer stellt dem Menschen bereits vorhandenes und inferiertes Wissen zur Verfügung, wenn dieser es verlangt. Die Verbindung zwischen beiden Wissenswelten, der Vorstellungswelt des Benutzers und den Objekten im Innern des Computers, wird durch die Benutzerschnittstelle des Softwaresystems hergestellt.

In herkömmlichen Softwaresystemen geschieht diese Wissensübertragung in Form eines Fernschreiberdialogs auf einem Bildschirmgerät. Der Benutzer erteilt Kommandos und richtet Anfragen an die Maschine, der Computer wendet sich mit Statusmeldungen und Abfragen an den Benutzer. Die ausgetauschten Informationen repräsentieren in der Regel Änderungen des Wissensbestands. Die Benutzerschnittstelle ist dabei lediglich Durchgangsstation für die ausgetauschten Informationen, der Bildschirm enthält - ähnlich wie das Endlosformular eines Fernschreibers - das Protokoll der zuletzt ausgetauschten Nachrichten.

Ein anderes Bild ergibt sich, wenn die Benutzerschnittstelle von einer externen Darstellungsform des gespeicherten Wissens Gebrauch macht. Besonders gut ist dies möglich, wenn ein hochauflösender Bildschirm und ein Zeigeinstrument, beispielsweise eine Maus zur Verfügung steht. Anstelle eines Protokolls der Wissensänderungen wird der aktuelle Wissensbestand selbst auf dem Bildschirm dargestellt. Es werden neue Darstellungskonzepte möglich, die sich an den Bedürfnissen des Benutzers eines Anwendungssystems ausrichten:

Die Eigenschaften eines Objekts können, wie beispielsweise in dem im folgenden Kapitel vorgestellten System D&I, in einem *Bildschirmformular* dargestellt werden. Mit Hilfe von Formularen kann Wissen in deskriptiver Form in das System eingebracht werden. Die Verwendung von Formularen erlaubt eine einfache Spezifikation von Attributen und Merkmalen. Syntaktische Konventionen, die bei der Verwendung von formalen Sprachen auftreten, müssen nicht beachtet werden. Die Funktionalität eines Objekts, bestehend aus einer Auswahl von Botschaften, auf die das Objekt reagiert, kann mit Hilfe eines *Menüs* [Fabian, Rathke C. 83] visualisiert werden. Dabei wird das aktive Erinnerungsvermögen des Menschen entlastet. Die Auswahl aus Menüs und das Ausfüllen von Formularen ist einfacher als die eindeutige sprachliche Formulierung von Sachverhalten, da der Benutzer im Angebot der Manipulationsmöglichkeiten lediglich die wiedererkennen muß, die für ihn von Bedeutung sind. Der Einsatz von *Graphik* wie etwa im Wissenseditor ZOO, der in Kapitel 6 vorgestellt

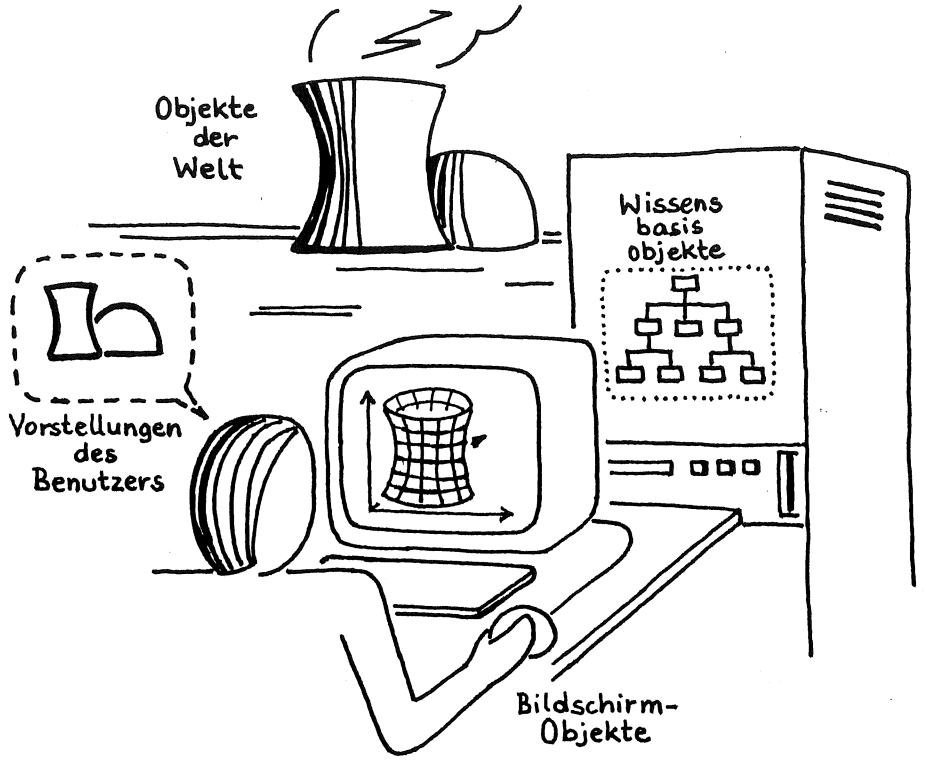


Abbildung 3-10: Multiple Repräsentationen von Wissen

wird, eröffnet eine weitere Dimension der benutzerorientierten Darstellung von Wissen. Das im System repräsentierte Wissen kann in einer gegenständlichen Darstellung auf dem Bildschirm erscheinen. Die Interaktion mit dem System geschieht dadurch, daß an den visualisierten Objekten Handlungen vorgenommen werden.

In allen diesen Fällen erscheint das intern gespeicherte Wissen in der Form von zweidimensionalen Objekten auf dem Bildschirm des Computersystems. Auf halbem Weg zwischen den Vorstellungen des Benutzers und den Softwareobjekten im Innern des Computers ist der Bildschirm ein dritter Ort, an dem Wissen über den Problembereich repräsentiert ist (Abbildung 3-10).

3.5.2 Direkte Manipulation

Die externe Darstellung von Wissen durch Bildschirmobjekte muß mit der zugehörigen rechnerinternen Repräsentation stets im Einklang stehen. Die Aufrechterhaltung der Konsistenz beider Darstellungen ist Aufgabe der Benutzerschnitt-

stelle eines Systems. Von Vorteil ist eine möglichst feste und unmittelbare Kopplung zwischen interner und externer Repräsentation, es wird dann eine neue Art der Interaktion mit dem System möglich, die als *direkte Manipulation* [Shneiderman 83; Hutchins et al. 86] bezeichnet wird:

- Wenn sich ein internes Datenobjekt ändert, wird dies an einem externen Bildschirmobjekt unmittelbar sichtbar.
- Der Benutzer kann mit Hilfe eines Zeigeinstruments Veränderungen an der externen Darstellung eines Objekts vornehmen, die sich automatisch auf den internen Zustand des Systems auswirken.

Die Interaktionsform der direkten Manipulation stellt Anforderungen an die Form der internen Repräsentation von Wissen:

- Es muß einfach feststellbar sein, welche internen Zustände von einer Benutzeraktion betroffen sind - am einfachsten ist dies möglich, wenn jedem Bildschirmobjekt ein internes Softwareobjekt zugeordnet ist. In einer objektorientierten Wissensrepräsentationssprache, wie es ObjTalk ist, läßt sich dies sehr einfach erfüllen.
- Änderungen am intern repräsentierten Wissen dürfen nicht zu teuer sein. Inkrementelle Änderungen am intern gespeicherten Wissen müssen leicht möglich sein und dürfen keinen langwierigen Compilationsvorgang erfordern. Als Wissensrepräsentationssprache kommt also nur eine Interpretersprache in Betracht, was nicht ausschließt, daß festbleibende Teile des Wissens sich dennoch compilieren lassen.
- Es muß einen Überwachungsmechanismus für interne Zustände geben, der beauftragt werden kann, die Benutzerschnittstelle zu aktivieren, wann immer eine Zustandsänderung eintritt. Nur so läßt sich gewährleisten, daß die Bildschirmdarstellung stets auf dem aktuellen Stand ist. Durch Constraint-Mechanismen wie die von ObjTalk oder durch an Zustände gebundene Prozeduren, beispielsweise durch sogenannte *active values* [KEE 85] läßt sich diese Forderung erfüllen.

Die Brauchbarkeit von Benutzerschnittstellen, die auf dem Prinzip der direkten Manipulation beruhen, hängt freilich davon ab, wie gut das externe Format zur Wissensdarstellung den Vorstellungen des Benutzers angepaßt ist. Sofern dies gelingt, stellt die Benutzerschnittstelle nicht nur einfach eine Verbindung her zwischen den Benutzervorstellungen und der internen Wissensdarstellung, vielmehr bringt sie die Wissenswelten von Mensch und Maschine unmittelbar in Kontakt. Wenn die Bildschirmdarstellungen den Vorstellungen des Benutzers von der realen Welt entsprechen, wird es für den Benutzer bedeutungslos, daß es in der Realität, in seinen Vorstellungen, auf dem Bildschirm und im Compu-

ter verschiedene Repräsentationen eines Phänomens gibt. Was der Benutzer auf dem Bildschirm sieht und womit er am Rechner hantiert, erscheint ihm nicht als ein Abbild, sondern als der Gegenstand selber. Diese Art der Interaktion wird von Fischer [Fischer 86] als *Mensch-Problembereich-Kommunikation* bezeichnet, wobei unklar ist, ob in dieser Situation der Begriff Kommunikation noch am Platze ist. Hutchins, Hollan und Norman verwenden hierfür die Bezeichnung *Direct Engagement* [Hutchins et al. 86].

4. Ein Informationssystem zur Verarbeitung von Sachwissen

4.1 Ein Modellfall

Das *Ostasien-Institut* in Bonn hat die satzungsgemäße Aufgabe, Informationen von nationalem Interesse aus dem ostasiatischen Raum zu beschaffen und diese der deutschen Wirtschaft und Verwaltung verfügbar zu machen. Von zunehmender Bedeutung sind hierbei Informationen aus Forschung und Entwicklung in Japan auf den Gebieten der sogenannten Zukunftstechnologien. Der Bedarf an Technologieinformation aus Fernost ist groß, aufgrund der Sprachbarriere ist er aber nur schwer zu decken. Dementsprechend wichtig ist die Aufgabe der Informationsakquisition in diesem Bereich.

Quellen der Information sind Reiseberichte von Institutsmitarbeitern, sowie japanische Veröffentlichungen und japanische Patentschriften; in neuester Zeit sind auch Auszüge aus japanischen Datenbanken verfügbar. Die Verarbeitung der Informationen im Ostasien-Institut erfolgt weitgehend manuell. Die einzelnen Mitarbeiter des Instituts sichten, übersetzen und archivieren das vorhandene Material. Daher ist ein großer Teil des erworbenen Wissens auf viele persönliche Informationsablagen verteilt.

Die japanischen Anstrengungen auf dem Forschungsgebiet der *fünften Computergeneration* sind ein typisches Beispiel für die Thematiken, mit denen sich das Ostasien-Institut befaßt. Für die Kenntnis dieses Sachgebiets ist Wissen über die Eigenschaften und Querbeziehungen von Institutionen, Personengruppen, Individuen und Gegenständen, sowie über Prozesse, Ereignisse und Situationen aus diesem Sachgebiet erforderlich:

- Firmen und Institutionen, die mit der Forschung und Entwicklung befaßt sind,
- Forschungs- und Entwicklungsprojekte und deren Status,
- Forscher und Forschergruppen,
- Förderinstitutionen, Förderungsprogramme,
- Kapitalflüsse, Firmenbeteiligungen, Umsätze,
- Firmenpolitik: Kooperationen und Konkurrenzsituationen,
- Produkte (z.B. auf dem Gebiet VLSI).

Bei der Erarbeitung und Verwaltung derartiger Wissensbestände tritt eine Reihe von besonderen Problemen in Erscheinung:

- Die zu verarbeitenden Informationen sind inhomogen und schlechtstrukturiert. Sie stammen aus verschiedenen Quellen und haben daher eine unterschiedliche äußere Form und treffen zu unterschiedlichen Zeiten ein. In der Regel handelt es sich um Mischungen aus textuellen Darstellungen und tabellarischen Übersichten.
- Die Informationen beschreiben komplexe Zusammenhänge, die nur bei entsprechenden Vorkenntnissen verstanden werden können. Zudem sind die Textquellen fremdsprachlich. Bei der Verarbeitung der Texte benötigt der Sachbearbeiter Hilfe zur Übersetzung und zum Verständnis der Texte.

Im Ostasien-Institut sind viele Personen gleichzeitig mit den Dokumentations- und Informationsaufgaben beschäftigt. Damit unterschiedliche Personen mit denselben Informationsinhalten in sinnvoller Weise umgehen können, müssen diese ein einheitliches Schema bei den Arbeitsabläufen einhalten. Insbesondere die Frage, wie Informationen klassifiziert und abgelegt werden, bedarf einer Regelung, die von allen Mitarbeitern gleichermaßen beachtet wird.

Eine computerunterstütztes System zur Wissensverarbeitung kann helfen, das Problem der Arbeitsorganisation von Dokumentations- und Informationsaufgaben zu lösen. Der Sachbearbeiter bräuchte nicht mehr eine Vielzahl von Konventionen zu erlernen, es genügte, daß das Computersystem diese kennt und dem Benutzer zum entsprechenden Zeitpunkt die adäquaten Arbeitsmethoden anbietet.

4.2 Zweckbestimmung des D&I-Systems

Technisches Ziel der hier dargestellten Forschungsarbeit war es, ausgehend von einem praktischen Anwendungsfall ein auf einem Computer lauffähiges experimentelles Informationssystem zur Wissensverarbeitung zu entwickeln. Dieses technische Ziel ordnet sich dem allgemeineren wissenschaftlichen Ziel unter, grundlegende Prinzipien der Wissensverarbeitung zu erforschen, und zwar in zweierlei Hinsicht: Zum einen lassen sich diese Prinzipien besser am Beispiel konkreter Benutzeranforderungen ableiten. Um die Resultate einer solchen wissenschaftlichen Untersuchung überprüfen zu können, ist es zum andern sinnvoll, diese an einem prototypischen Programmsystem zu demonstrieren und fachlicher Kritik auszusetzen.

Bei der Festlegung der Anforderungen an das prototypische Programmsystem wurde von der in den Abschnitten 2.1 und 4.1 vorgestellten Problemstellung eines Dokumentations- und Informationsinstituts ausgegangen. Das Arbeitsziel

bestand darin, einen automatischen Digester und Informanten (D&I) als prototypisches System zur Unterstützung von Dokumentations- und Informationsaufgaben zu entwickeln. Dabei stehen die Begriffe *Digester*[24] und *Informant* für die beiden Hauptaufgaben des Systems, den Erwerb und die Nutzung von Fachwissen.

Die Anwendbarkeit des Systems D&I beschränkt sich in der Grundversion auf die Verarbeitung von Berichten aus der Computerindustrie; die betrachtete Wissenswelt umfaßt die Beschreibung von Sachverhalten und Ereignissen, die Firmen, Personen, Produkte und Projekte aus der Computerszene zum Gegenstand haben. Ein Entwurfskriterium für das System D&I war jedoch dessen Erweiterbarkeit und Adaptierbarkeit, so daß die Konzentration auf ein Anwendungsgebiet keine wirkliche Einschränkung der Eigenschaften des Systems darstellt.

4.2.1 Anforderungen an die Benutzerschnittstelle

Für die Verarbeitung von Informationen, die in Form von Texten vorliegen, sind verschiedene Arten von Schnittstellenmodellen denkbar. Am weitesten geht hierbei die Vorstellung einer vollautomatischen inhaltlichen Verarbeitung von Texten: Die Quellinformationen, welche als Druckerzeugnisse vorliegen, werden mit einem Scanner eingelesen und mit Methoden der Symbolverarbeitung in eine Zeichenfolge gewandelt. Mit Hilfe eines Analysesystems zum Verständnis natürlichsprachlicher Texte wird die Information in eine semantischer Repräsentation umgewandelt. Das System ist imstande, in natürlicher Sprache formulierte Fragen zu allen gespeicherten Informationsinhalten zu beantworten.

In einem derartigen hypothetischen System tritt eine Mensch-Computer-Interaktion lediglich bei Anfragen eines Informationssuchenden auf. Mit Ausnahme der Auswahl von Publikationen, die als Quellinformationen dienen, wird die eigentliche Dokumentations- und Informationstätigkeit voll vom Computer übernommen. In den hier beschriebenen Forschungsaktivitäten wird die Zielvorstellung eines solchen vollautomatischen Systems zur Wissensverarbeitung nicht verfolgt, und zwar aus mehreren Gründen:

- Der technologische Fortschritt auf dem Gebiet der Natursprachverarbeitung und der maschinellen Intelligenz ist noch nicht so weit, daß das skizzierte Ziel in absehbarer Zeit erreichbar wäre.

---

[24] to digest (engl.) = verdauen

- In einem derartigen System verliert der Mensch leicht die Kontrolle über die Vorgänge der Wissensverarbeitung. Es ist daher fraglich, ob es überhaupt Anwendungsfälle gibt, in denen ein solches System erwünscht ist.
- Das eigentliche Forschungsziel ist die Entwicklung von Prinzipien und Methoden der Wissensverarbeitung. Ein prototypisches System zur Demonstration dieser Prinzipien und Methoden sollte sich auf die mit diesen Fragen zusammenhängenden Probleme beschränken.

Die Schnittstellenspezifikation des System D&I beruht infolgedessen auf einem kooperativem Modell. Ein Mensch führt die Verarbeitung eigenverantwortlich, aber mit maschineller Unterstützung durch. Das System stellt Informationen und Werkzeuge zur Wissensverarbeitung bereit. Der Benutzer vollzieht den Verarbeitungsprozeß, indem er eine Auswahl aus dem Angebot von Funktionen und Informationen trifft.

Der typische Benutzer des Systems ist Sachbearbeiter. Er soll mit Hilfe des D&I die eintreffende Fachinformation aus einem Sachgebiet archivieren und bei Bedarf auch wiedergewinnen können. Der Sachbearbeiter ist Experte auf seinem Sachgebiet, aber Laie auf dem Gebiet der Datenverarbeitung.

Beim Entwurf des Systems D&I wurde deshalb folgendes Benutzungsmodell zugrundegelegt: Das System unterstützt den Sachbearbeiter bei der *Archivierung, Erarbeitung und Zusammenfassung* von Zeitungsmeldungen, Agenturmeldungen oder ähnlichen stückweise eintreffenden Sachinformationen. Es entlastet den Bearbeiter bei der Aufgabe, diese Informationen zu *abstrahieren*, sie in eine von der Natursprache unabhängige Darstellung überzuführen und sie in eine Datenstruktur einzubetten, die das angesammelte Wissen repräsentiert. Es ermöglicht dem Benutzer, Ausschnitte des angelegten Wissensbestands abzurufen und in einer geeigneten Darstellung zu betrachten.

4.2.2 Anforderungen an die Funktionalität

Die Funktionalität des Systems D&I ist durch die beiden Begriffe *Digester* und *Informant* bestimmt.

Digester

Der Begriff *Digester* steht für die Funktionalität des Systems D&I als unterstützendes Werkzeug bei den Vorgängen des Wissenserwerbs und der Modellierung des Wissens in der Wissensbasis des Systems. Im System D&I treten diese Vorgänge an zwei Stellen in Erscheinung:

1. Bei den Dokumentationsaufgaben liegen die zu verarbeitenden Fachinformationen in unstrukturierter Form als Inhalt von Zeitungsartikeln und anderen Veröffentlichungen vor. Mit Hilfe der Digester-Komponente kann ein Sachbearbeiter die eintreffende Fachinformation in eine strukturierte Form überführen und in eine Faktenwissensbasis einbringen. Diese Art von Wissenserwerb wird durch das D&I Anwendungssystem unterstützt.

2. Bei der Anpassung des Systems auf neue Anwendungsgebiete stellt der Digester ein Werkzeug zum Erwerb des erforderlichen gebietsspezifischen Wissens dar. Ein Organisationsexperte kann das System so in seiner Funktionalität umgestalten und auf andere Anwendungsgebiete anpassen. Für diese Aufgabe steht das D&I Metasystem zur Verfügung, das in Kapitel 5 beschrieben wird.

Ziel der Forschung war es, die Wissenserwerbskomponente des Systems so zu gestalten, daß der Benutzer beide Aufgaben lösen kann, ohne DV-Spezialist sein zu müssen. Zu diesem Zweck bietet der Digester eine Unterstützung auf verschiedenen Ebenen an.

Der Digester soll eine *inhaltliche Verarbeitung* von Sachinformationen ermöglichen. Das heißt, nicht die Quelltexte selbst, sondern deren Bedeutungen werden in der Wissensbasis des Systems abgespeichert. Die Digestkomponente unterstützt den Sachbearbeiter bei diesem Abstraktionsvorgang, wobei sie auf bereits im System gespeichertes Fachwissen zurückgreifen kann. Dabei muß der Digester *intelligent* sein im ursprünglichen Wortsinn, demzufolge Intelligenz die Fähigkeit ist, zwischen den Zeilen zu lesen: Das heißt, eine wichtige Aufgabe des Digesters besteht darin, implizite Informationen explizit zu machen. Schließlich übernimmt der Digester die *Ablage* der Informationsinhalte auf den Dateien des Rechnersystems.

Informant

Die Nutzung des im System D&I gespeicherten Wissens wird möglich durch seine Funktion als *Informant*. Aufgabe des Informanten ist es, die gespeicherten Informationen entsprechend den Wünschen des Benutzers aufzufinden und wahrnehmungsgerecht darzubieten. Besondere Darstellungsarten werden erforderlich, um die Komplexität der im System gespeicherten Informationen zu reduzieren:

- Durch die Wahl von *Perspektiven* werden ausgewählte Teile des Wissensbestands in den Mittelpunkt der Betrachtung gestellt.
- Durch sogenannte *Filter* können unwesentliche Teilinformationen ausgeblendet und wichtige Informationen sichtbar gemacht werden.

Die Auffindung der im System abgelegten Informationen ist auf verschiedene Weisen möglich:

1. durch die eindeutige Bezeichnung einer Wissenseinheit, z.B. durch Angabe eines Namens.
2. durch unvollständige Bezeichnung einer Wissenseinheit und Auswahl aus den verbleibenden Möglichkeiten.
3. Durch den Übergang von einer Wissenseinheit zu einer benachbarten.

Informant und Digester sind keine getrennten Betriebsarten des Systems D&I, beide Funktionen sind gleichzeitig verfügbar. Die Informant-Funktion des Systems bietet daher auch eine Unterstützung des Digest-Vorgangs. Der Informant hält *Hilfe zum Verständnis* der zu verarbeitenden Informationen bereit. Es ist dem Benutzer möglich, bei der Verarbeitung von Quelltexten bereits verarbeitete Informationen einzusehen, um die zu erfassenden textlichen Informationen im Kontext des bereits bekannten Wissens begreifen und an die passende Stelle in der Wissensbasis einbringen zu können. Diese Verständnishilfe schließt gegebenenfalls[25] eine *Übersetzungshilfe* ein: Das System muß imstande sein, auch fremdsprachliche Bezeichnungen darzubieten und als Eingabe zu akzeptieren.

4.2.3 Entwurfskriterien der Wissenskomponente

Das D&I-System läßt sich in zwei Komponenten zerlegen: in seine Benutzerschnittstelle und in seinen Kern, der für die eigentliche Wissensverarbeitung zuständig ist. Dieser Kern soll im folgenden als *Wissenskomponente* bezeichnet werden. Die Aufgaben der Wissenskomponente sind typisch für eine große Zahl von wissensverarbeitenden Systemen. Die Wissenskomponente des D&I-Systems berücksichtigt zweierlei Arten von Anforderungen:

1. *Mittelbare Anforderungen vom Benutzer des D&I-Systems:* Beim Entwurf der Wissenskomponente müssen die Eigenschaften der Benutzerschnittstelle berücksichtigt werden.
2. *Unmittelbare Anforderungen vom Entwerfer eines wissensbasierten Systems:* Dieser erwartet ein Baukastensystem zur Konstruktion von wissensverarbeitenden Systemen.

---

[25]Obwohl diese Forderung mit den vorhandenen Mitteln einfach zu erfüllen ist, wurde sie bisher nicht verwirklicht.

Insgesamt wurden bein Entwurf der Wissenskomponente folgende Kriterien zugrundegelegt:

- Faktenwissen muß inhaltlich, unabhängig von einer textuellen Form darstellbar sein.
- Das System muß für die Darstellung verschiedener Arten von Faktenwissen anpaßbar sein.
- Das im System vorhandene Wissen muß extern darstellbar sein.
- Die Wissenskomponente muß über Inferenzmechanismen verfügen, z.B. zur Konsistenzerhaltung des Wissens.
- Die Wissenskomponente muß die Einbettung neuer Wissenseinheiten in die Wissensbasis unterstützen.
- Die Wissenskomponente benötigt Mechanismen zum Auffinden und Vergleichen von Wissenseinheiten.

4.3 Verarbeitung von Sachwissen mit dem D&I Anwendungssystem

Am Beispiel eines einfachen Zeitungsartikels soll der Vorgang des Erwerbs von Sachwissen mit Hilfe des Systems D&I deutlich gemacht werden. Der betreffende Artikel ist am 7. Januar 1982 in der *Computerzeitung* erschienen:

> *Dipl.-Ing. Benno Durst*, 42, übernahm die *Leitung des Werkes für Medizinelektronik* der *Hewlett-Packard GmbH*, Böblingen. Nach HP-Angaben wurde sein Vorgänger, *Dr. Otto Brand*, zum *Leiter des Werkes für Tischcomputer* berufen.[26]

Zeitungsartikel wie dieser beschreiben Objekte aus einem bestimmten Sachgebiet und Ereignisse, die diese Objekte betreffen [Rosenberg 77]. Der Begriff "Objekt" ist dabei in einem allgemeinen Sinn zu verstehen, der alle Objekte unseres Denkens umfaßt, welche unverwechselbare Eigenschaften tragen; der Leser mag ihn nach Belieben durch die Worte Gegenstand, Wesenheit, Gebilde oder begriffliche Einheit ersetzen, wann immer eines von diesen treffender erscheint.

---

[26]Die Namen der beiden Personen wurden vom Verfasser geändert.

4.3.1 Die Wissensbasis

Die in diesem Zeitungsartikel erwähnten *Objekte* sind die Herren Durst und Brand, deren frühere und jetzige Tätigkeiten, sowie die Hewlett-Packard GmbH. Jedes dieser Objekte läßt sich einem bestimmten *Typ* zuordnen. In der Zeitungsmeldung ist insgesamt von drei Typen die Rede: PERSON, TÄTIGKEIT und FIRMA. Zwei *Ereignisse* werden beschrieben: Die Neubesetzung einer Stelle und ein Stellenwechsel. Ziel des Wissenserwerbs mit dem D&I ist es, diese Objekte und die Implikationen der beschriebenen Ereignisse in der *Faktenwissensbasis* des Systems D&I zu repräsentieren.

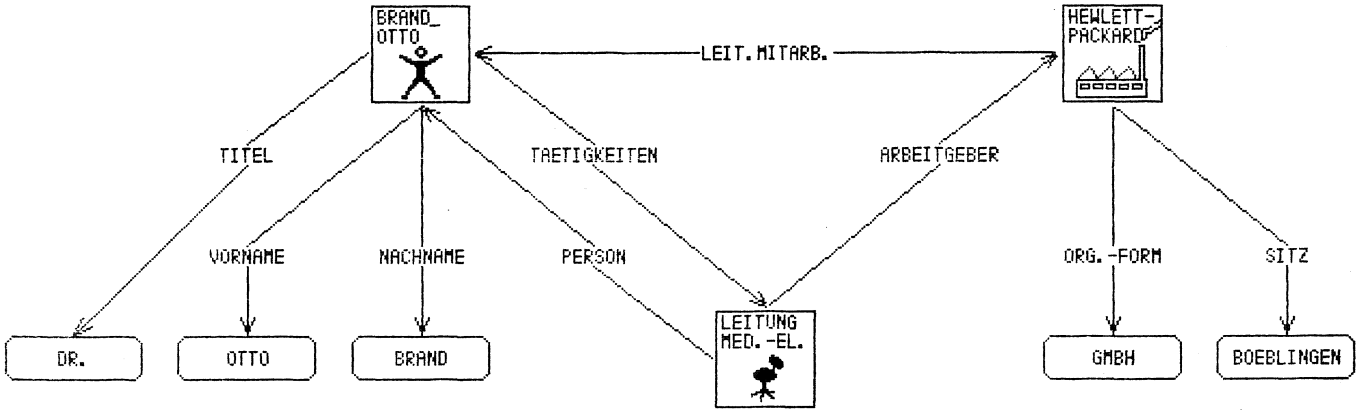


Abbildung 4-1: Ursprünglicher Zustand der Faktenwissensbasis

Wir wollen nun einmal annehmen, daß die bisherige Tätigkeit von Dr. Brand als Leiter des Werks für Medizinelektronik der Firma HP dem System bereits aufgrund einer früher verarbeiteten Zeitungsmeldung bekannt ist. Das heißt, in der Wissensbasis sind bereits drei Objekte angelegt, die PERSON Dr. Brand, die TÄTIGKEIT "Leitung Medizinelektronik" und die FIRMA HP (siehe Abbildung 4-1). Diese Objekte tragen *Merkmale*, die mit *Merkmalswerten* belegt sind. Merkmale einer Person sind beispielsweise ihr Titel, ihre Adresse oder ihre berufliche Tätigkeit. Die Merkmalswerte sind entweder Texte wie der Titel "Dr." oder aber Bezeichnungen anderer Objekte. Auf die letztere Art lassen sich Beziehungen zwischen Objekten darstellen, wie etwa die Tatsache, daß Dr. Brand leitender Mitarbeiter der Firma HP ist. Die Objekte bilden auf diese Weise ein Netz.

Die Verarbeitung der in dem Zeitungsbericht beschriebenen Ereignisse mit dem Digester soll nun zwei Arten von Änderungen an der Wissensbasis bewirken:

1. Es werden neue Objekte mit ihren Merkmalen in der Wissensbasis angelegt und mit den vorhandenen Objekten verknüpft.
2. Die Objekte in der Wissensbasis erhalten zusätzliche Merkmale beziehungsweise bereits vorhandene Merkmale erhalten neue Werte.

Wie dies vor sich geht, ist in den folgenden Abschnitten näher beschrieben.

4.3.2 Die Benutzerschnittstelle

Abbildung 4-2 zeigt den Bildschirmaufbau des Systems D&I bei einer derartigen Verarbeitung. Der Bildschirm ist aufgeteilt in vier Bereiche, die auch als *Fenster* bezeichnet werden:

1. Das *Arbeitsfenster* (links) enthält ein Formular, das einen Aspekt, das heißt einen Teil der gesamten Merkmale und Merkmalswerte des gerade betrachteten Objekts in einzelnen Formularfeldern zeigt.
2. Das *Skriptfenster* (Mitte) enthält ein Menü, dessen Inhalt vom gerade gezeigten Aspekt abhängig ist. Das Menü enthält die Namen von sogenannten Skripts; dabei handelt es sich um Funktionen, die der Benutzer auswählt, um die zu verarbeitenden Ereignisse zu interpretieren.
3. Das *Kommandomenü* (rechts) bietet kontextunabhängige Funktionen zur Bearbeitung der Wissensbasis, zum Rückgängigmachen fehlerhafter Eingaben und zum Abruf von Hilfe und allgemeinen Informationen an.
4. Das *Kopffenster* (oben) enthält Statusinformationen, nämlich den Namen des aktuell bearbeiteten Objekts, seinen Typ (z.B. Person, Firma etc.) und die Bezeichnung des gewählten Aspekts. Außerdem werden dort kurze Abfragen abgewickelt.

Zu jedem Zeitpunkt kann über eine *Hilfetaste* ein kontextabhängiger Hilfetext abgerufen werden, der den Bildschirm teilweise überlagert. Falls das System die Eingabe von Werten erwartet, erscheint zusätzlich ein Menü von Eingabeparametern, unter denen der Benutzer auswählen kann. Mittels einer *Abbruchtaste* kann jeder eingeleitete Verarbeitungsschritt wieder abgebrochen werden, ohne daß inkonsistente Systemzustände entstehen.

Eine vollständige Beschreibung der Benutzerschnittstelle des Systems D&I findet sich in [Csima 83].

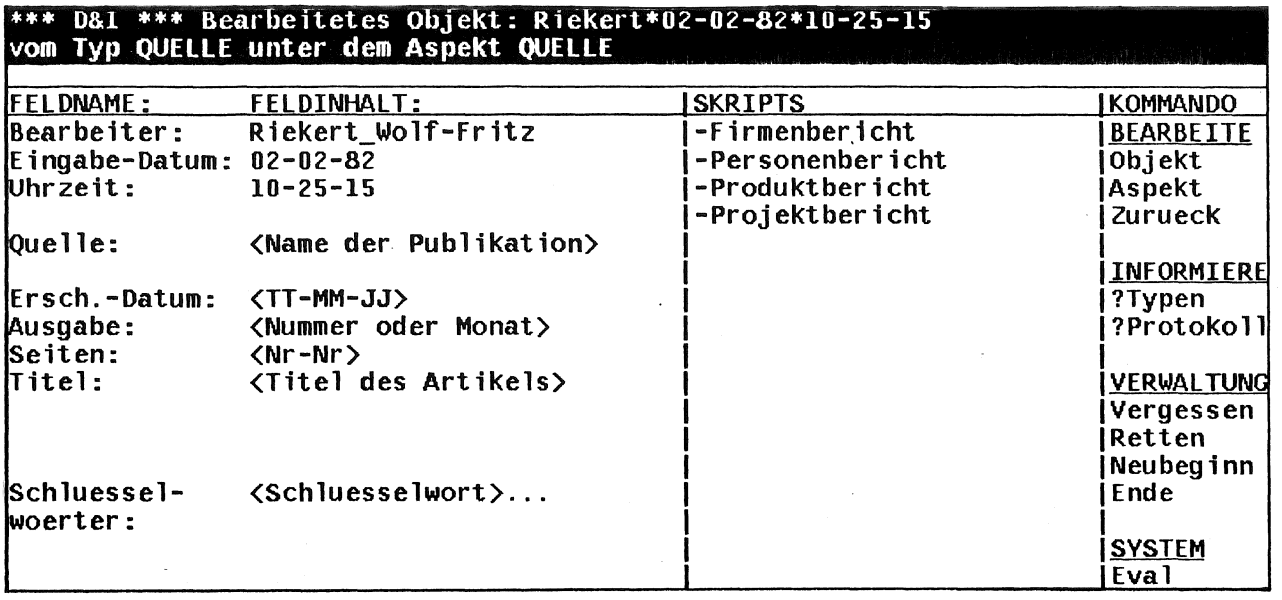


Abbildung 4-2: Bildschirmaufbau des Systems D&I

4.3.3 Identifikation von Objekten

Im Grundzustand enthält das Skriptfenster ein Menü aller verarbeitbaren Arten von Berichten. Der Sachbearbeiter wählt das Skript "Personenbericht" aus. Im sogenannten Skriptdialog, der daraufhin im Kopffenster abläuft, wird der Name einer beteiligten Person erfragt. Der Sachbearbeiter gibt den Namen "Durst—Benno" ein. Das System meldet, daß ihm die genannte Person noch nicht bekannt ist, und bittet den Sachbearbeiter zu bestätigen, daß ein entsprechendes Objekt in der Wissensbasis neu erzeugt werden soll.

Dies ist ein ganz typischer Vorgang beim Erwerb von Wissen: Es liegt ein Objekt vor, von dem bestimmte Eigenschaften bekannt sind. Zunächst muß das Objekt nach seinem Typ *klassifiziert* werden. Dieser Vorgang wird vom Digester durch das angebotene Skriptmenü unterstützt: Die Auswahl des Skripts Personenbericht klassifiziert das vorliegende Objekt als Person. Anschließend muß das Objekt *identifiziert* werden. Dies geschieht, indem seine Eigenschaften mit denen der bereits bekannten Objekte verglichen werden. Im gegebenen Fall sucht der Digester nach einer Person mit gleichem oder ähnlichem Namen. Wenn die Suche wie im vorliegenden Fall fehlschlägt, wird ein neues Objekt in der Wissensbasis angelegt.

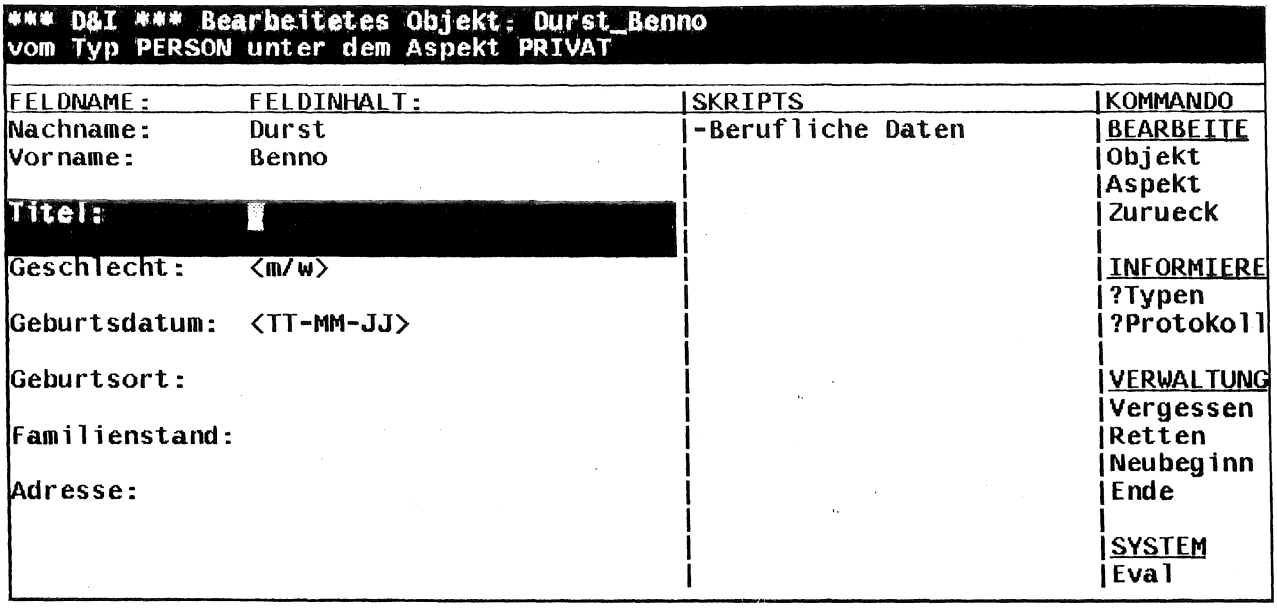


Abbildung 4-3: Der Aspekt PRIVAT der Person Benno Durst

4.3.4 Aspekte und Skripts

Als Folge des Skriptaufrufs wechseln die Bildschirmfenster ihren Inhalt und beziehen sich nun auf die Person Benno Durst als Privatmann (siehe Abbildung 4-3). Es wird nun ein anderer Ausschnitt der Wissensbasis, der *Aspekt* PRIVAT der Person dargestellt. Aspekte haben die Funktion von *Filtern*: sie gewährleisten, daß nur diejenigen Informationen sichtbar werden, die in der gegenwärtigen Dialogphase relevant sind.

Diese Filterwirkung betrifft zum einen den Inhalt des Arbeitsfensters. Dort sind nur die Merkmale zu sehen, die für Benno Durst als Privatperson von Belang sind: Die Merkmale "Vorname:" und "Nachname" sind bereits als Ergebnis des Identifikationsvorgangs gefüllt. Andere Angaben, wie z.B. der "Titel:" der Person können noch vom Benutzer eingetragen werden.

Die Art der gerade bearbeiteten Information ist auch ein Kriterium dafür, welche Verarbeitungsschritte voraussichtlich als nächste vom Benutzer vorgenommen werden. Daher bestimmt der aktuelle Aspekt auch über den Inhalt des Skriptmenüs. Das Skriptmenü enthält ein Angebot von Operationen, die voraussichtlich zu diesem Zeitpunkt der Verarbeitung benötigt werden.

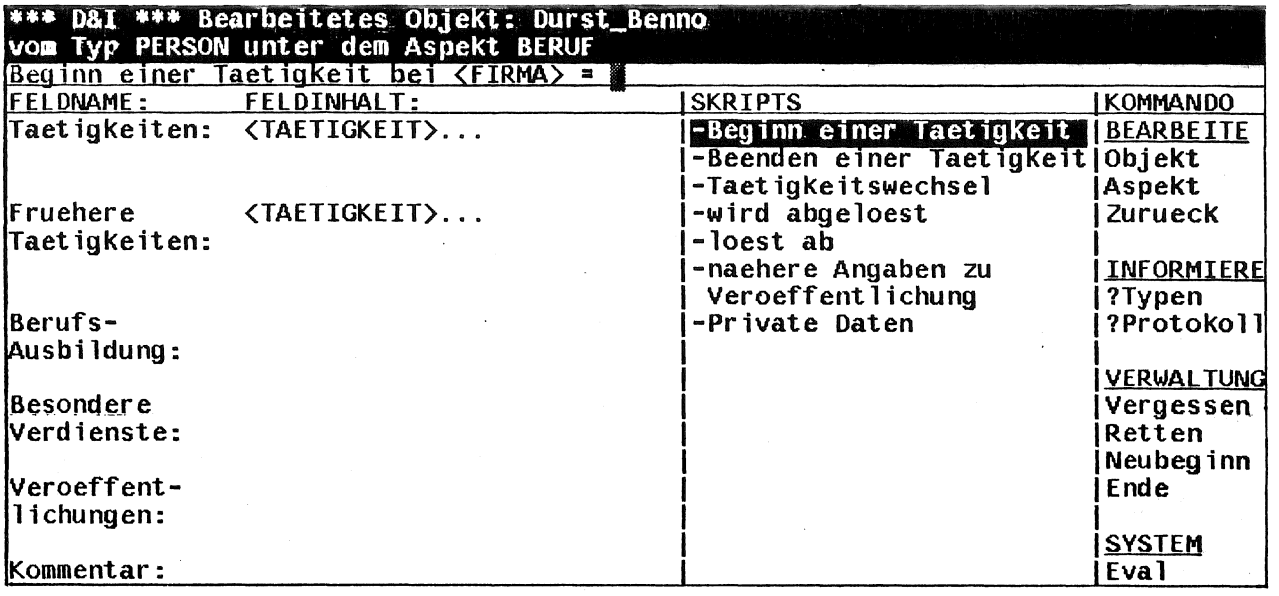


Abbildung 4-4: Der Aspekt BERUF der Person Benno Durst

Es lassen sich zwei Wirkungsweisen von Skripts unterscheiden, die auch gemeinsam auftreten können. Zum einen dienen Skripts zur Interpretation von Sachverhalten, indem sie eine entsprechende Modifikation der Wissensbasis ermöglichen. Skripts mit dieser Eigenschaft bezeichnen wir als *Interpretationsskripts*. Zum andern bewirken Skripts einen Kontextwechsel, indem sie auf einen neuen Aspekt der Wissensbasis umschalten. Solche Skripts tragen den Namen *Navigationsskripts*. Das oben beschriebenen Skript "Personenbericht" gehört zu beiden Kategorien. Es führt die Identifikation einer Person durch und bewirkt den Wechsel zum Aspekt PRIVAT.

4.3.5 Eintrag von Merkmalen

Mit Hilfe des Navigationsskripts "Berufliche Daten" gelangt man zum Aspekt BERUF der Person Benno Durst (Abbildung 4-4). Auf dem Bildschirm wird unter anderem das Skript "Beginn einer Tätigkeit" angeboten. Der Sachbearbeiter wählt dieses Skript aus. Daraufhin werden die Berufsbezeichnung und der Name des Arbeitgebers im Kopffenster erfragt. Wenn diese Argumente eingegeben sind, ist der Sachbearbeiter bei einem Aspekt angelangt, der die neuangetretene Tätigkeit von Herrn Durst zeigt (Abbildung 4-5). Die Berufsbezeichnung "Leitung des Werks Medizinelektronik" und der Arbeitgeber "Hewlett-Packard" wurden bereits bei der Identifikation der Tätigkeit vom Sy-

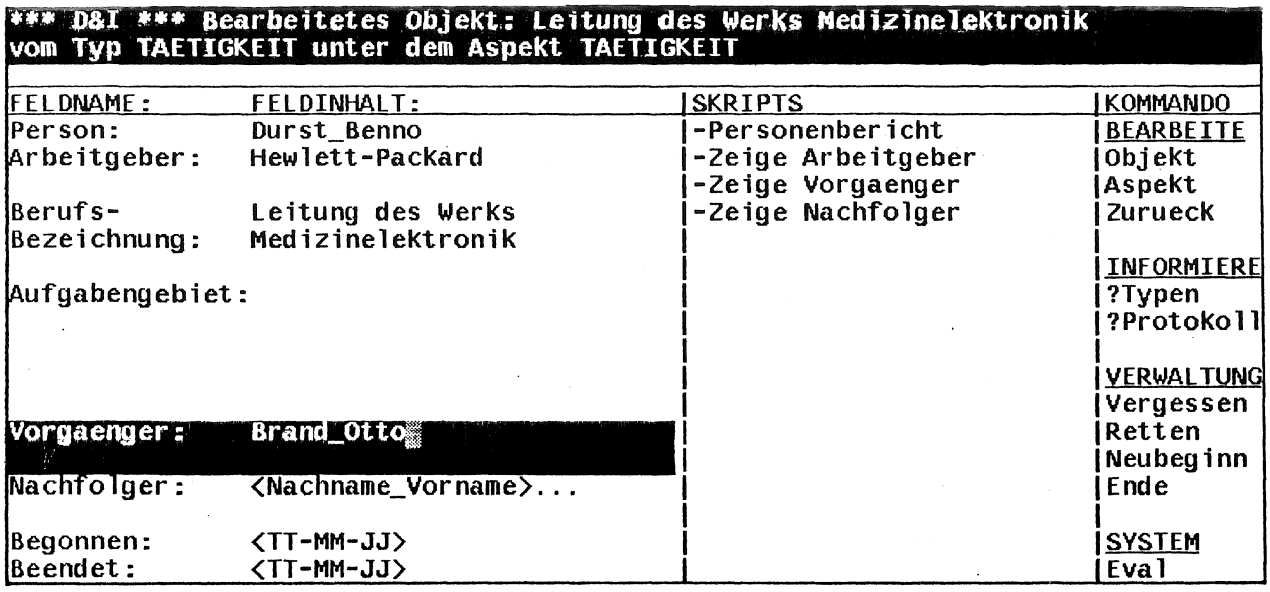


Abbildung 4-5: Darstellung einer Tätigkeit

stem erfragt und sind schon in das Formular im Arbeitsfenster eingetragen. Die diese Tätigkeit ausübende Person ist Herr Durst selbst, das entsprechende Feld wurde vom System automatisch gefüllt.

Die übrigen Formularfelder sind nicht mit Informationen gefüllt und können vom Benutzer beschrieben werden. Das Formularfeld "Vorgänger:" ist mit dem Syntaxvorschlag <Nachname—Vorname> vorbelegt. Der Sachbearbeiter gibt den Namen Brand—Otto ein. Diese Eintragung wird vom System ohne Rückfrage akzeptiert, da es die Person ohne weiteres identifizieren kann. Die neu in die Wissensbasis übernommenen Objekte sind nun mit den bereits vorhandenen verknüpft. Das Beschreiben des Formularfelds bewirkte die unmittelbare Veränderung des zugehörigen Merkmalswertes.

4.3.6 Inferenzen des Systems

Der Eintrag des Vorgängers bewirkte jedoch noch mehr als nur die Veränderung des zunächst betroffenen Objekts, also der Tätigkeit. Aus der Eigenschaft, daß Herr Brand Vorgänger von Herrn Durst ist, kann geschlossen werden, daß eine gleichartige Tätigkeit von Herrn Brand existieren muß. Diese wird vom Digester auch identifiziert, da Berufsbezeichnung und Arbeitgeber übereinstimmen. Wäre die Tätigkeit nicht bekannt gewesen, so hätte das Sy-

stem eine solche neu angelegt. In dieser Tätigkeit kann nun automatisch als Nachfolger Herr Durst eingetragen werden. Da nun der Nachfolger feststeht, wechselt die Tätigkeit ihre Rolle als jetzige Tätigkeit von Herrn Brand und wird zu einer früheren Tätigkeit.

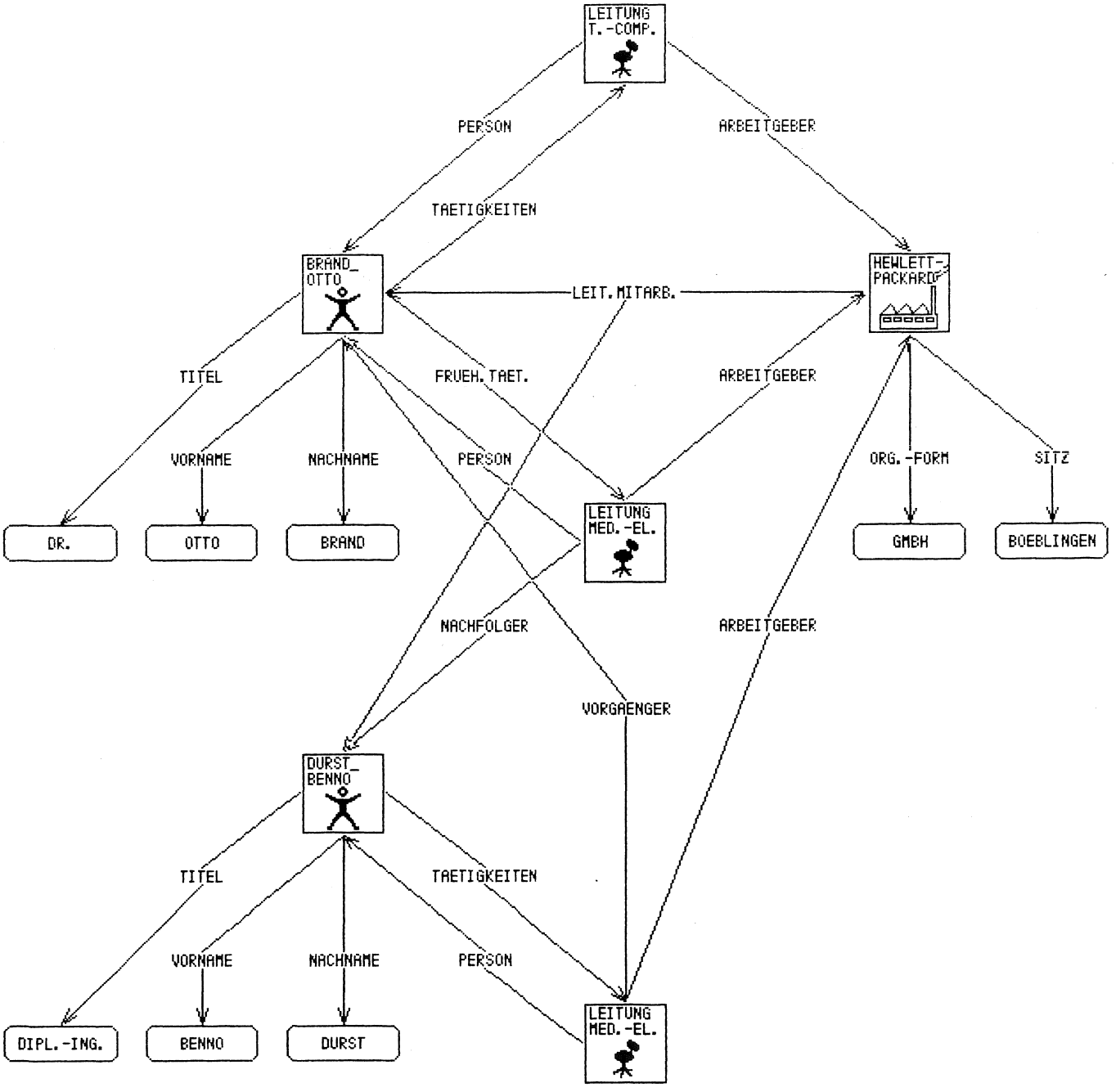


Abbildung 4-6: Zustand der Faktenwissensbasis nach der Verarbeitung

Wenn nun der Sachbearbeiter mittels eines Navigationsskripts die Informationen über den Vorgänger Otto Brand abruft, erkennt er, daß die bisherige Tätigkeit bereits richtig eingeordnet ist, und muß nur noch die neue Tätigkeit des Managers im Werk für Tischcomputer eingeben. Dann ist der Inhalt des Artikels verarbeitet. Das entstandene Wissensnetz ist in der Abbildung 4-6 dargestellt.

Diese Art des Systems, Schlüsse zu ziehen oder *Inferenzen* zu bilden, ist notwendig für eine sinnvolle Wissensverarbeitung. Einmal erleichtert sie die Aufgabe des Wissenserwerbs an sich. Zum anderen enthält die Wissensbasis durch Mehrfachvernetzung von Objekten gewollte Redundanzen, die konsistent erhalten werden müssen.

4.4 Die Wissenskomponente des Systems D&I

Die Wissenskomponente des Systems D&I ist objektorientiert mit Hilfe der Sprache ObjTalk aufgebaut. Dies betrifft sowohl die Wissenseinheiten zur Repräsentation von Sachwissen als auch die Skripts zur Interpretation der Sachinformationen, die beide mit Hilfe von ObjTalk-Objekten dargestellt sind.

4.4.1 Wissensbasisobjekte

Als Wissenseinheiten zur Repräsentation von Sachwissen dienen im System D&I sogenannte Wissensbasisobjekte. Wie bereits im Abschnitt 4.3 angeführt, besitzt jedes derartige Wissensbasisobjekt einen Typ und einen oder mehrere Aspekte. Jedem Aspekt eines Objekts zugeordnet ist eine Menge von Merkmalen, die mit Merkmalswerten belegt sein können, sowie eine Menge von Skripts. Abbildung 4-7 zeigt dies am Beispiel des Wissensbasisobjekts Brand—Otto. Das Wissensbasisobjekt ist vom Typ PERSON und besitzt die drei Aspekte PERSON, PRIVAT und BERUF. Bezüglich der Aspekte PRIVAT und BERUF gibt es jeweils eine Reihe von Skripts und Merkmal-Wert-Paaren, der Aspekt PERSON ist demgegenüber nicht weiter spezifiziert.

Manche der Eigenschaften eines Wissensbasisobjekts sind individuell, d.h. sie haben für verschiedene Objekte unterschiedliche Ausprägungen, andere sind generisch, d.h. sie sind für alle Objekte eines Typs gemeinsam. Individuelle Eigenschaften müssen dem Objekt selbst zugeordnet werden, die generischen Eigenschaften hingegen sind abhängig vom Typ des Objekts und können aus diesem abgeleitet werden. In Abbildung 4-7 sind die individuellen

```
Wissensbasisobjekt Brand_Otto
.....................................................................
Typ       PERSON

Aspekte   PERSON

          PRIVAT    Merkmale  Nachname:                 Brand
                              Vorname:                  Otto
                              Geschlecht:               männlich
                              Titel:                    Dr.
                              Geburtsdatum:             (undefiniert)
                              Geburtsort:               (undefiniert)
                    Skripts   Berufliche Daten

          BERUF     Merkmale  Tätigkeiten:              Ltg. Tischcomp.
                              Früh. Tätigkeiten:        Ltg. Med.-El.
                              Berufsausbildung:         (undefiniert)
                    Skripts   Beginn einer Tätigkeit
                              Beenden einer Tätigkeit
                              Tätigkeitswechsel
                              Beginn einer Tätigkeit
                              wird abgelöst
                              löst ab
                              Nähere Angaben zu Tätigkeit
```

Abbildung 4-7: Eigenschaften des Wissensbasisobjekts Brand—Otto

Eigenschaften[27] von Brand—Otto in *Schrägschrift* dargestellt, alle anderen Eigenschaften sind generischer Art und treffen für jedes Objekt vom Typ PERSON zu. Was man am Beispiel des Objekts Brand—Otto erkennt, gilt durchweg für die D&I Faktenwissensbasis: Die einzigen individuellen Eigenschaften eines Wissensbasisobjekts sind die Werte seiner Merkmale. Alle anderen Eigenschaften eines Wissensbasisobjekts, wie Aspekte, Namen und Eigenheiten der Merkmale sowie die einem Objekt zugeordneten Skripts sind generischer Natur und können aus dem Typ des Objekts abgeleitet werden. Für die Darstellung der Wissensbasisobjekte mit Hilfe der Sprache ObjTalk hat sich daher folgende Vorgehensweise als sinnvoll erwiesen:

Wissensbasisobjekte werden durch ObjTalk-Objekte dargestellt. Die individuellen Eigenschaften von ObjTalk-Objekten sind gegeben durch die Filler ihrer Slots. Die *Merkmale* der Wissensbasisobjekte werden deshalb mit Hilfe von

[27] Genau gesagt, ihre Ausprägungen sind in Schrägschrift dargestellt.

ObjTalk-Slots dargestellt, jeder Merkmalswert entspricht einem Filler eines Slots. Wann immer wir im folgenden ein (Wissensbasis-)Objekt betrachten, werden wir es mit dem ObjTalk-Objekt gleichsetzen, durch das es repräsentiert ist. Genauso werden wir die Merkmale eines Objekts mit den ObjTalk-Slots identifizieren, durch die sie repräsentiert sind.

Typen und *Aspekte* werden durch ObjTalk-Klassen dargestellt: Der Typ eines Wissensbasisobjekts wird repräsentiert durch die Klasse des zugehörigen ObjTalk-Objekts. Die Menge aller Klassen in der Vererbungshierarchie oberhalb dieser Klasse dient dazu, die Aspekte des Wissensbasisobjekts zu repräsentieren. Wir werden die Typen und Aspekte des Systems D&I mit den Klassen identifizieren, durch die sie repräsentiert sind. Die in der Grundversion des Systems D&I vorhandenen Typen und Aspekte sind in Abbildung 4-8 dargestellt.

Die generischen Eigenschaften von Merkmalen sind durch *Merkmalsbeschreibungen* repräsentiert, die den Aspekten zugeordnet sind. Merkmalsbeschreibungen sind ObjTalk-Slotbeschreibungen, die um zusätzliche Sprachelemente erweitert sind. Demzufolge sind Merkmale nichts anderes als Slots mit zusätzlichen generischen Eigenschaften. Auf Merkmale wird in Abschnitt 4.4.2 noch genauer eingegangen.

Skripts sind ebenfalls den Aspekten zugeordnet und durch Beschreibungen repräsentiert. Derartige Skriptbeschreibungen haben Ähnlichkeiten, mit den Beschreibungen von ObjTalk-Methoden; sie können aber nicht mit diesen gleichgesetzt werden, da sie reichhaltigere Strukturen darstellen. Im System D&I existieren Skripts und Methoden nebeneinander. Skripts werden in Abschnitt 4.4.3 noch detaillierter beschrieben.

Das Wissensbasisobjekt Brand—Otto mit seinen individuellen und generischen Eigenschaften läßt sich mit Hilfe von ObjTalk dementsprechend folgendermaßen darstellen (Abbildung 4-9): Brand—Otto ist eine Instanz der ObjTalk-Klasse PERSON. PERSON als unteres Ende eines Teilbaums in der ObjTalk-Klassenhierarchie repräsentiert den Typ des Wissensbasisobjekts. Zugleich zählt PERSON zusammen mit seinen Superklassen BERUF und PRIVAT zu den Aspekten des Typs PERSON. Den Aspekten BERUF und PRIVAT sind Skriptbeschreibungen und Merkmalsbeschreibungen zugeordnet. Für jedes durch eine derartige Merkmalsbeschreibung definierte Merkmal kann das Objekt Brand—Otto einen Merkmalswert aufweisen.

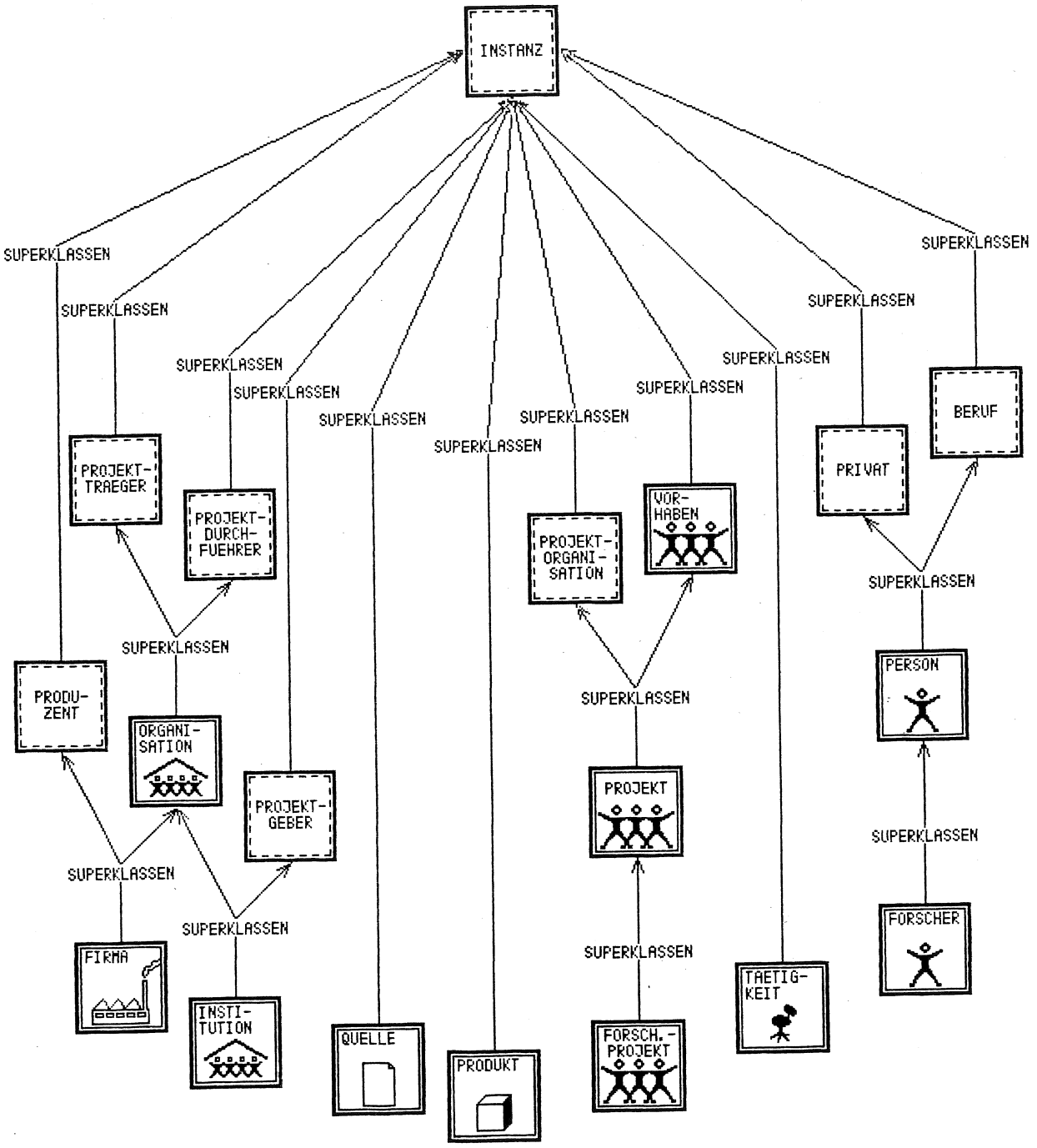


Abbildung 4-8: Hierarchie der Typen und Aspekte des Systems D&I

Die Piktogramme mit einbeschriebenen graphischen Symbolen stellen Typen dar, die übrigen Piktogramme stehen für Aspekte.

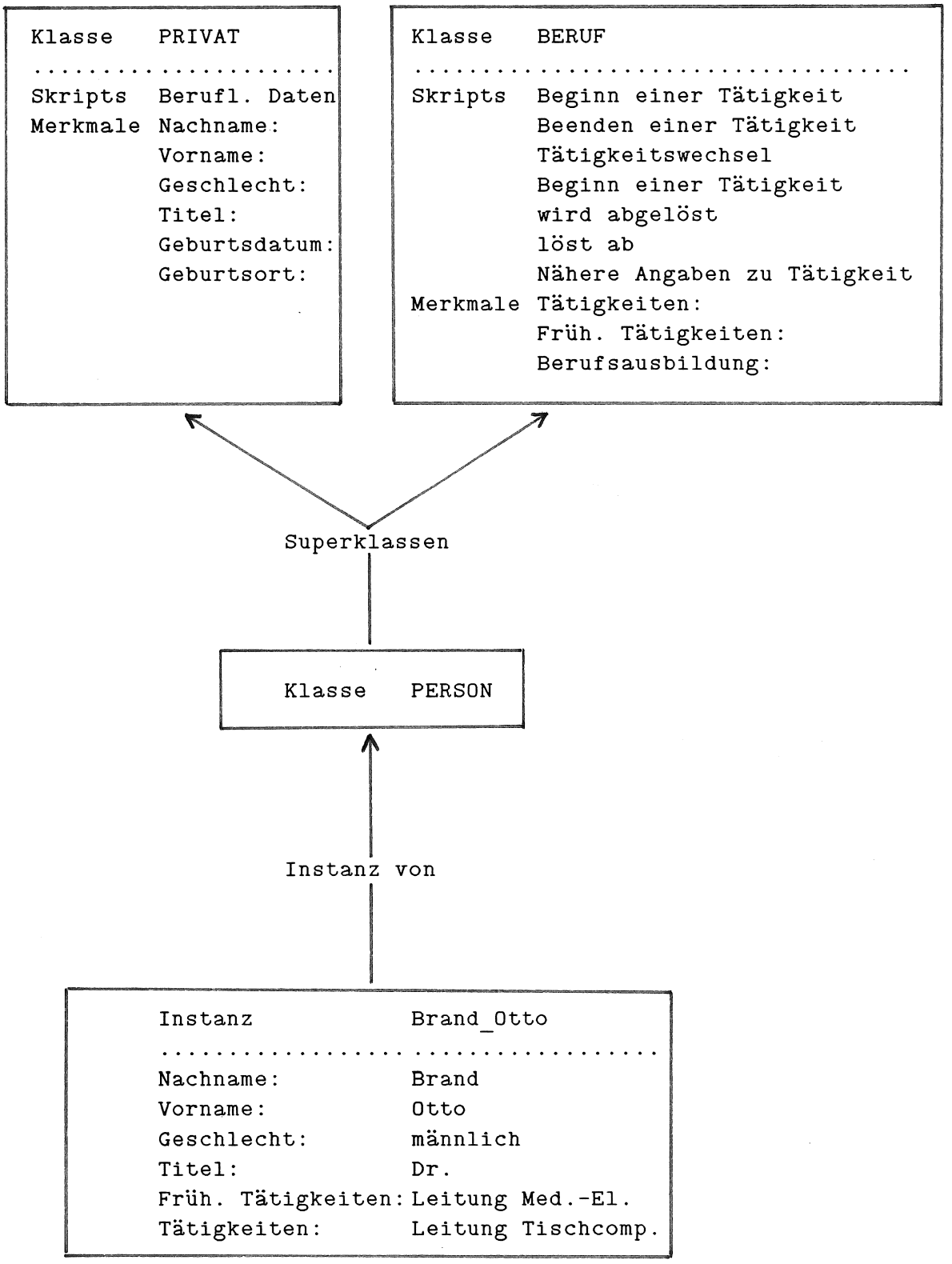


Abbildung 4-9: ObjTalk-Repräsentation der Person Brand—Otto

4.4.2 Merkmale

Wenn wir die Rolle von Merkmalen im System D&I untersuchen, müssen wir unterscheiden zwischen dem Wert eines Merkmals, der von Instanz zu Instanz variieren kann, und den generischen Eigenschaften eines Merkmals, die unabhängig vom konkreten Objekt sind und in den Aspekten beschrieben sind. Abbildung 4-10 zeigt beispielhaft die generischen Eigenschaften des Merkmals "Frühere Tätigkeiten:".

Die generischen Eigenschaften von Merkmalen haben zum einen Auswirkungen auf die Benutzerschnittstelle des Systems; sie bestimmen die Art der *Präsentation* von Merkmalen und Merkmalswerten. Zum anderen leiten sich aus den generischen Eigenschaften von Merkmalen *Inferenzmechanismen* ab, wie beispielsweise in Abschnitt 4.3.6 beschrieben.

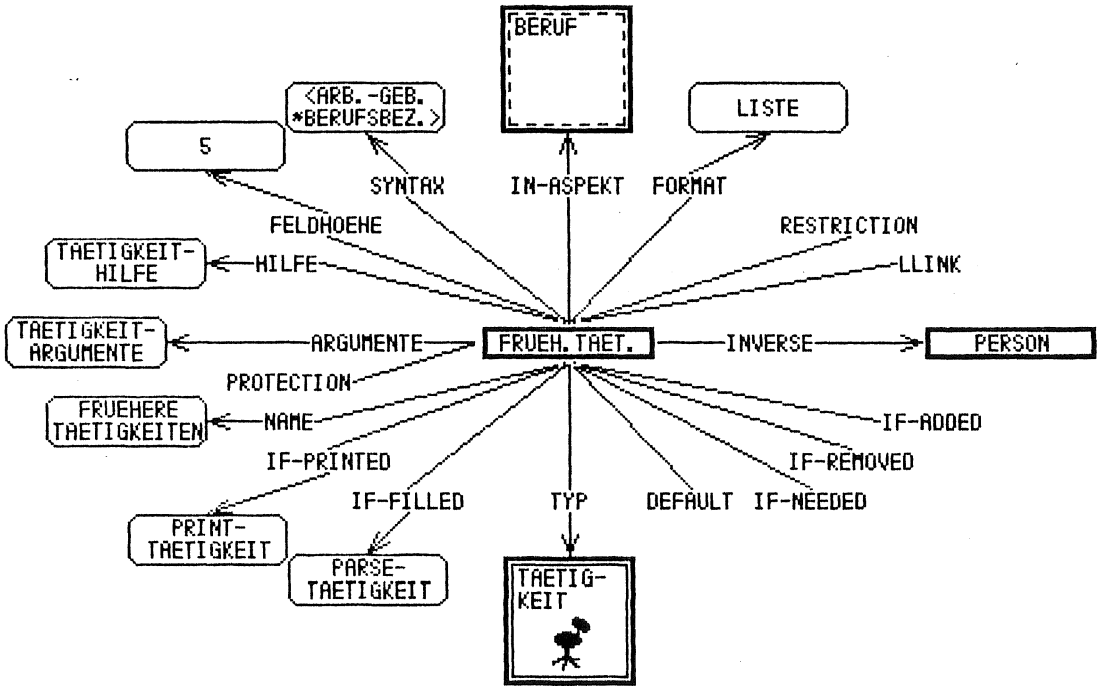


Abbildung 4-10: Generische Eigenschaften des Merkmals "Frühere Tätigkeiten:"

Präsentationseigenschaften

Folgende Gesichtspunkte der Präsentation sind bestimmt durch generische Eigenschaften von Merkmalen:

- Der *Name* eines Formularfelds leitet sich ab vom dargestellten Merkmal, im Normalfall ist er mit dem Namen des Merkmals identisch.
- Die erforderliche *Größe* des Formularfelds ist abhängig vom darzustellenden Merkmal.
- Merkmalswerte können für den Benutzer *schreibgeschützt* oder *lesegeschützt* sein; im ersten Fall kann das betreffende Formularfeld nicht editiert werden, im zweiten Fall wird das Merkmal überhaupt nicht in einem Formularfeld sichtbar.
- Merkmalswerte können sich dem Benutzer unterschiedlich von der internen Repräsentation anbieten und können in einer vom internen Format abweichenden Darstellung eingegeben werden. Die verwendeten *Konvertierungsfunktionen* sind abhängig vom jeweiligen Merkmal.
- Es gibt *Restriktionen* für die Werte von Merkmalen, die beim Eintrag in die Formularfelder erfüllt sein müssen. Manche Einträge müssen ein bestimmtes Prädikat erfüllen. Manchmal ist nur ein einzelner Textstring als Eintrag erlaubt oder es wird eine Liste von Namen erwartet.
- Bestimmte Merkmale dürfen nur Objekte von einem bestimmten Typ zum Wert haben. Beim Eintrag eines Namens in ein entsprechendes Formularfeld wird ein *Identifikationsvorgang* angestoßen: d.h. ein Objekt wird gesucht, auf das der Name paßt, falls das nicht möglich ist, wird ein neues Objekt erzeugt.
- Merkmale können einen *Beschreibungstext* besitzen, der als Hilfeleistung für den Benutzer ausgegeben werden kann.
- Merkmale können ein *Syntaxmuster* besitzen, das als Hilfe beim Ausfüllen eines leeren Formularfelds dienen kann.
- Merkmale können eine Berechnungsvorschrift für eine Auswahl möglicher *Eintragungsvorschläge* besitzen. Aus diesen Vorschlägen kann zum Zweck der Hilfeleistung ein Menü generiert werden.

Für die Benutzerschnittstelle sieht das System D&I besondere Zugriffsmethoden[28] auf Merkmalswerte vor. Diese Zugriffsmethoden berücksichtigen eine Reihe der genannten Präsentationseigenschaften der Merkmale, nämlich den Lese- und den Schreibschutz, die Konvertierungsfunktionen, die Restriktionen für Merkmalswerte und die Identifikation von Objekten. Die übrigen Präsentationseigenschaften der Merkmale liegen in Form von Beschreibungen vor, die von der Benutzerschnittstelle des Systems abgerufen und in geeigneter Form interpretiert werden können.

---

[28]Diese Zugriffsmethoden sind auf Seite 82 beschrieben

Inferenzmechanismen

Bei den Inferenzmechanismen, die den Merkmalen zugeordnet sind, handelt es sich durchweg um Triggerfunktionen. Das sind Prozeduren, die ausgeführt werden, wenn auf ein Merkmal zugegriffen wird. Im wesentlichen gibt es drei Arten von Zugriffen, die derart überwacht werden:

- *Die Initialisierung:* Wenn ein Wissensbasisobjekt erzeugt wird, werden seine Merkmale initialisiert. Eine Triggerfunktion kann hierbei *Defaults* (Voreinstellungen) für bestimmte Merkmale vorsehen.
- *Das Besetzen von Merkmalen mit Merkmalswerten:* Wenn ein neuer Merkmalswert eingetragen wird, kann eine sogenannte If-added-Prozedur angestoßen werden, die weitere Eintragungen in der Wissensbasis vornehmen kann. Wenn ein Merkmalswert ausgetragen wird, kann entsprechend eine If-removed-Prozedur aufgerufen werden. Derartige Inferenzen werden auch als *Read-Time-Inferenzen*[29] bezeichnet.
- *Das Lesen von Merkmalen:* Wenn der Wert eines Merkmals angefordert wird, das noch undefiniert ist, kann eine If-needed-Prozedur aufgerufen werden, die die Berechnung des Wertes vornimmt. Hierfür ist auch der Name *Question-Time-Inferenz*[30] gebräuchlich.

Ein generelles Prinzip des Systems D&I besteht darin, daß inferierbares Wissen möglichst frühzeitig in die Wissensbasis eingetragen wird. Es werden also im System D&I sehr häufig Read-Time-Inferenzen vorgenommen. Der Grund hierfür liegt darin, daß Question-Time-Inferenzen in objektorientierten Systemen oft eine aufwendige Suche erfordern. Zum Zeitpunkt des Eintrags eines Merkmalswerts hingegen sind meist alle Objekte bekannt, die von der Modifikation betroffen sind.

Read-Time-Inferenzen werden im System D&I dazu genutzt, um konsistente Konfigurationen von Objekten zu gewährleisten. Ein Beispiel hierfür ist die X-Link-Konfiguration zweier Objekte, wie sie in Abbildung 4-11 dargestellt ist. Die graphische Darstellung der Verbindungen zwischen den beteiligten Objekten erinnert an den Buchstaben "X"; dies gab den Anlaß für die Bezeichnung X-Link. X-Link-Konfigurationen treten immer auf, wenn zwei Merkmale von Objekten zueinander inverse Relationen bezeichnen. Im abgebildeten Beispiel ist das Merkmal "Taetigkeiten" einer Person invers zum Merkmal "Person" einer Tätigkeit. Die Werte der zweier korrespondierender inverser Merkmale bedin-

---

[29] Read-Time bezeichnet den Zeitpunkt, zu dem eine Information erstmalig anfällt.

[30] Question-Time bezeichnet den Zeitpunkt, zu dem eine Information gebraucht wird.

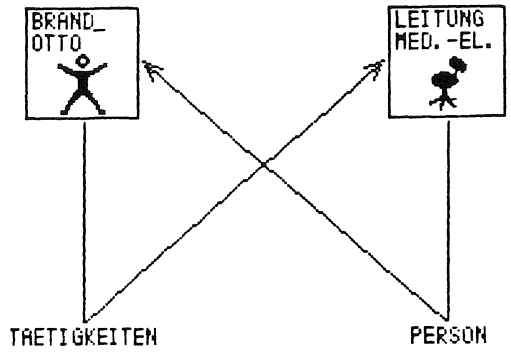


Abbildung 4-11: Eine X-Link-Konfiguration zweier Objekte

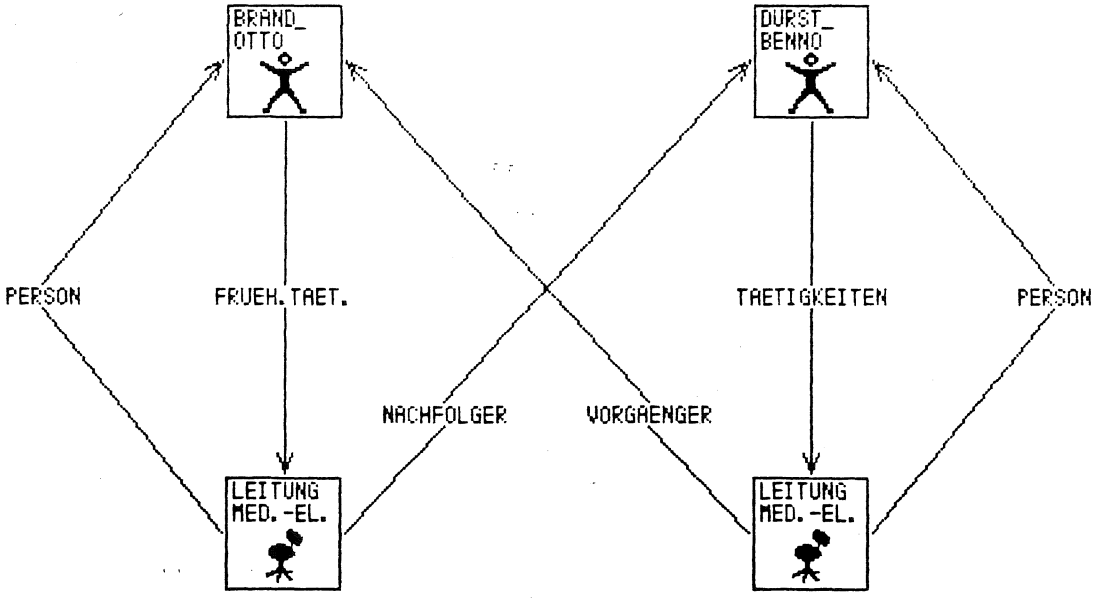


Abbildung 4-12: Eine L-Link-Konfiguration

gen sich gegenseitig. Das heißt, wann immer ein Merkmalswert gesetzt wird, muß der Wert des entsprechenden inversen Merkmals ebenfalls aktualisiert werden. Wann immer ein derartiger Merkmalswert gelöscht wird, muß sein Partner ebenfalls entfernt werden.

Eine Verallgemeinerung des X-Link ist der sogenannte L-Link. L-Link-Konfigurationen treten auf, wenn zwei Merkmale m_1 und m_2 von Objekten in einem direkten Abhängigkeitsverhältnis stehen. Es muß eine Abbildungsvorschrift geben, die jeder Wertebelegung w_1 des Merkmals m_1 an einem Objekt o_1, für welches das Merkmal definiert ist, eine davon abhängige Wertebelegung w_2 des Merkmals m_2 an einem anderen Objekt o_2 zuordnet. Wann immer das Merkmal m_1 eines Objekts o_1 gesetzt wird, muß auch die abhängige Wertebelegung w_2 vorgenommen werden. Wann immer eine derartige Wertebelegung zurückgenommen wird, muß auch die davon abhängige Wertebelegung rückgängig gemacht werden. Abbildung 4-12 gibt ein Beispiel für eine L-Link-Konfiguration von Objekten. In dem Moment, in dem Brand—Otto als Vorgänger in einer Tätigkeit von Durst—Benno feststeht, steht dieser als Nachfolger des erstgenannten in der entsprechenden Tätigkeit fest.

4.4.3 Skripts

Skripts dienen zur Interpretation von Sachinformationen und deren Einbettung in die Wissensbasis des Systems D&I. Die Aufgabe von Skripts ist es, stereotype Vorgänge zu repräsentieren, diese Eigenschaft haben sie mit den Skripts im Sinne von [Schank, Abelson 77] gemeinsam. Realisiert sind Skripts im System D&I in Form explizit repräsentierter Methoden, die für die interaktive Benutzung vorgesehen sind.

Es lassen sich zwei Arten von stereotypen Vorgängen unterscheiden, die mit Hilfe von Skripts dargestellt werden.

- *Stereotypische Vorgänge aus der darzustellenden Wissenswelt*, wie etwa ein Tätigkeitswechsel einer Person, der Konkurs einer Firma oder die Gründung eines Projekts. Die Aktivierung eines derartigen Skripts bewirkt eine entsprechende Erweiterung und Veränderung der Wissensbasis. Derartige Skripts werden auch als Interpretationsskripts bezeichnet.
- *Stereotypische Vorgänge der Systembenutzung*, wie etwa die Navigation zu einem anderen Aspekt des aktuellen Objekts, oder die Korrektur einer fehlerhaften Eintragung in die Wissensbasis. Die Aktivierung eines derartigen Skripts geschieht aus technischen Gründen und nicht auf Grund einer Veränderung der abzubildenden Wissenswelt.

Beide Arten von Vorgängen sind mit Hilfe der Skripts dargestellt. Skripts sind den Aspekten von Wissensbasisobjekten zugeordnet. Skripts sind gekennzeichnet durch folgende Merkmale (siehe auch Abb. 4-13):

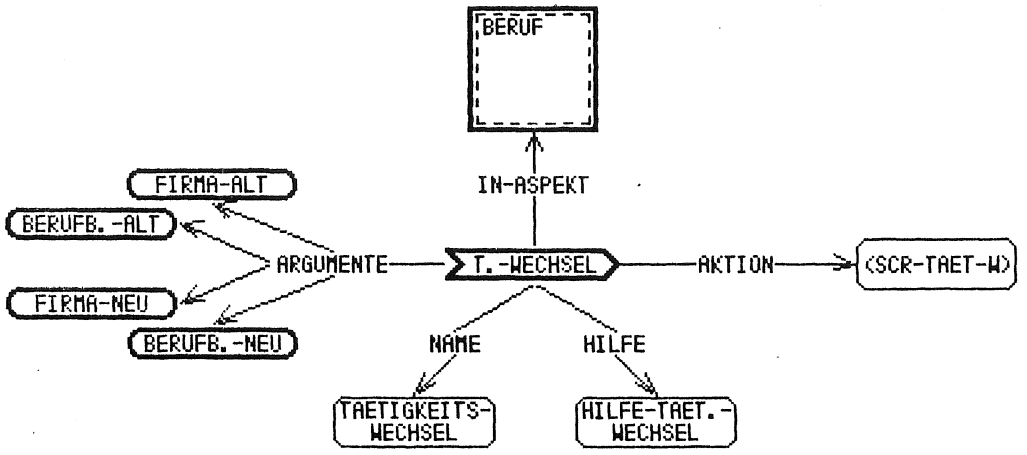


Abbildung 4-13: Das Skript *Tätigkeitswechsel* und seine Eigenschaften

- Jedes Skript ist einem *Aspekt* zugeordnet, in dessen Skriptmenu es aufgeführt ist.
- Jedes Skript besitzt einen *Namen*, mit dem das Skript im Skriptmenu erscheint.
- Skripts besitzen einen *Erklärungstext*, der zum Zweck der Hilfeleistung vom System abgerufen werden kann.
- Skripts besitzen eine Liste von formalen *Argumenten*, die beim Aufruf des Skripts vom Benutzer erfragt werden.
- Skripts besitzen eine *Aktion*, das ist eine Reihe von Anweisungen, die ausgeführt werden, nachdem alle Argumente eingegeben worden sind. Diese Aktion bewirkt gegebenenfalls eine Veränderung der Wissensbasis bzw. eine Navigation zu einem neuen Aspekt.

Die formalen Argumente eines Skripts sind durch ähnliche Eigenschaften gekennzeichnet wie die Merkmale eines Objekts: Sie besitzen einen Namen, sie können eine Konvertierungsfunktion und Restriktionen besitzen und das Eingeben eines Werts kann einen Identifikationsvorgang auslösen. Auch die Argumente besitzen einen Erklärungstext, ein Syntaxmuster und eine Berechnungsvorschrift für Eingabevorschläge. Außerdem besitzen Argumente ein Prompt, das ist eine Eingabeaufforderung, die bei der Abfrage des Arguments im Skriptdialog auf dem Bildschirm erscheint. Ein Beispiel für ein Skriptargument zeigt Abbildung 4-14.

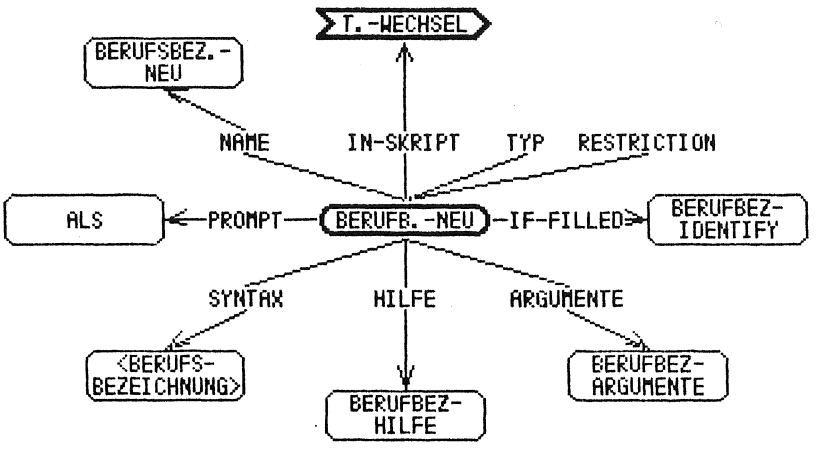


Abbildung 4-14: Das Skriptargument *Berufsbezeichnung-neu*

4.4.4 generische Operationen

Im System D&I gibt es eine Reihe von generischen Operationen auf Wissensbasisobjekten. Darunter sollen Operationen verstanden werden, die auf alle Wissensbasisobjekte angewandt werden können, aber für Objekte von unterschiedlichem Typ unterschiedliche Wirkungen haben können. Diese generischen Operationen werden durch Versenden von Botschaften aktiviert. Die Reaktion auf diese Botschaften ist dann abhängig davon, welche Methoden das Empfängerobjekt vorsieht. Wenn beim Empfängerobjekt keine passende Methode definiert ist, wird höher in der Vererbungshierarchie gesucht. Für alle generischen Operationen ist deshalb an einem weit oben in der Vererbungshierarchie gelegenen Knoten eine Methode definiert, die das standardmäßige Verhalten der Wissensbasisobjekte bestimmt.[31]

Identifikation von Objekten

Die wichtigste unter den generischen Methoden ist die sogenannte *Identifikationsmethode*, die zum Identifizieren von Objekten dient. Diese Methode wird typischerweise beim Ausfüllen eines Formularfelds oder bei der Eingabe eines Skriptarguments aktiviert, wenn dort der Name eines Objekts von einem bestimmten Typ verlangt wird. Die Methode bringt als Ergebnis entweder das Objekt zurück, das durch den eingegebenen Namen bezeichnet wird, oder es wird eine Fehlermeldung zurückgegeben, wenn es nicht möglich war, das Objekt zu identifizieren.

[31] Wie wir in Kapitel 5 sehen werden, handelt es sich dabei um die Klassen TYP und INSTANZ.

Wenn an einer Stelle ein Objekt vom Typ *typ* erwartet wird und der Benutzer den Namen *objektname* eingibt, so wird folgende Botschaft versandt:

(**ask** *typ* **identify:** *objektname*)

Diese Botschaft aktiviert die für die Klasse *typ* definierte Identifikationsmethode. Daraufhin läuft ein Identifikationsprozeß ab, wobei abhängig von der Klasse *typ* die im folgenden beschriebenen Vorgänge allesamt oder zumindest teilweise stattfinden:

1. Der Name *objektname* wird normalisiert, z.B. auf eine standardisierte Groß/Kleinschreibung gebracht.
2. Es wird nach einem Objekt mit dem Namen *objektname* gesucht.
3. Es wird nach Objekten mit dem Synonymnamen *objektname* gesucht. Synonymbeziehungen sind im System D&I durch Objekte vom Typ SYNONYM repräsentiert.
4. Es wird nach Objekten mit ähnlichen Namen gesucht. Bei der Suche nach einer Person werden beispielsweise alle Personen mit gleichem Nachnamen in Betracht gezogen.
5. Für jedes gefundene Objekt wird untersucht, ob sein Typ mit dem verlangten Typ verträglich ist. Dies ist genau dann der Fall, wenn beide Typen in einer Vererbungslinie untereinander liegen. Beispielsweise wenn ein Objekt vom Typ PROJEKT verlangt wird, aber ein Objekt vom Typ FORSCHUNGSPROJEKT gefunden wurde, steht dies mit der Forderung im Einklang, da jedes Forschungsprojekt als Spezialisierung des Begriffs Projekt auch ein Projekt ist. Umgekehrt, wenn ein Forschungsprojekt gefordert wird, aber lediglich ein Projekt gefunden wurde, kann man annehmen, daß das richtige Objekt gefunden wurde, es aber aufgrund geringen Wissens ursprünglich zu allgemein eingeordnet worden war.
6. Die Namen der gefundenen Objekte und der ursprünglich eingegebene Name *objektname* werden in einer Liste zusammengefaßt. Wenn diese Liste mehr als ein Element enthält, wird der Benutzer aufgefordert, einen Namen auszusuchen.
7. Wenn schließlich ein einziger Name übrig ist, gibt es nur noch drei Möglichkeiten:

   a. Es gibt ein Objekt mit diesem Namen und dieses besitzt den Typ *typ* oder einen spezielleren Typ in der Klassenhierarchie: Das gesuchte Objekt ist gefunden.

   b. Es gibt ein Objekt mit diesem Namen, der Typ des Objektes ist aber allgemeiner als der geforderte Typ *typ*. Beispiel: Es ist ein Forschungsprojekt gesucht worden, gefunden wurde ein Projekt. Folge: Das Objekt muß "verfeinert" werden, d.h. es wird von nun an dem spezielleren Typ *typ* zugeordnet und bekommt dadurch mehr Eigenschaften.

   c. Es gibt kein Objekt mit diesem Namen. Dann wird ein Objekt vom Typ *typ* mit diesem Namen erzeugt.

Im Zusammenhang mit dem Identifikationsprozeß werden weitere Botschaften an Typen verschickt, welche die folgenden Methoden aktivieren können:

- Die Methode **find**: wird benötigt, um Objekte mit synonymen oder ähnlichen Namen zu finden, wobei bereits die Verträglichkeit mit dem verlangten Typ geprüft wird.
- Die Methode **refine**: dient dazu, um Instanzen zu verfeinern, das heißt, sie einer spezielleren Klasse in der Spezialisierungshierarchie zuzuordnen.
- Die Methode **build**: dient dazu, um neue Instanzen des betreffenden Typs zu erzeugen.

Bereitstellung von Hilfeinformation

Zwei weitere Methoden bieten Hilfeinformation an, die von der Benutzerschnittstelle verwertet wird:

- Die Methode **syntax**: gibt einen Muster zurück, das bei der Eingabe des Namens eines Objekts vom betreffenden Typ eingehalten werden soll.
- Die Methode **help**: gibt einen Erklärungstext über den betreffenden Typ zurück.

Zugriffe auf Merkmale

Für die Zugriffe auf Merkmale dienen die von ObjTalk vordefinierten Botschaften, die aber im System D&I besondere Auswirkungen besitzen:

- Mit der Botschaft (**ask** *objekt merkmal*)
  werden Merkmalswerte abgerufen. Diese Botschaft kann die im Abschnitt 4.4.2 beschriebenen Question-Time-Inferenzen auslösen.
- Mit der Botschaft (**ask** *objekt merkmal = wert*)
  werden Merkmalswerte eingetragen. Diese Botschaft kann die im Abschnitt 4.4.2 beschriebenen Read-Time-Inferenzen auslösen.

Für Zugriffe auf Merkmale von Seiten der Benutzerschnittstelle sind die folgenden zwei Arten von Zugriffen vorgesehen.

- Die Botschaft (**ask** *objekt merkmal* **print:**)
  ruft ein Merkmal ab und bereitet es zur externen Darstellung auf, indem eventuell eine Konvertierungsfunktion aufgerufen wird.
- Die Botschaft (**ask** *objekt merkmal* **fill:** *wert*)
  bewirkt die Überprüfung und den Eintrag des Merkmalswerts *wert*. Gegebenenfalls wird der Wert über eine Konvertierungsfunktion verändert und identifiziert. Schließlich wird der Merkmalswert beim Objekt eingetragen.

5. Ein Metasystem zur Modellierung von konzeptuellem Wissen

5.1 Metasysteme

Computer werden immer häufiger in Bereichen eingesetzt, die sich schwer strukturieren lassen, beispielsweise in der Bürowelt, bei Planungsaufgaben, bei Designproblemen, in der Diagnose oder in der Konstruktion. Zwei Merkmale sind allen diesen Arbeitsbereichen gemeinsam: Es fehlt die formale Beschreibung des Problemraums, und es gibt keine wohldefinierten Handlungsziele. Daher ist es schwierig oder gar unmöglich, eindeutig definierte Lösungsmethoden für die Problemstellungen aus diesen Arbeitsgebieten zu finden.

Bei der Anwendung von Computersoftware in diesen Gebieten hat der Benutzer des Systems deshalb große Schwierigkeiten zu überwinden:

- Da die Zielkriterien und Lösungswege nicht von vornherein eindeutig sind, fällt es einem Menschen schwer, die vom Computer gefundenen Lösungen zu akzeptieren. Der Benutzer weiß nicht, welches konzeptuelle Modell einer Computerentscheidung zugrundeliegt. Die Dokumentation des Programmes hilft häufig nicht weiter, da dort meist nur die Softwarestruktur des Programmes, aber nicht die zugrundeliegenden Konzepte des Anwendungsgebiets beschrieben sind. Außerdem gibt es eine Reihe von Phänomenen dynamischer Natur, die nicht alle in einer statischen Beschreibung vorweggenommen werden können. Oft liegt die einzige Erklärung für das Verhalten eines Softwaresystems im Programmcode und den aktuellen Werten der Daten.
- Da das Anwendungsgebiet nicht ausreichend formalisiert ist, ist die erstellte Software prinzipiell unvollständig. Der Systemersteller kann nur Heuristiken folgen, die sich erst noch in der Praxis bewähren müssen. Beim Einsatz des Systems werden dann dem Benutzer neue Kriterien offenbar, die eine Modifikation der Software erfordern, die wiederum durch den Softwarespezialisten vorgenommen werden muß. Dadurch entsteht ein fortwährendes Kommunikationsproblem zwischen dem Benutzer und dem Programmierer der Software. Für viele praktische Anwendungen dauert ein solcher Revisionszyklus zu lange. Daher haben viele Benutzer den Wunsch, fällige Änderungen selbst vornehmen zu können.

Bei der Beurteilung von komplexen Softwaresystemen gewinnen also zwei Kriterien wachsende Bedeutung: die *Durchschaubarkeit* und die *Adaptierbarkeit* des Systems durch seinen Benutzer. Diese Kriterien sind jedoch beim Einsatz von herkömmlich aufgebauten Systemen in schwer strukturierbaren Problemräumen nicht erfüllt. Bisher blieb dem Systembenutzer nichts anderes übrig, als Kenntnisse in der Computerprogrammierung zu erwerben, um das Verhalten der Programme verstehen und es selbständig verändern zu können. Dies ist

aber nicht der richtige Weg, da sich der Systembenutzer auf die Probleme des Anwendungsgebiets konzentrieren will und sich mit der Problematik der Softwareerstellung eigentlich nicht beschäftigen möchte.

Benötigt werden daher neue Softwarekonzepte und neue Systemkomponenten, die besser auf die Sichtweise der Benutzer abgestimmt sind als die herkömmlichen Programmkonstrukte und Programmierwerkzeuge. Einen Weg hin zu diesem Ziel eröffnen wissensbasierte Systeme. Das Verhalten dieser Systeme ist nicht in erster Linie durch Programme im herkömmlichen Sinn bestimmt, sondern vielmehr durch ihre *Wissensbasis*. Technisch gesehen ist das Wissen in einer Wissensbasis eine Ansammlung von Softwareobjekten. Es ist daher möglich, für wissensbasierte Systeme Komponenten zu entwerfen, die diese Softwareobjekte selbst zum Gegenstand haben. Solche Systemkomponenten stehen über dem eigentlichen Anwendungssystem und werden als *Metasysteme* bezeichnet. Metasysteme können das Anwendungssystem kontrollieren, da sie Zugriff auf das Wissen haben, welches das Verhalten des Anwendungssystems steuert. Metasysteme sind zwar nicht notwendig mit dem Vorhandensein einer Wissensbasis verknüpft, sie lassen sich aber in einer wissensbasierten Umgebung am besten realisieren. Die ersten wissensbasierten Metasysteme wurden in der Universität Stanford entwickelt: Typische Beispiele sind die Systeme EMYCIN [Shortliffe 76], TEIRESIAS [Davis, Lenat 82] und Meta-DENDRAL [Buchanan 78], bei denen es sich um Metakomponenten zu den bekannten Expertensystemen MYCIN und DENDRAL handelt.

Mit Hilfe geeigneter Mensch-Computer-Schnittstellen ermöglichen wissensbasierte Metasysteme das Untersuchen, Einbringen und Aktualisieren des Wissens, das einem Anwendungssystem zugrundeliegt:

- Die Konzepte des Anwendungsgebiets sind nicht mehr in den Prozeduren eines Anwendungsprogramms verborgen sondern explizit in Form von Einträgen in einer Wissensbasis repräsentiert. Daher lassen sich Metakomponenten zur *Untersuchung* von Softwaresystemen konstruieren, welche den Wissensbestand des Systems in textueller, tabellarischer oder graphischer Form benutzergerecht darstellen können.
- Eine zweite Anwendung von Metasystemen liegt im Aufbau und der Modifikation von Wissensbasen. Es ist nicht mehr erforderlich, Programme zu verstehen und zu modifizieren, um das Systemverhalten zu verändern. Das Anwendungssystem wird vielmehr dadurch modifiziert, daß der Benutzer neue Eintragungen in der Wissensbasis des Systems vornimmt. Ein Softwaresystem, das diesen Vorgang unterstützt, wird als Metasystem zum Wissenserwerb bezeichnet.

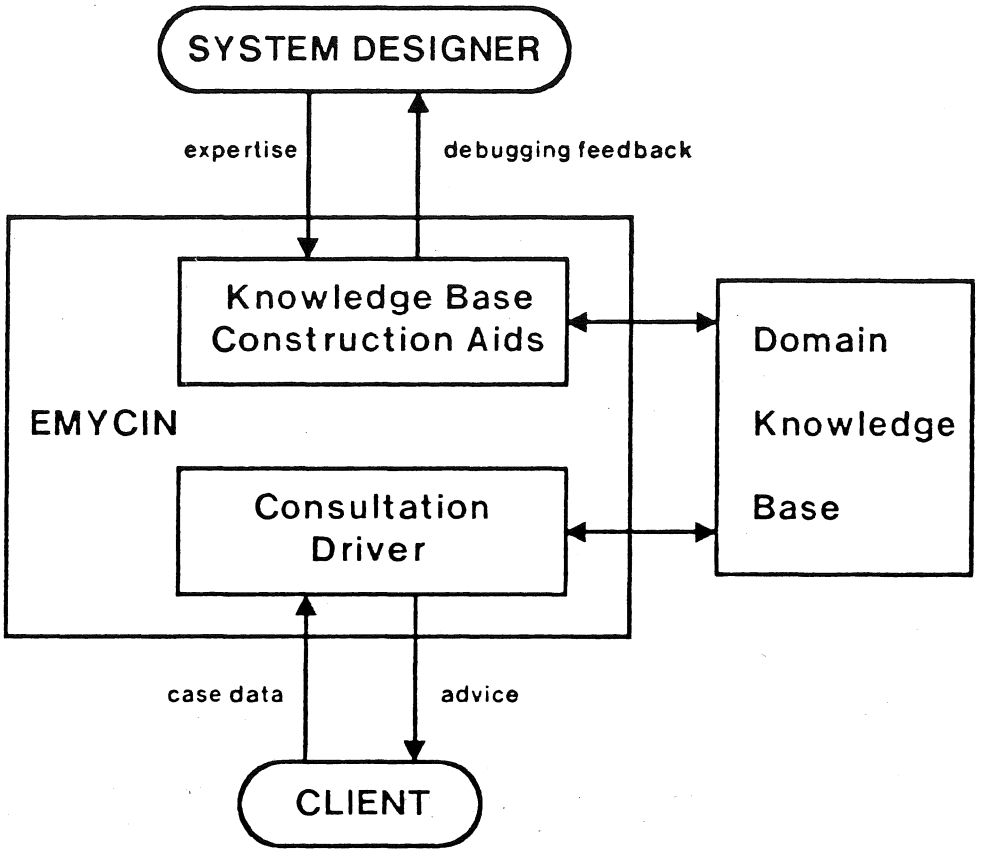


Abbildung 5-1: Das wissensbasierte System EMYCIN und seine Benutzung als Metasystem (oberer Teil der Abbildung) und Anwendungssystem (unterer Teil der Abbildung).

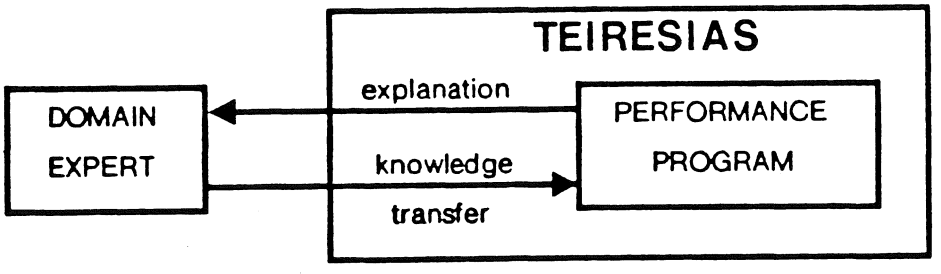


Abbildung 5-2: Das Metasystem TEIRESIAS ermöglicht den Wissenstransfer vom Gebietsexperten zum Anwendungssystem.

Metasysteme haben im Grunde eine ähnliche Funktion wie Software-Entwicklungswerkzeuge und lassen sich von diesen mitunter nur schwer unterscheiden. Dennoch gibt es eine Reihe von Kriterien, anhand deren sich die Besonderheiten von Metasystemen feststellen lassen:

- Metasysteme sind für den Anwendungsexperten, Software-Entwicklungswerkzeuge für den Software-Experten gedacht. Gegenüber Software-Entwicklungswerkzeugen, mit denen man im Prinzip alles machen kann, haben Metasysteme zugunsten einer besseren Verständlichkeit eine reduzierte Funktionalität.
- Metasysteme operieren zur Laufzeit des Anwendungssystems und modifizieren die zugrundeliegenden Konzepte unmittelbar mit sofortiger Wirkung; Software-Entwicklungswerkzeuge funktionieren meist offline auf Dateien, die erst noch übersetzt und geladen werden müssen.
- Metasysteme sind häufig mit dem Anwendungssystem integrierte Systemkomponenten; Software-Entwicklungswerkzeuge sind universell anwendbare, eigenständige Systeme.

Diese Kriterien treffen auf die in diesem Kapitel vorgestellte Metakomponente des Systems D&I zu. Diese Metakomponente ist in das D&I Anwendungssystem integriert und verwendet dieselbe Benutzerschnittstelle wie dieses System. Sie erlaubt die Erweiterung der Konzepte des D&I-Systems, ist aber an den Anwendungsfall der Dokumentation und Information gebunden und daher kein universelles Werkzeug. Der in Kapitel 6 vorgestellte Wissenseditor ZOO erfüllt die ersten beiden Kriterien, er ist jedoch nicht auf ein besonderes Anwendungssystem zugeschnitten. ZOO ist ein anwendungsneutrales Kernsystem, das sich mit den verschiedensten Anwendungen kombinieren läßt.

5.2 Das D&I Metasystem

Wesentliches Entwurfskriterium bei der Entwicklung des Systems D&I war es, ein System zu schaffen, das von seinem Benutzer verändert werden kann, ohne daß dieser mit den üblichen Softwareproduktionswerkzeugen wie Programmeditoren, Compiler, Binder usw. umzugehen braucht. Einem Organisationsexperten ohne besondere DV-Kenntnisse oder einem erfahrenen Sachbearbeiter muß es möglich sein, das System für eine Verarbeitung von Sachwissen aus anderen Fachgebieten umzurüsten. Das System muß imstande sein, seine eigene Abänderung durch den Benutzer zu unterstützen. Für diesen Zweck stellt das System eine Metakomponente bereit: das *D&I Metasystem.*

Die Arbeitsweise des D&I Anwendungssystems ist durch das im System D&I repräsentierte konzeptuelle Wissen bestimmt. Dies gilt in zweierlei Hinsicht: Zum einen bestimmt das konzeptuelle Wissen, welche Arten von Sachverhalten in der Wissensbasis dargestellt und konsistent gehalten werden können. Zum andern ergibt sich aus dem konzeptuellen Wissen die Funktionalität des Sy-

stems, die dem Benutzer bei der Verarbeitung von Informationen zur Verfügung steht.

Das D&I Metasystem ermöglicht den Zugang zu diesem konzeptuellen Wissen. Es stellt eine Benutzerschnittstelle zum Erwerb von konzeptuellem Wissen bereit. Mit Hilfe des Metasystems ist der Benutzer des Systems imstande, die Konzepte der Wissensbasis zu inspizieren, zu erzeugen und zu modifizieren. Das System kann so in seiner Funktionalität erweitert oder auch auf gänzlich neue Anwendungsgebiete umgestellt werden. Invariant dabei bleibt jedoch das Metawissen des Systems, das den grundsätzlichen Verwendungszweck des Systems, die Dokumentation von Fachinformationen bestimmt.

Das System D&I erlaubt also zwei Arten der Benutzung zum Zweck des Wissenserwerbs. Als Anwendungssystem dient es zur Assimilation von Ereignissen und Sachverhalten des Anwendungsgebiets. Als Metasystem unterstützt es die Akkommodation auf veränderte Einsatzgebiete.

5.2.1 Konzepte

Der Begriff "konzeptuelles Wissen" ist zunächst nur eine Wortschöpfung, die sich auf jedes Programmsystem anwenden läßt, um einen bestimmten Aspekt desselben noch gänzlich unabhängig von der Art seiner Implementierung benennen zu können. Die Funktionsweise jedes nichttrivialen Programms beruht auf dem konzeptuellen Wissen seines Programmierers, und die Realisierung des Programms ist eine Ausprägung dieses konzeptuellen Wissens.

Das dem System D&I zugrundeliegende konzeptuelle Wissen ist aber nicht etwa im Programmcode fest verdrahtet, vielmehr ist es ebenso wie das Sachwissen in Form von Objekten der Wissensbasis repräsentiert. Zum Zweck der Unterscheidung von den "gewöhnlichen" Objekten der Wissensbasis bezeichnen wir die Objekte, die das konzeptuelle Wissen repräsentieren, als *Konzepte*. Wir haben am Beispiel der Verarbeitung des Personenberichts in Abschnitt 4.3 bereits einige Konzepte kennengelernt, ohne daß wir sie schon als solche bezeichnet haben. Diese Konzepte lassen sich in fünf verschiedene Arten einteilen:

Typen: Jedes Objekt gehört einem *Typ* an: Die im Zeitungsartikel vorkommenden Typen heißen PERSON, TAETIGKEIT und FIRMA. Die Typen im System D&I sind eine besondere Art von ObjTalk-Klassen.

Aspekte: Den Typen von Objekten sind *Aspekte* zugeordnet: Beispielsweise gehören zu dem Typ PERSON die Aspekte BERUF und PRIVAT. Außerdem wird der Typ PERSON selbst auch als Aspekt angesehen.

Merkmale: Den Aspekten sind *Merkmale* zugeordnet: Beispielsweise zum Aspekt BERUF gehören die Merkmale "Tätigkeiten:", "Frühere Tätigkeiten:" und "Berufsausbildung:".

Skripts: Den Aspekten sind außerdem *Skripts* zugeordnet: Das Skript "Beginn einer Tätigkeit" gehört zum Aspekt BERUF.

Argumente: Skripts verfügen über *Argumente*: Das Skript "Beginn einer Tätigkeit" besitzt die Argumente "Berufsbezeichnung" und "Arbeitgeber".

Typen, Aspekte, Merkmale, Skripts und Skriptargumente sind also die grundlegenden Arten von Konzepten, mit deren Hilfe das gebietsspezifische Wissen des Systems D&I dargestellt ist. Gegenwärtig enthält die Wissensbasis des Systems D&I Konzepte aus der Wissenswelt der Computerindustrie, daher läßt sich das System in diesem Fachgebiet einsetzen. Ersetzt man die vorhandenen Konzepte mit Hilfe des Metasystems durch solche aus einem anderen Anwendungsgebiet, so ist das System D&I bereit, Zeitungsartikel mit dieser neuen Thematik zu verarbeiten.

5.2.2 Uniforme Benutzerschnittstelle

Konzepte sind zwar eine besondere Art von Objekten der Wissensbasis; dennoch sind sie im Grunde genauso aufgebaut wie alle anderen Objekte. Das heißt, sie können identifiziert und erzeugt werden und sie besitzen Merkmale, aus denen sich ihre charakteristischen Eigenschaften ergeben. Daher kann beim Erwerb von Konzepten dieselbe Benutzerschnittstelle Verwendung finden wie beim Erwerb von Sachwissen.

Der Bildschirmaufbau des Metasystems ist derselbe wie der des Anwendersystems (Abb. 5-3). Das Arbeitsfenster enthält ein Formular, das die Eigenschaften eines Konzeptes darstellt. Das Skriptfenster enthält ein Menü, welches auf die Aufgabe abgestimmt ist, konzeptuelles Wissen in ein Netz von Konzepten abzubilden und in diesem Netz zu navigieren.

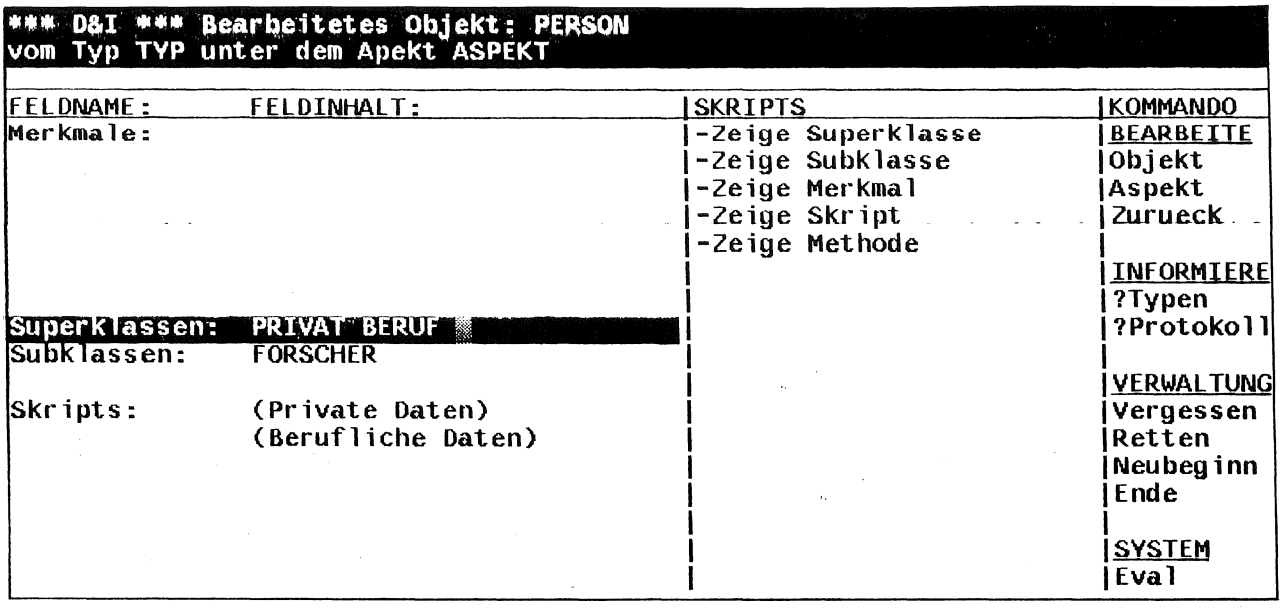


Abbildung 5-3: Benutzerschnittstelle des Metasystems

Auch im Metasystem ist die Auswahl der gezeigten Formularfelder im Arbeitsfenster und das Angebot der Funktionen im Skriptmenü abhängig vom aktuellen Aspekt. Die Anzahl der erforderlichen Aspekte zur Darstellung der verschiedenen Arten von Konzepten ist beschränkt, da es nur endlich viele Arten von Konzepten im System D&I gibt, nämlich im wesentlichen nur die oben genannten fünf Arten: Typen, Aspekte, Merkmale, Skripts und Argumente. Im Gegensatz zum Anwendersystem, das wegen der unbegrenzten Anzahl von Konzepten des Anwendungsgebiets prinzipiell unvollständig ist, wird es so möglich, die Funktionalität des Metasystems auf einen weitgehend vollständigen Standard zu bringen.

5.3 Modellierung von konzeptuellem Wissen mit dem D&I Metasystem

Welche Art von Sachwissen in der Wissensbasis des Systems D&I dargestellt werden kann, ist zunächst einmal davon abhängig, welche *Typen* von Objekten existieren. Weil in der Wissensbasis der Typ PERSON existiert, ist es möglich, die Eigenschaften einer Person, zum Beispiel die der Person Durst—Benno in der Wissensbasis darzustellen.

Im Typ PERSON ist also das Wissen darüber repräsentiert, welche Merkmale eine konkrete Person tragen kann. Ein solches Merkmal ist beispielsweise der "Vorname:" der Person; das Merkmal "Organisationsform:" hingegen läßt sich nicht auf Personen anwenden, es ist dem Typ FIRMA und nicht dem Typ PERSON zugeordnet. Der Typ eines Objektes ist also ausschlaggebend dafür, nach welchem *Schema* das Objekt aufgebaut ist.

5.3.1 Typen und Aspekte repräsentieren schematisches Wissen

Schemata sind häufig sehr umfangreich und komplex. Daher sind die Typen im System D&I keine elementaren Objekte, sondern sind mit Hilfe anderer Konzepte aufgebaut. Die Bausteine zur Konstruktion von Typen sind die *Aspekte*. Wenn wir also mit dem System D&I beispielsweise den Typ PERSON inspizieren (Abbildung 5-3), sehen wir im Arbeitsfenster unter der Benennung "Superklassen:" eine Auflistung der Aspekte, aus denen der Typ PERSON gebildet ist, in diesem Fall die beiden Namen PRIVAT und BERUF für die beiden Aspekte einer Person.

Wenn wir auch noch die Verwandtschaftsbeziehungen von Personen darstellen wollten (was das System derzeit nicht tut), müßten wir in der Liste einen weiteren Aspekt, sagen wir VERWANDTSCHAFT, eintragen. Das System würde versuchen, diesen Aspekt zu identifizieren, und da er noch nicht existiert, würde es einen solchen Aspekt in der Wissensbasis erzeugen. Mit Hilfe des Navigationsskripts "Zeige Superklasse" könnten wir uns dann diesen Aspekt auf dem Bildschirm zeigen lassen.

In den Aspekten sind die Merkmale festgelegt, die bei der Betrachtung eines Objekts unter diesem Aspekt von Bedeutung sind. Als wir die Person Durst—Benno unter dem Aspekt BERUF betrachteten (Abbildung 4-4 auf Seite 66) war im Arbeitsfenster ein Formularfeld mit der Bezeichnung "Tätigkeiten:" vorgesehen. Daß dies so war, liegt in den Eigenschaften des Aspekts BERUF. Eine wesentliche Funktion der Aspekte ist es also, daß sie als *Filter* für die Darstellung von Objekten dienen.

Lassen wir uns den Aspekt BERUF vom System D&I zeigen (Abbildung 5-4). Das Formularfenster zeigt uns vier Felder, von denen für uns im Moment nur dasjenige von Interesse ist, das die Benennung "Merkmale:" trägt. Dieses Feld enthält die Liste aller Merkmale, die zur Charakterisierung der beruflichen Be-

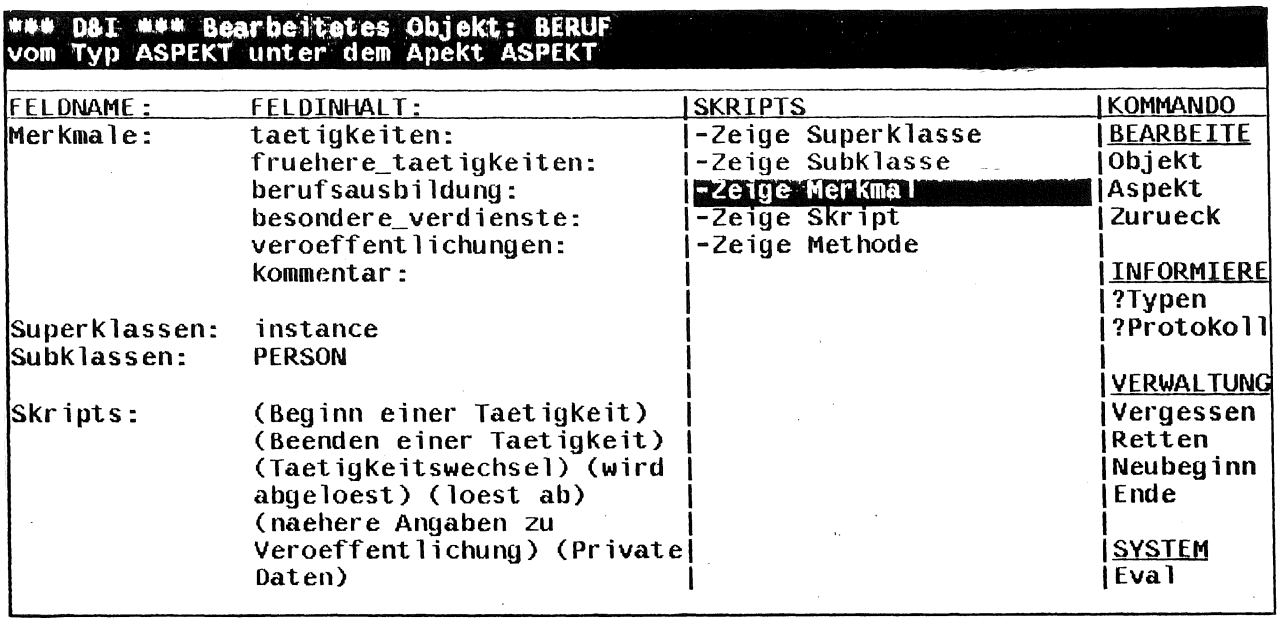


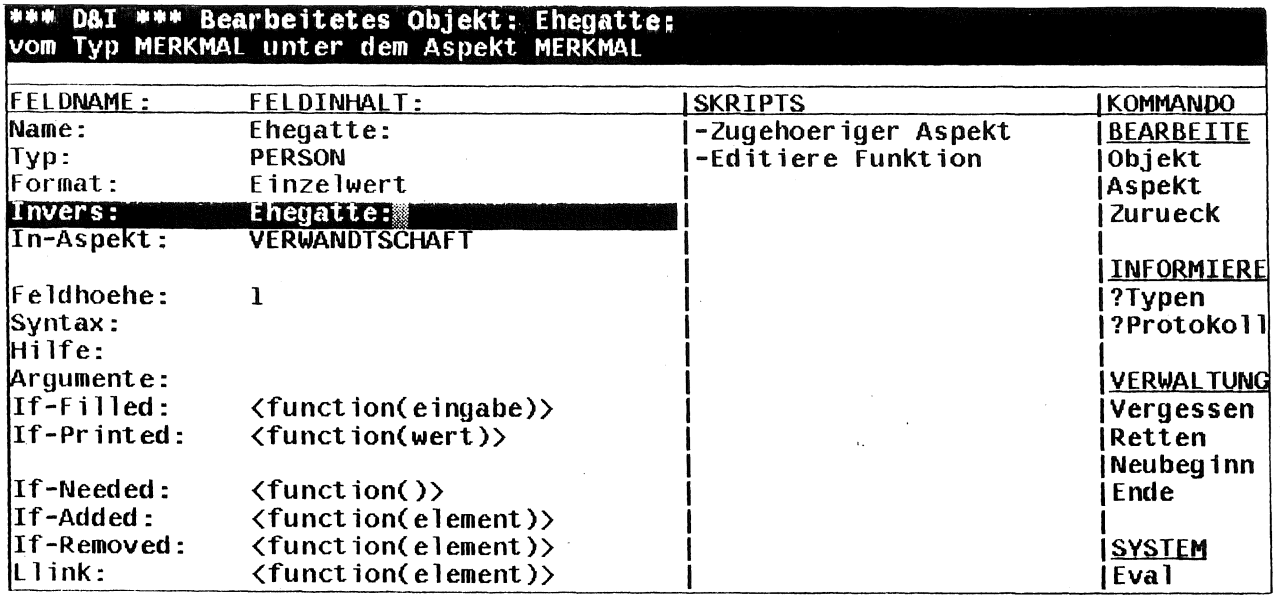
Abbildung 5-4: Darstellung des Aspekts BERUF

tätigung einer Person erforderlich sind, unter anderem die Merkmale "Tätigkeiten:", "Frühere Tätigkeiten:" und "Berufsausbildung:"

Ebenso könnten wir für den oben eingeführten Aspekt VERWANDTSCHAFT die erforderlichen Merkmale spezifizieren. Wir bräuchten dort nur im Formularfeld "Merkmale:" die entsprechenden Benennungen, beispielsweise "Eltern:", "Kinder:" und "Ehegatte:" einzutragen, und das Metasystem würde diese zusätzlichen Merkmale für alle vorhandenen und künftigen Objekte vom Typ PERSON vorsehen.

5.3.2 Aus Merkmalen leiten sich Inferenzen ab

Wenn wir ein Merkmal betrachten, müssen wir unterscheiden zwischen dem *Merkmal* selbst als abstraktem Begriff und seinen konkreten Ausprägungen, also den *Merkmalswerten*, die sich in den einzelnen Objekten manifestieren. Ein Merkmalswert, beispielsweise der Merkmalswert "Leiter des Werks für Medizinelektronik", ist Bestandteil eines konkreten Objektes der Wissensbasis (hier Durst—Benno). Ein Merkmal (hier das Merkmal "Tätigkeiten:" des Aspekts BERUF) ist definierender Bestandteil eines abstrakten Konzepts (hier des Typs PERSON) und selbst wiederum ein Konzept.



*** D&I *** Bearbeitetes Objekt: Ehegatte:
vom Typ MERKMAL unter dem Aspekt MERKMAL

FELDNAME:	FELDINHALT:	SKRIPTS	KOMMANDO
Name:	Ehegatte:	-Zugehoeriger Aspekt	BEARBEITE
Typ:	PERSON	-Editiere Funktion	Objekt
Format:	Einzelwert		Aspekt
Invers:	Ehegatte:		Zurueck
In-Aspekt:	VERWANDTSCHAFT		
			INFORMIERE
Feldhoehe:	1		?Typen
Syntax:			?Protokoll
Hilfe:			
Argumente:			VERWALTUNG
If-Filled:	<function(eingabe)>		Vergessen
If-Printed:	<function(wert)>		Retten
			Neubeginn
If-Needed:	<function()>		Ende
If-Added:	<function(element)>		
If-Removed:	<function(element)>		SYSTEM
Llink:	<function(element)>		Eval

Abbildung 5-5: Darstellung des Merkmals "Ehegatte:"

Für das oben angeführte Merkmal "Ehegatte:" des Aspekts VERWANDTSCHAFT des Typs PERSON soll gezeigt werden, wie das konzeptuelle Wissen spezifiziert werden kann, das an die Merkmale gebunden ist. Mit Hilfe des Navigationsskripts "Zeige Merkmal" gelangen wir von der Bildschirmdarstellung des Aspekts VERWANDTSCHAFT zu der des Merkmals "Ehegatte:" (Abbildung 5-5). Im Arbeitsfenster müssen wir zwei Formularfelder ausfüllen, das mit der Benennung "Typ:" und das mit der Benennung "Invers:". Weitere Formularfelder sind in diesem Fall nicht von Belang und brauchen nicht ausgefüllt werden.

Als "Typ:" tragen wir "PERSON" ein, da als Wert des Merkmals "Ehegatte:" nur ein Objekt vom Typ PERSON sinnvoll ist. Dies hat Auswirkungen auf die Dialoggestaltung beim Editieren des betreffenden Merkmalswerts als Formularfeld des Arbeitsfensters. Die Hilfekomponente des Systems D&I kann aus einer solchen Angabe Eingabemuster, Eingabevorschläge und Erklärungstexte generieren. Des weiteren steuert die Angabe des Typs den Identifikationsprozess nach dem Eintragen eines Ehegatten, weil dann in der Wissensbasis nach einer Person gesucht wird und dort gegebenenfalls eine solche neu angelegt wird.

Wenn die Person x Ehegatte der Person y ist, so gilt auch umgekehrt, daß y Ehegatte von x ist. Das Merkmal "Ehegatte:" ist also zu sich selbst invers. Daher tragen wir im Formularfeld "Invers:" die Bezeichnung "Ehegatte:" ein. Damit haben wir die Semantik des Merkmals festgelegt und befähigen den Digester zur Bildung von Inferenzen. Wenn wir in Zukunft den Ehegatten x einer konkreten Person y angeben, wird das System automatisch auch die Person y als Ehegatten von x eintragen.

Ähnlich können wir die Merkmale "Eltern:" und "Kinder:" spezifizieren. Beide nehmen Werte vom Typ PERSON an und stehen in einer Inversbeziehung zueinander.

Es gibt noch eine Reihe weiterer Beschreibungselemente, mit denen die Bedeutung von Merkmalen spezifiziert werden kann, darüber hinaus auch solche, die das Druckbild von Merkmalen bei der Darstellung als Formularfelder beeinflussen. Wesentlich für alle diese Beschreibungselemente ist jedoch, daß sie deskriptives konzeptuelles Wissen darstellen, aus dem das System D&I für den Benutzer unsichtbares prozedurales Wissen ableitet, das beim Dialog mit dem System zur Anwendung kommt.

5.3.3 Skripts gestalten den Dialog

Durch die Anzahl und Zusammenstellung der im Menü angebotenen Skripts ergeben sich die Möglichkeiten, die dem Sachbearbeiter beim Dialog mit dem Anwendersystem zur Verfügung stehen. Der Benutzer kann die Form des Dialogs selbst bestimmen und umgestalten, indem er diese Skripts mit Hilfe des Metasystems definiert und verändert.

Ebenso wie die Merkmale sind auch die Skripts einem Aspekt zugeordnet. Wenn wir den Aspekt BERUF auf dem Bildschirm betrachten (Abbildung 5-4), erscheint im Formularfeld "Skripts:" eine Auflistung aller Skripts, die für die Verarbeitung von Ereignissen vorgesehen sind, in deren Zusammenhang eine berufstätige Person steht. In diese Liste können weitere Skripts aufgenommen werden, indem die Liste ergänzt wird.

Ein Skript selbst, beispielsweise das Skript "Beginn einer Tätigkeit" kann wie jedes andere Konzept auch auf dem Bildschirm dargestellt und durch das Ausfüllen eines Formulars definiert werden. Zwei Formularfelder haben hierbei zentrale Bedeutung. Das eine enthält eine Liste der "Argumente:", die nach

der Auswahl des Skripts im Kopffenster erfragt werden. Das andere enthält die "Aktion:", die nach der Abfrage der Argumente ausgeführt wird.

Das Skript "Beginn einer Tätigkeit" besitzt zwei solche Argumente: Die Berufsbezeichnung und der Name des Arbeitgebers. Diese Argumente sind in einer ähnlichen Form repräsentiert wie die oben beschriebenen Merkmale. Die Eigenschaften von Argumenten lassen sich daher in ähnlicher Weise wie die der Merkmale auf dem Bildschirm darstellen und modifizieren.

Skriptaktionen sind im System D&I noch mangelhaft repräsentiert. Im Feld "Aktion:" steht in der Regel ein kurzes Stück Programmcode. Sofern es sich um eine Navigationsaktion handelt, was der häufigste Fall ist, ist die Aktion auch einem DV-unkundigen Benutzer verständlich. Soll das Skript komplexere Aktionen bewirken, so sind Programmierkenntnisse unumgänglich. Bei der Erprobung des Systems D&I hat sich jedoch gezeigt, daß die wesentlichen Aufgaben der Skripts in der Navigation zu Aspekten und der Identifikation von Objekten bestehen, so daß eine deskriptive Repräsentation von Skriptaktionen in der Zukunft möglich erscheint.

5.4 Repräsentation von Metawissen

Die Funktionsweise der Metakomponente des Systems D&I beruht auf Metawissen. Auch das Metawissen ist in Form von Konzepten dargestellt. Diese Konzepte bezeichnen wir als *Metakonzepte.* Metakonzepte sind die Konzepte, nach denen Konzepte aufgebaut sind. Repräsentiert sind die Metakonzepte als ObjTalk-Klassen. Es gibt im System D&I genau fünf Metakonzepte, welche die bereits angesprochenen fünf Arten von Konzepten repräsentieren:

- Das Metakonzept ASPEKT ist die Klasse aller Aspekte des Systems. Die Aspekte BERUF und PRIVAT gehören somit zu den Instanzen von ASPEKT. ASPEKT ist eine Subklasse der ObjTalk-Metaklasse class; dies ist die Ursache dafür, daß die Aspekte unter anderem die Eigenschaften von Klassen haben. Aspekte sind aber speziellere Strukturen als ObjTalk-Klassen, da sie zudem noch Merkmale und Skripts besitzen können.

- Das Metakonzept TYP ist die Klasse aller Typen des Systems. Die Typen PERSON, TAETIGKEIT, FIRMA usw. sind also Instanzen der Klasse TYP. TYP ist eine Subklasse von ASPEKT; deshalb können alle Typen des Systems D&I zugleich auch als Aspekte angesehen werden. Was die Typen von den Aspekten unterscheidet, ist daß sie die Methoden "identify:", "find:", "help:" und "syntax:" besitzen und Instanzen bilden.

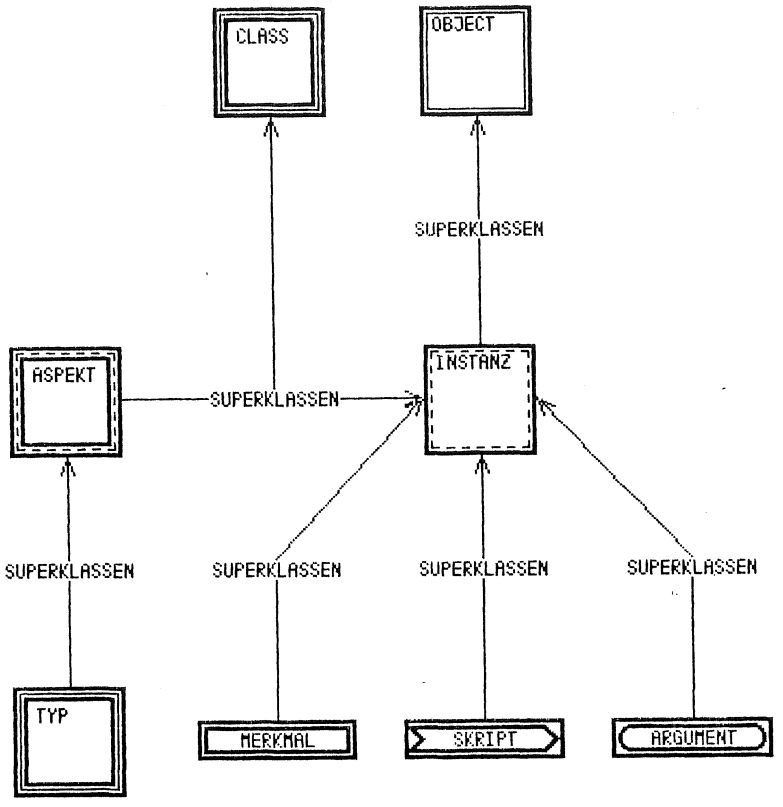


Abbildung 5-6: Metakonzepte der D&I-Wissensbasis

- Das Metakonzept MERKMAL ist die Klasse aller Merkmale des Systems. In MERKMAL ist definiert, welche Beschreibungselemente zur Charakterisierung von Merkmalen verwendet werden können. Diese Beschreibungselemente, wie z.B. *if-added*, *default* und *typ* sind gegeben als Merkmale von Instanzen der Klasse MERKMAL.
- Das Metakonzept SKRIPT ist die Klasse aller Skripts des Systems. SKRIPT definiert die Merkmale, die alle Skripts besitzen können, wie z.B. die Merkmale "Argumente:" und "Aktion:".
- Das Metakonzept ARGUMENT ist die Klasse aller Skriptargumente. In ARGUMENT sind die Merkmale definiert, die alle Skriptargumente aufweisen können.

Abbildung 5-6 zeigt die fünf Metakonzepte des Systems D&I zusammen mit der Wurzel der D&I-Klassenhierarchie, dem Aspekt INSTANZ und den ObjTalk-Metaobjekten *class* und *object*.

Alle fünf Metakonzepte sind Instanzen der Klasse TYP und sind wiederum definiert mit Hilfe von Typen, Aspekten, Merkmalen, Skripts und Skriptargumenten. Das heißt, zur Definition der Metakonzepte ist es nicht nötig, noch weitere Meta-Meta-Konzepte einzuführen. Die vorhandenen Metakonzepte sind imstande, sich selbst zu beschreiben, nachdem sie einmal mit Hilfe eines Bootstrap-Prozesses erzeugt worden sind. Eine Folge dieser Tatsache besteht darin, daß auch die Metakonzepte des Systems D&I Wissensbasisobjekte sind, die mit Hilfe des Systems D&I inspiziert und modifiziert werden können. Da derartige Modifikation im System D&I unmittelbare Wirkung haben, erfordern sie freilich große Sorgfalt.

5.5 Zusammenfassung

Wissensbasierte Systeme ermöglichen die Darstellung konzeptuellen Wissens in einer Wissensbasis. Mit Hilfe eines Metasystems ist es möglich, die Funktionalität eines wissensbasierten Systems zu ändern und zu erweitern. Das Metasystem stellt den Kontakt her zwischen dem Benutzer und den Konzepten der Wissensbasis. Dadurch kann der Benutzer das bestehende System nach seinen Vorstellungen umgestalten.

Der im System D&I eingeschlagene Weg, das konzeptuelle Wissen deskriptiv in Form von Konzepten darzustellen, und aus den Konzepten automatisch prozedurales Wissen abzuleiten, führt weg von der Frage nach dem *WIE*. An die Stelle von Programmiervorgängen tritt der Erwerb konzeptuellen Wissens. Programme werden abgelöst durch konzeptuelle Objekte, die auf der Benutzeroberfläche des Systems in Erscheinung treten können. Dadurch wird es dem Benutzer möglich, das Verhalten des Systems von Grund auf zu verstehen und zu verändern.

Dennoch zeigt das System D&I eher Ziele auf als Lösungen. Es ist durchaus noch nicht klar, wie sich die Prinzipien des Systems auf gänzlich andere Anwendungsgebiete als das der inhaltlichen Verarbeitung von Zeitungsartikeln übertragen lassen. Auf jeden Fall demonstriert das System D&I jedoch Kriterien für eine benutzerangemessene Anwendung wissensbasierter Systeme. Insbesondere wird deutlich, daß ein einfacher und bequemer Zugang zu dem Wissen, das die Arbeitsweise eines Computersystems steuert, dem Benutzer ein tiefes Verständnis und eine hohe Kontrollierbarkeit der Vorgänge ermöglicht, die er bei der Arbeit mit dem System auslöst.

6. Der Wissenseditor ZOO

Funktion und Inhalt eines Computersystems lassen sich mit Hilfe graphischer Methoden auf eine Art darstellen, die den modellhaften Vorstellungen des Benutzers vom Problembereich sehr nahe kommt. Dies hat sich am Erfolg der neuen graphischen Benutzerschnittstellen gezeigt, die erstmals im Büroinformationssystem Xerox Star [Smith et al. 82] zum Einsatz kamen und heute bereits in einer Vielzahl von Kleinrechnern zu finden sind: Die in diesen Systemen gespeicherten Informationen ebenso wie die anwendbaren Operationen zur Informationsverarbeitung sind auf dem Bildschirm durch graphische Symbole, sogenannte *Piktogramme* [Staufer 85] dargestellt, die der Bürowelt entstammen; zum Beispiel gibt es Piktogramme, die einen Aktenordner, einen Archivschrank, einen Zeilendrucker oder einen Papierkorb symbolisieren (Abbildung 6-1). Piktogramme sind aber mehr als nur graphische Abbilder interner Rechnerzustände. Man kann auf sie zeigen und dadurch die von ihnen repräsentierten Objekte bezeichnen. Piktogramme spielen daher eine ähnliche Rolle wie die Namen von Objekten.

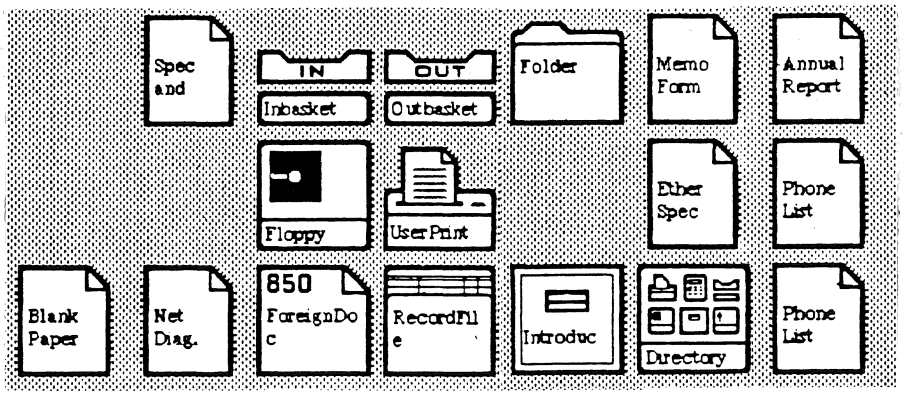


Abbildung 6-1: Darstellung von Objekten durch Piktogramme im Xerox Star

Die Darstellung interner Objekte mit Hilfe von Piktogrammen ist sehr sinnfällig und ermöglicht die Konstruktion leicht erlernbarer Benutzerschnittstellen. Zur externen Darstellung von Wissen wünscht man sich aber eine größere Darstellungskapazität, als sie der Xerox Star erlaubt:

- Man möchte nicht nur Objekte aus der Bürowelt darstellen sondern Objekte aus beliebigen Gegenstandsbereichen.

- Es muß möglich sein, beliebige Beziehungen zwischen Objekten graphisch darzustellen. (Das Star Interface erlaubt nur hierarchische Beziehungen, z.B. zwischen einem Aktenordner und den in ihm enthaltenen Dokumenten.)
- Es muß möglich sein, nicht nur konkrete Objekte (wie etwa die Akte *XYZ*), sondern auch abstrakte Begriffe (etwa den Begriff *Akte*) graphisch darzustellen.

Es gibt zwar sogenannte *Knowledge Engineering Tools*, also Werkzeuge zur Bearbeitung von Wissensbasen, die alle diese Anforderungen erfüllen. Diese machen aber nur sparsamen Gebrauch von Graphik, wie etwa der Smalltalk-Browser [Goldberg 84] oder die Wissensbasisverwaltung des Systems KEE [KEE 85]. Bei der Entwicklung des hier vorgestellten Wissenseditors ZOO wurde das Ziel verfolgt, die Universalität eines Knowledge Engineering Tools mit der Anschaulichkeit einer piktogrammbasierten Benutzerschnittstelle zu verbinden.

6.1 Zweckbestimmung des Wissenseditors ZOO

Das System ZOO hat zwei Hauptfunktionen: es ist gleichzeitig ein Instrument zur Untersuchung von Wissensbasen und ein Werkzeug zum Aufbau und zur Modifikation von Wissensbasen.

Der Begriff *Wissensbasis* ist hier in einem technischen Sinn zu verstehen. Eine Wissensbasis ist eine Ansammlung von Softwareobjekten, die einen (sinnvollerweise) abgrenzbaren Wissensbereich modellieren. Für derartige Wissensbereiche, die in einzelnen Wissensbasen dargestellt sein können, lassen sich folgende typischen Beispiele nennen: das spezielle Anwendungsgebiet, die Funktionalität, das Benutzermodell und das Systemmodell (Selbstbild) eines wissensbasierten Systems.

Eine derartige Wissensbasis kann auf verschiedene Weise entstanden sein: Sie kann mit einem Texteditor auf einer Datei in einem externen Format erstellt worden sein und ins System geladen worden sein, sie kann als Arbeitsergebnis einer Sitzung an einem wissensbasierten Anwendersystem entstanden sein, oder sie kann mit Hilfe des Systems ZOO aufgebaut worden sein.

6.1.1 Untersuchung von Wissensbasen

Als *Untersuchungsinstrument* hat das System ZOO den Zweck, eine Wissensbasis (oder Teile davon) in graphischer Form zu visualisieren. Es soll einem Gebietsexperten, der nicht notwendig Software-Experte ist, ermöglichen, die Wissensbasis in einer für ihn angemessenen Form betrachten zu können. Das graphische Darstellungsformat des Systems ZOO soll es gestatten, den Aufbau der Software eines Computers anhand einer zweidimensionalen Darstellung in einer ähnlichen Form zu inspizieren, wie dies auf einem elektrischen Schaltbild mit der Hardware-Architektur eines Computers möglich ist. Das Ziel war es, das im System gespeicherten Wissens mittels einer graphischen Wissensrepräsentationssprache extern darzustellen.

ZOO hat die Funktionalität eines Browsers, also eines Softwarewerkzeugs, das seinen Benutzer befähigt, durch ein Netz von Wissensbasisobjekten zu navigieren. Bei der Darstellung von Wissensbasen besitzt das System ZOO die Funktion eines Filters: Unwesentliche Informationen werden versteckt, wesentliche Informationen werden visualisiert.

6.1.2 Aufbau und Modifikation von Wissensbasen

Das System ZOO soll seinen Benutzer sowohl bei der Spezifikation und dem Design, als auch bei der Implementierung eines wissensbasierten Systems unterstützen. Da die Funktion eines wissensbasierten Systems durch Wissensbasen bestimmt ist, hat das System ZOO den Entwurf und die Aktualisierung von Wissensbasen zum Gegenstand.

In der Praxis zeigt sich, daß bei der traditionellen Problemspezifikation ohne besondere Rechnerunterstützung gerne Blockbilder und Netzgraphiken als Darstellungsmittel verwendet werden. Im System ZOO werden derartige graphische Darstellungsmethoden ebenfalls angewandt. Das System ZOO soll für seinen Benutzer ähnlich einfach erscheinen wie ein Editor zur Konstruktion von Bürographiken. Es können graphische Objekte erzeugt werden und beschriftete Pfeile zwischen ihnen gezogen werden. Die Auswirkungen sind freilich viel größer als beim Erstellen einer Bürographik. Mit jedem erzeugten graphischen Objekt wird ein Objekt der Wissensbasis angelegt. Mit jeder gezogenen Linie werden zwei interne Objekte durch eine Relation verknüpft.

Weil gleichzeitig mit der Graphik eine Wissensbasis entsteht, sind die Vorgänge des Designs und der Implementation wissensbasierter Systeme parallelisiert. Die

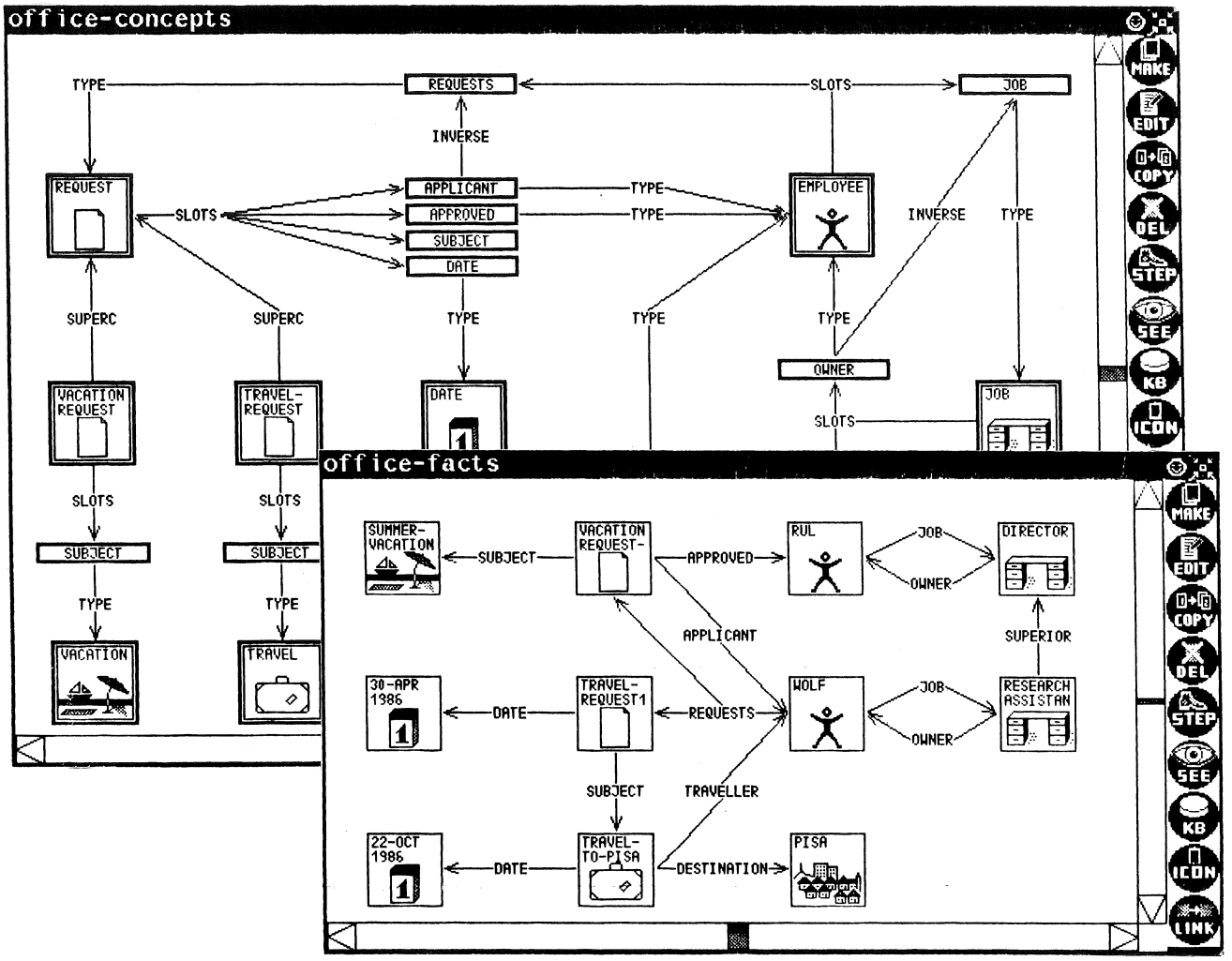


Abbildung 6-2: Visualisierung von Wissensbasen mit dem System ZOO

erstellten Spezifikationen stellen zugleich unmittelbar ablauffähige Software dar. Das System ZOO macht so das Rapid Prototyping wissensbasierter Systeme möglich.

6.2 ObjTalk-Wissensbasen

Das gesamte in einem wissensbasierten System explizit repräsentierte Wissen läßt sich auf einzelne Wissensbasen aufteilen. In einem in ObjTalk implementierten wissensbasierten System besteht eine derartige Wissensbasis aus einer Menge von ObjTalk-Objekten. Wissensbasen sind in ObjTalk selbst als Objekte repräsentiert, und zwar als Instanzen der Klasse *knowledge-base*.[32]

[32]Dies ist keine Eigenschaft des ObjTalk-Grundsystems, sondern eine ObjTalk-Erweiterung, die vom Autor vorgenommen wurde.

Jede ObjTalk-Wissensbasis hat Wissen einer bestimmten Abstraktion zum Gegenstand: Faktenwissen, konzeptuelles Wissen oder Metawissen:

- *Faktenwissensbasen* enthalten die Arbeitsergebnisse und Zwischenzustände einzelner Sitzungen am Anwendungssystem. In herkömmlichen Systemen ist dieses Wissen repräsentiert in der Datenbasis des Systems. Faktenwissensbasen enthalten in der Regel einfache Objekte (Instanzen) und deren Slots. Faktenwissensbasen sind dynamisch, ihr Inhalt ändert sich bei der normalen Benutzung des Systems und ist daher von Sitzung zu Sitzung unterschiedlich. Eine Faktenwissensbasis kann extern auf einem File bis zur nächsten Benutzung des Systems gespeichert werden. Notwendig ist dies aber nicht, da der Inhalt einer Faktenwissensbasis häufig nur Bedeutung für die aktuelle Sitzung hat.

- *Konzeptuelle Wissensbasen* beschreiben die dem System zugrundeliegenden Konzepte, die den Verwendungszweck des Systems bestimmen. Konzeptuelle Wissensbasen enthalten konzeptuelle Objekte, also im wesentlichen Klassen, Slotbeschreibungen und Methoden; diese stellen das eigentliche "ObjTalk-Programm" dar. Konzeptuelle Wissensbasen besitzen meist eine externe Darstellung auf einem File, der die Definitionen der betreffenden konzeptuellen Objekte enthält. Konzeptuelle Wissensbasen sind in erster Näherung statisch, ihr Inhalt ändert sich bei der normalen Benutzung eines Systems nicht. Unterschiedliche Systeme benötigen freilich unterschiedliche konzeptuelle Wissensbasen. Zum Zweck der Umgestaltung eines Systems kann eine konzeptuelle Wissensbasis durch den Programmierer oder bei Verwendung eines Metasystems auch durch den Benutzer des Systems verändert werden.

- Es gibt im wesentlichen nur eine *Metawissensbasis*, und diese besteht aus dem ObjTalk-Grundsystem selbst. Diese Wissensbasis enthält die fundamentalen ObjTalk-Konzepte, also die Metaobjekte *class*, *object*, *slot* usw., und deren Slots. Die ObjTalk-Metawissensbasis ist für alle in ObjTalk implementierten Systeme dieselbe. Änderungen der ObjTalk-Metawissensbasis bedeuten Änderungen an der ObjTalk-Sprachdefinition. Sie werden nur von den ObjTalk-Implementatoren vorgenommen. Als Metawissensbasen im weiteren Sinne können die Erweiterungen des ObjTalk-Grundsystems angesehen werden.

Unterschiedliche Wissensbereiche werden durch verschiedene Wissensbasen modelliert. Zu jedem Wissensbereich gibt es konzeptuelle Wissensbasen und Faktenwissensbasen:

- *Interaktionswissensbasen* enthalten Objekte, welche die Interaktion mit dem Benutzer beschreiben und definieren. Die möglichen Interaktionskonzepte sind dargestellt in konzeptuellen Wissenbasen; Beispiele für derartige Kon-

zepte sind etwa die Klassen von Fenstern und Menüs des verwendeten Fenstersystems [Fabian 84]. Die bei der Interaktion tatsächlich verwendeten Interaktionsmedien und Interaktionsformen sind dargestellt in Faktenwissensbasen, diese enthalten beispielsweise die tatsächlich verwendeten Fenster- und Menü-Instanzen.

- *Benutzermodelle* sind dargestellt durch konzeptuelle Wissensbasen und Faktenwissensbasen. Die einen enthalten beispielsweise abstrakte Handlungsmodelle des Benutzers, die anderen konkrete Default-Einstellungen und Dialog-Aufzeichnungen. Das aktive Hilfesystem AKTIVIST [Schwab 84] nutzt derartige Benutzermodelle und Handlungsmodelle, um den Benutzer eines Editorsystems zu unterstützen.

- Die grundlegenden Begriffe des *Gebietswissens* sind repräsentiert in Form konzeptueller Objekte einer konzeptuellen Wissensbasis, also durch Klassen, Methoden und Slotbeschreibungen von Objekten aus der Welt des Anwendungsgebiets. Konkrete, im Moment betrachtete Sachverhalte aus dem Anwendungsgebiet werden dargestellt durch die Instanzen von Klassen der konzeptuellen Wissensbasis; diese Instanzen gehören einer Faktenwissensbasis an. Über derartige Gebietswissensbasen verfügt das in Kapitel 4 und 5 beschriebene System D&I.

- *Systemmodelle* sind ebenfalls durch Wissensbasen dargestellt: Eine konzeptuelle Wissensbasis repräsentiert die Funktionalität eines Systems; diese enthält konzeptuelle Objekte, welche die Kommandos und Eingabedaten eines Systems beschreiben. Aktuelle und frühere Systemzustände werden durch die Objekte und Slots einer Faktenwissensbasis dargestellt und verknüpft. Die Skripts im System D&I können als Teil eines derartigen Systemmodells betrachtet werden. Auf ein explizit repräsentiertes Systemmodell greift auch das System XS-2 [Stelovsky 84] zurück, das mit herkömmlichen Mitteln (Programmiersprache MODULA-2) implementiert wurde.

Mit der Aufteilung von Wissen auf einzelne Wissensbasen wird die Zerlegung eines Problemraums in voneinander weitgehend unabhängige Teilbereiche bezweckt im Sinne des von Simon [Simon 81] geprägten Begriffs von einem *"nearly decomposable system"*. Dennoch gibt es Abhängigkeitsrelationen zwischen einzelnen Wissensbasen:

Eine schwache Abhängigkeit besteht dann, wenn bestimmte Slots von Objekten einer Wissensbasis Referenzen auf Objekte anderer Wissensbasen besitzen. Die Objekte der beteiligten Wissensbasen sind in ihrem gegenwärtigen Zustand miteinander gekoppelt, sie sind aber prinzipiell unabhängig voneinander denkbar.

Eine starke Abhängigkeit besteht dann, wenn die Objekte einer Wissensbasis aus Objekten einer anderen Wissensbasis durch Instantiierung, Subklassenbildung oder Kopieren hervorgegangen sind. In diesem Fall enthält die letztere Wissensbasis die Konzepte zur Erzeugung der ersteren Wissensbasis. Faktenwissensbasen, konzeptuelle Wissensbasen und Metawissensbasen sind entsprechend dieser Abhängigkeitsrelation geordnet: Die Objekte der Faktenwissensbasen sind erzeugt mit Hilfe von konzeptuellen Objekten aus konzeptuellen Wissensbasen. Zur Erzeugung konzeptueller Objekte werden die Metaobjekte der ObjTalk-Metawissensbasis benötigt. Da ObjTalk durch Anwendung von Bootstrapping-Techniken weitgehend in sich selbst implementiert ist, sind die Objekte der Metawissensbasis so anzusehen, als wären sie aus ihrer eigenen Mitte ohne äußeres Zutun von alleine entstanden. Abbildung 6-3 zeigt (in etwas vereinfachter Form), wie sich die Metaklasse *class* unter Voraussetzung ihrer eigenen Existenz als Instanz ihrer selbst definieren läßt.[33]

```
(ask class make: class with:
    (slots   = (superc slots methods corefs constraints))
    (methods = (make: instantiate: init: redefining-form:)))
```

Abbildung 6-3: Definition der Metaklasse *class* aus sich heraus

6.3 Graphische Darstellung von Wissen mit dem System ZOO

6.3.1 Graphische Darstellungselemente des Systems ZOO

Gewöhnliche Objekte, konzeptuelle Objekte und Metaklassen der Sprache ObjTalk lassen sich mit dem System ZOO in graphischer Form darstellen. Zu diesem Zweck stehen zwei fundamentale graphische Darstellungsmöglichkeiten zur Verfügung: Piktogramme und Pfeile.

[33] Dies ist keinesfalls so absurd, wie es auf den ersten Blick anmutet, beispielsweise ein Pascal-Compiler, der selbst in Pascal implementiert ist, stellt eine Entsprechung zu diesem Verhalt dar.

Piktogramme bieten eine Vielfalt von Gestaltmerkmalen, mit welchen die unterschiedlichen Eigenschaften von Objekten visualisiert werden können:

- Piktogramme enthalten als wichtigstes Merkmal eine bildliche Darstellung eines Objekts. Einer informellen Konvention folgend gibt es für viele Arten von Objekten weithin anerkannte standardisierte bildliche Darstellungen, die für einen großen Personenkreis unmittelbar verständlich sind.
- Piktogramme haben eine Lage in der Ebene und bilden untereinander eine zweidimensionale Anordnung.
- Piktogramme haben einen Umriß, welcher rechteckige, runde oder beliebig unregelmäßige Form annehmen kann, und können einen Rahmen, das heißt eine Umrandung durch Umrißlinien, besitzen.
- Piktogramme können eine Inschrift tragen. Diese kann mit unterschiedlichen Schrifttypen (Fonts) und Darstellungsattributen (Inversschrift, Versalien, Unterstreichung) ausgeführt sein.
- Das gesamte Piktogramm kann durch eine besondere Darstellung herausgehoben erscheinen, z.B. durch Inversdarstellung oder durch eine Markierung.

Gruppen von Piktogrammen können zusammen mit verbindenden Pfeilen Netze bilden. Pfeile bieten ebenfalls eine Reihe von unterschiedlichen Darstellungsmöglichkeiten:

- Pfeile stellen gerichtete Beziehungen zwischen Ausgangs- und Endpunkten her.
- Pfeile können eine Beschriftung tragen.
- Pfeile können sich in verschiedene Richtungen gabeln.
- Pfeile können verschiedene Erscheinungsformen besitzen (strichliert, mit Spitzen versehen, verschiedene Linienbreiten)

Durch den sinnvollen Einsatz derartiger Gestaltmerkmale kann die Wahrnehmbarkeit von Information entscheidend erhöht werden. Die Gestaltgesetze der sogenannten Berliner Schule [Wertheimer 23; Moritz 83] geben hierfür hilfreiche Kriterien ab. Bei der Umsetzung derartiger Kriterien zur Visualisierung von Objekten wurde von folgenden Überlegungen ausgegangen:

- Die Gestaltmerkmale sollen gut mit den visualisierten Eigenschaften korrespondieren. So sollte ein zweiwertiges Attribut (etwa Klasse/Instanz) nicht auf ein Gestaltmerkmal großer Variation (etwa Inschrift des Piktogramms) abgebildet werden.
- Wesentliche globale Kennzeichen (z.B. die Unterscheidung von Klassen, Instanzen, Slots und Methoden) sollten auf einen Blick ersichtlich sein. Es eignen sich hierfür die Gestaltmerkmale Umriß und Umrandung.

- Wesentliche Gemeinsamkeiten, z.B. zwischen gleichartigen Objekten bzw. die Zusammengehörigkeit von Instanz und Klasse sollten sich in einem gleichartigen Erscheinungsbild zeigen. Hierfür bieten sich gleichartige Piktogramme an.
- Als individuelles Unterscheidungskennzeichen für Objekte gleicher Art bietet sich die Beschriftung eines Piktogramms mit dem Namen an. Darüber hinaus sollten möglichst keine weiteren individuellen Eigenschaften im Piktogramm dargestellt werden, damit die Übersichtlichkeit gewahrt bleibt.
- Es sollte einfache Regeln geben, die festlegen, auf welche Weise ein Objekt im Standardfall dargestellt wird. Dadurch ist eine automatische Generierung von Piktogrammen durch den Rechner möglich. Diese Regeln sollten aber auf Wunsch durchbrochen werden können.
- Wegen der Verwechslungsgefahr sollten nicht mehrere Objekteigenschaften durch dasselbe Gestaltmerkmal dargestellt werden, z.B. zwei Inschriften zur Darstellung von zwei Objekteigenschaften.
- Relationen zwischen Objekten erfordern verknüpfende Darstellungselemente. Für gerichtete Beziehungen kommen Pfeilverbindungen und räumliche oben-unten-Beziehungen in Frage, für Äquivalenzrelationen räumliche Gruppenbildung.

Nicht alle oben genannten Gestaltungsmöglichkeiten werden vom System ZOO ausgeschöpft. Manche, wie zum Beispiel die räumliche Anordnung der Piktogramme, sind Sache des Benutzers und ergeben sich nicht automatisch aus Regeln. Manche sind für künftige Erweiterungen vorbehalten. In den folgenden Abschnitten wird der definierte Standard der graphischen Wissensdarstellung des Systems ZOO beschrieben.

6.3.2 Graphische Darstellung gewöhnlicher Objekte

Wie bereits erwähnt, dienen Piktogramme zur Repräsentation von Objekten. Im Normalfall hat ein solches Piktogramm quadratischen Umriß. Das Piktogramm ist mit dem Namen des Objekts beschriftet. Es enthält ein graphisches Symbol, das in der Regel die Klassenzugehörigkeit des Objekts verbildlicht. Merkmale eines Objekts werden in textueller Form durch den Namen des Merkmals visualisiert, der durch eine gerade Linie mit dem Piktogramm des betreffenden Objekts verbunden ist. Wenn der Wert des Merkmals aus einem oder mehreren Objekten besteht, gehen von dem Merkmalsnamen Pfeile aus, die auf die Piktogramme der zugeordneten Objekte verweisen. Merkmale, die keine Objekte als Werte aufweisen, können nicht als Pfeile, sondern nur mit Hilfe eines Text-Formulars visualisiert werden.

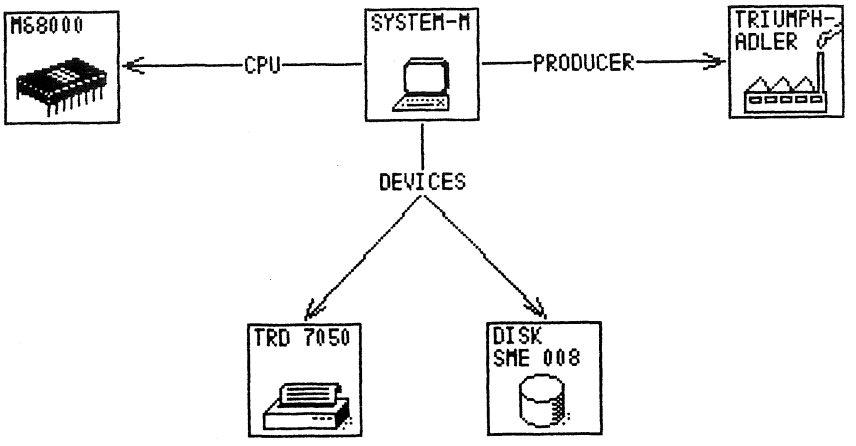


Abbildung 6-4: Graphische Darstellung des Objekts *System-M*

Wissen, insbesondere Faktenwissen, erscheint also im Darstellungsformat des Systems ZOO als ein Netz aus Piktogrammen als Knoten und beschrifteten Pfeilen als Kanten. Dabei werden die Objekte der Wissensbasis durch Piktogramme dargestellt, die beschrifteten Pfeile stehen für die Relationen zwischen den Objekten (Abbildung 6-4).

6.3.3 Graphische Darstellung konzeptueller Objekte

Konzeptuelle Objekte[34] werden mit dem System ZOO auf ähnliche Weise dargestellt wie die gewöhnlichen Objekte, die zur Repräsentation von Fakten dienen.

Klassen erscheinen als Piktogramme, die mit einem dicken Rahmen umgeben sind (Abbildung 6-5). Die Instanzen einer Klasse erhalten in der Regel dasselbe graphische Symbol in ihrem Piktogramm wie ihre Klasse. Im System ZOO wird die Superklassenrelation ebenso wie die Relationen zwischen gewöhnlichen Objekten mit Hilfe beschrifteter Pfeile *("superc")* dargestellt. Im graphischen

[34]Siehe hierzu auch Abschnitt 3.4.2 auf Seite 45.

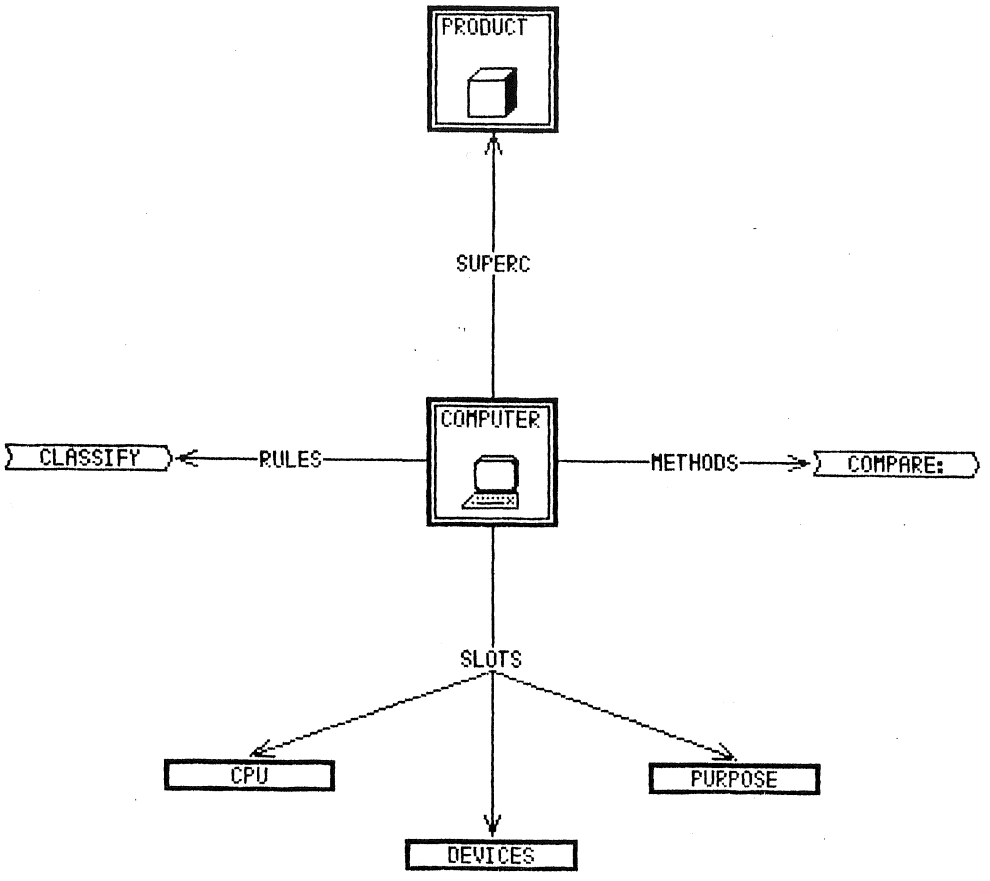


Abbildung 6-5: Graphische Darstellung der Klasse *Computer*

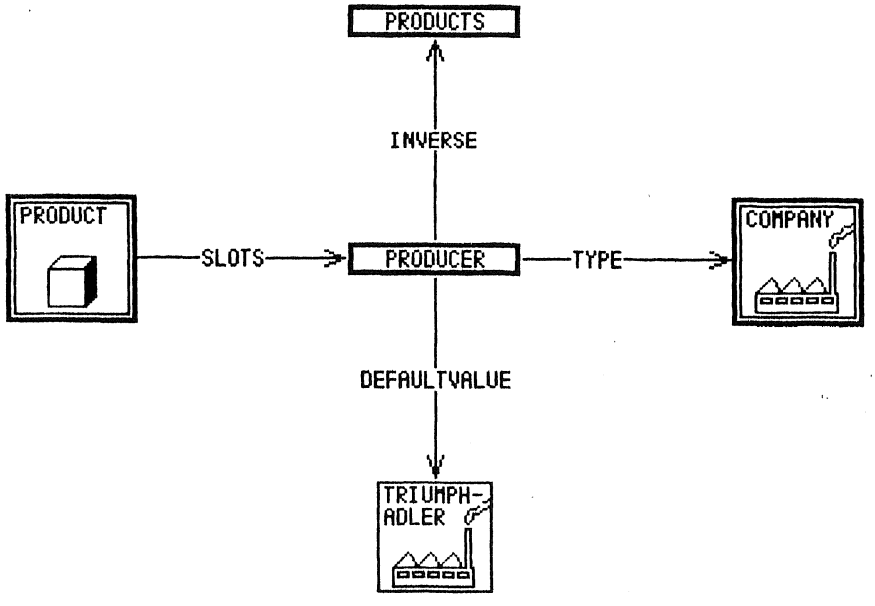


Abbildung 6-6: Graphische Darstellung der Slotbeschreibung *Producer*

Darstellungsformat des Systems ZOO übernimmt eine Klasse ohne eigens für sie definiertes graphisches Symbol das graphische Symbol ihrer nächsten Superklasse. Die graphischen Symbole der Piktogramme im System ZOO werden also von einer Klasse auf ihre Subklassen und auf ihre Instanzen vererbt und stellen so selbst eine Visualisierung des Vererbungsvorgangs dar. In gleicher Art wie die Klassen werden auch die Constraints im System ZOO dargestellt.

Im System ZOO sind *Slotbeschreibungen* durch dick umrahmte längliche Rechtecke dargestellt, in denen der Namen des Slots eingetragen ist (Abbildung 6-6). Die Relation *slots*, die Klassen und Slotbeschreibungen verknüpft, ist auf ganz normale Weise als beschrifteter Pfeil dargestellt. Die Merkmale *format*, *type*, *inverse* und *defaultvalue* von Slotbeschreibungen lassen sich im System ZOO in einheitlicher Weise durch beschriftete Pfeile bzw. durch Formularfelder darstellen.

Auch der *filter:* und der *body:* von *Methoden* sowie die beschreibenden Merkmale von *Regeln* sind Slots, die in einem Formular oder mit Hilfe von Pfeilen dargestellt werden können. Beide Arten von konzeptuellen Objekten, Methoden und Regeln werden im System ZOO durch Piktogramme dargestellt, deren äußere Form an das Implikationssymbol "$=>$" erinnert (Abbildung 6-5).

6.3.4 Graphische Darstellung der Metaobjekte

Im System ZOO sind die Metaobjekte[35] *object*, *class*, *Action*, *Method*, *Extended-Method*, *Rule*, *slot* und *constraint* an den vordefinierten graphischen Symbolen erkennbar, mit denen ihre Piktogramme ausgestattet sind. Ansonsten erscheinen sie selbst, ebenso wie ihre Methoden und Slotbeschreibungen, sowie ihre Relationen zu Superklassen und Subklassen in derselben Weise wie bei allen anderen Klassen auch.

Das Piktogramm von *object* erscheint als ein Quadrat, das mit einem dicken Rahmen (als Klassenkennzeichen) umgeben ist. Das Piktogramm von *class* ähnelt dem von *object*, hat aber ein dickes Quadrat im Innern einbeschrieben, also einen Rahmen mehr als das *object*-Piktogramm. Durch die dreifache Umrahmung wird ausgedrückt, daß die Instanzen von *class* wiederum Klassen sind, d.h. daß *class* eine Metaklasse ist.

---

[35]Die Bedeutung der Metaobjekte ist im Abschnitt 3.4.3 auf Seite 48 beschrieben.

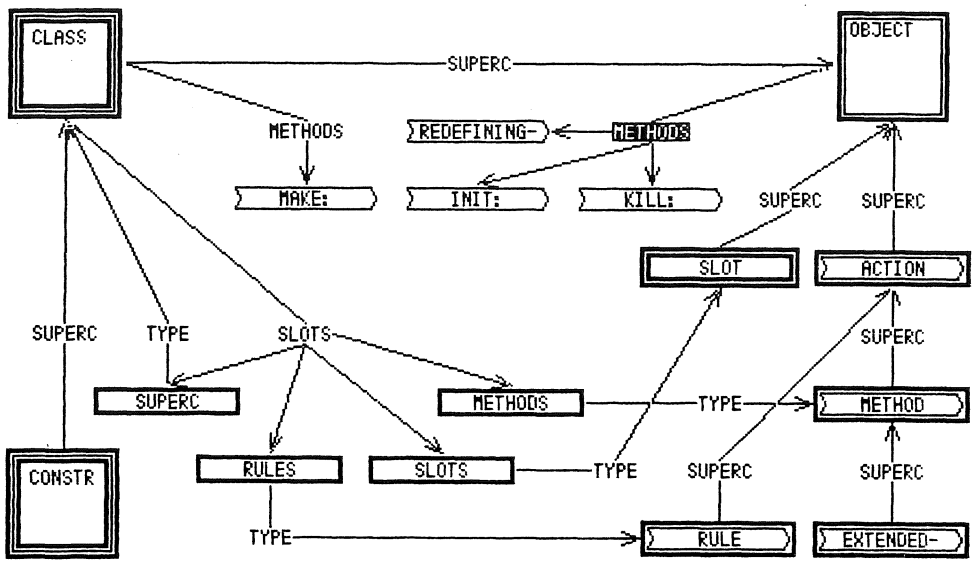


Abbildung 6-7: Graphische Darstellung der Metaobjekte

Die Klasse aller Slotbeschreibungen, das Metaobjekt *slot*, erscheint als doppelt umrandete Rechteckfläche und weist einen Rahmen mehr auf als ihre Instanzen. Die vier Metaobjekte *Action*, *Method*, *Extended-Method* und *Rule*, welche die Klassen aller Aktionen, Methoden, erweiterter Methoden, bzw. Regeln repräsentieren, werden im System ZOO durch pfeilförmige Ikonen dargestellt, die mit einem dicken Rahmen umgeben sind. Die Metaklasse *constraint* besitzt als Subklasse von *class* diesselbe graphische Darstellung wie *class*.

6.4 Der Wissenseditor ZOO

Der Wissenseditor ZOO ermöglicht einen interaktiven Zugang zu dem Wissen, das von einem Anwendungssystem erworben und genutzt wird. Dieser Zugang ist freilich nicht für den normalen Anwendungsfall der Systembenutzung gedacht. Im Normalfall soll der Benutzer nach wie vor über eine auf das Problem zugeschnittene Benutzerschnittstelle mit dem Anwendungssystem interagieren. Eine solche Benutzerschnittstelle verdeckt die Implementationsdetails des Anwendungssystems, faßt atomare Operationen zu mächtigeren Funktionen zusammen und sieht für die Benutzereingaben Defaults vor. Die interne Repräsentation der Fakten und Konzepte des Anwendungssystems bleibt dabei verborgen. Dies ist durchaus erwünscht, weil dadurch die Benutzung des Systems im Regelfall vereinfacht wird.

Wenn jedoch der Einblick in die Interna eines Anwendungssystems gewünscht wird, sei es, um dessen Funktionsweise besser zu verstehen, sei es, um das Anwendungssystem im Verhalten zu verändern, werden andere Schnittstellen erforderlich; denn auf diese Aufgabenstellung sind übliche Benutzerschnittstellen nicht eingerichtet. In einem solchen Fall ist der Einsatz des Systems ZOO von Nutzen. Mit dem System ZOO wird es dann möglich, sowohl die vom Benutzer aufgebauten Faktenwissensbasen, als auch die vom Programmierer entworfenen konzeptuellen Wissensbasen des Anwendungssystems in graphischer Form zu inspizieren und mit Hilfe direkter Manipulation zu editieren.

Das System ZOO verfügt über eine breites Spektrum von Funktionen zur Bearbeitung von Wissensbasen:

- Funktionen zur *Wissensbasisverwaltung* ermöglichen das Erzeugen, Selektieren und Löschen von Wissensbasen sowie das Laden und Abspeichern von Wissensbasen als Inhalt von Dateien.
- Funktionen zum *Visualisieren* von Objekten sowie ihrer Relationen und Attribute ermöglichen das Inspizieren von bestehenden Wissensbasen.
- Funktionen zur *Manipulation* von Objekten sowie ihrer Relationen und Attribute dienen zum Aufbau und Verändern von Wissensbasen.
- Mittels *graphischer Funktionen* können Layout und Gestalt der graphischen Objekte verändert werden, die den Inhalt einer Wissensbasis visualisieren.

Das System ZOO läßt sich somit ansehen als eine Benutzerschnittstelle zum gesamten durch ObjTalk-Objekte repräsentierten Wissen eines Softwaresystems. ZOO ermöglicht die externe Repräsentation dieses Wissens auf einem hochauflösenden Rasterbildschirm im bereits beschriebenen graphischen Darstellungsformat. Die Aufgabe von ZOO besteht darin, beide Wissensdarstellungen, die interne und die externe, stets miteinander konsistent zu halten.

Als Schnittstelle zur internen Wissensrepräsentation dienen Bildschirmfenster vom Typ *ZOO-Window* (Abbildung 6-8). Jedes derartige Bildschirmfenster enthält die graphische Darstellung einer Wissensbasis. Ein ZOO-Window stellt also die externe Entsprechung einer Wissensbasis dar. Die Elemente der externen Wissensdarstellung des Systems ZOO, Piktogramme, Pfeile und Beschriftungen, sind als graphische Objekte innerhalb eines ZOO-Windows sichtbar. Jedes dieser graphischen Objekte repräsentiert ein Objekt der Wissensbasis bzw. eine Eigenschaft eines derartigen Objekts.

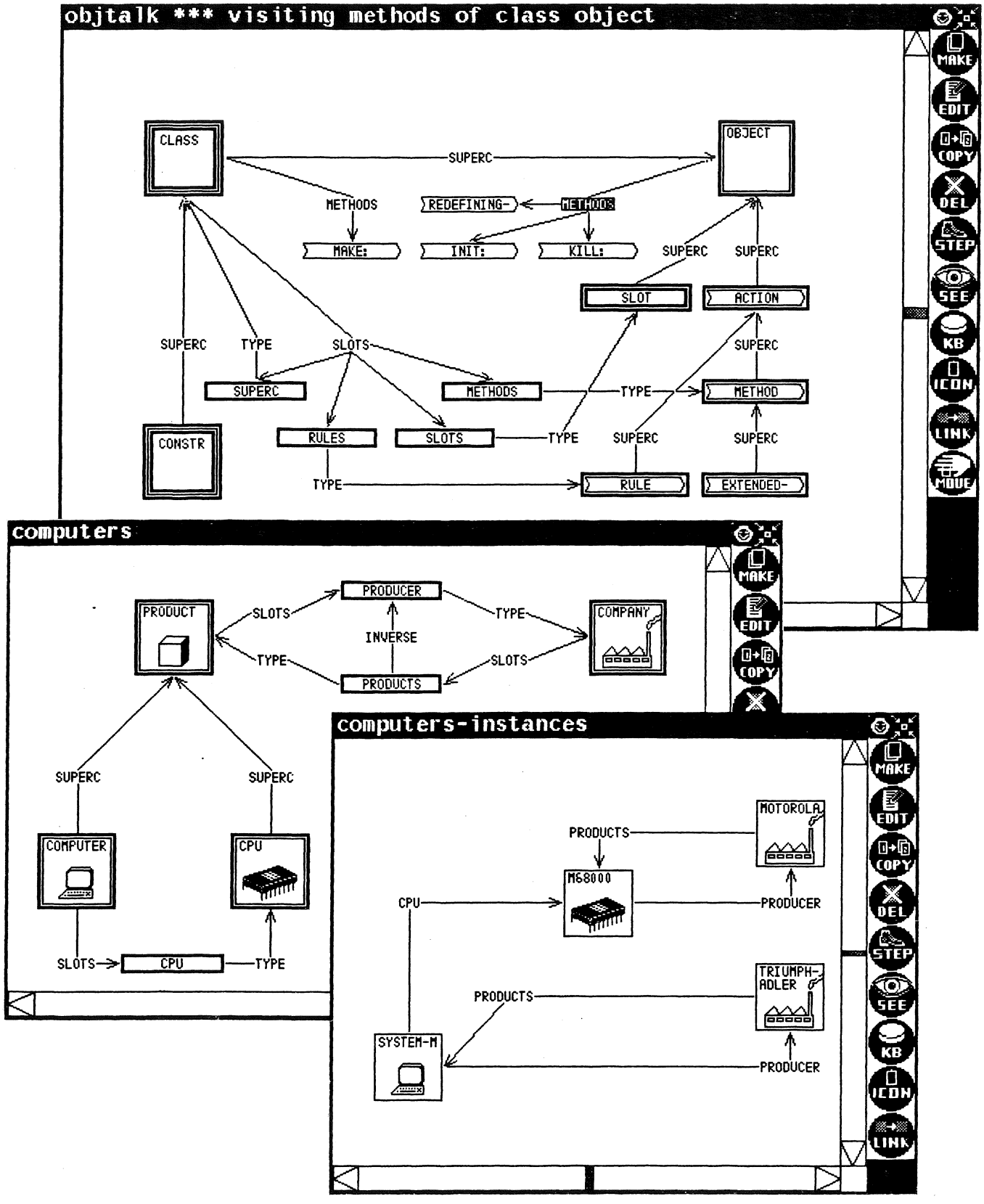


Abbildung 6-8: ZOO-Windows

Unter allen in einem ZOO-Window dargestellten Objekten kann eines graphisch hervorgehoben werden, seine graphische Repräsentation erscheint für den Benutzer deutlich ersichtlich in Inversdarstellung. Ein solches Objekt gilt als *selektiert*, d.h. alle objektspezifischen Operationen beziehen sich auf dieses Objekt. Objekte können durch eine Zeigehandlung selektiert werden; denn sie sind auf Bildschirmbereichen dargestellt, die mit Hilfe eines Zeigeinstruments (Maus) aktivierbar sind.

Der Titel eines ZOO-Windows ist mit dem Namen der im Moment editierten Wissensbasis und dem Namen des selektierten Objekts beschriftet. Ein ZOO-Window besitzt am rechten Rand eine Reihe von graphischen Objekten, durch welche die Benutzerfunktionen des Systems ZOO visualisiert sind. Diese Benutzerfunktionen können mit Hilfe der Maus ausgelöst werden. Dabei handelt es sich zum einen um Funktionssymbole, die das Auslösen von Anwendungsfunktionen des Systems ZOO erlauben. Zum andern finden sich dort sogenannte Scrollbars mit Lifts [Bauer J. 85], die das Verschieben des Fenster-Inhalts ermöglichen.

Die Selektionen von Objekten und das Auslösen von Anwenderfunktionen des Systems erfolgen also stets über Zeigehandlungen, wobei nur ein Knopf der Maus, nämlich der linke, betätigt werden muß. Dies macht den Umgang mit dem System sehr einfach. Dabei erfolgt die Interaktion mit dem System nach der *Noun-Verb-Syntax*: Es muß zuerst ein Objekt bezeichnet werden, und dann erst die Operation, die auf das Objekt angewandt werden soll. Dies hat den Vorteil, daß die Interaktion weitgehend ohne besondere *System-Modi* [Tesler 81] erfolgt. Nach der Selektion eines Objekts gerät man nicht in einen besonderen Systemzustand, der die Angabe einer Operation zwingend erfordert. Auf eine irrtümliche Selektion eines Objekts kann jederzeit eine weitere Selektion folgen. Umgekehrt kann auch nach dem Auslösen einer Anwenderfunktion sofort eine zweite betätigt werden, wenn sie sich auf das bereits selektierte Objekt bezieht.

Die angebotenen Anwenderfunktionen sind alle von elementarer Natur. Komplexe Operationen, welche die Spezifikation einer Vielzahl von Parametern erfordern, und dadurch den Benutzer in eine systemgesteuerte Dialogsequenz zwingen, sind nicht vorgesehen. Stattdessen legt das System bei der Ausführung von Anwenderfunktionen Defaults zugrunde und konfrontiert den Benutzer sofort mit dem so ermittelten Ergebnis. Wenn der Benutzer mit dem Ergebnis nicht gleich zufrieden ist, kann er es dann durch Auslösen weiterer Funktionen

im gewünschten Sinne korrigieren. Dadurch ist die Interaktion für den Benutzer an beliebiger Stelle unterbrechbar.[36] Dabei ist die Interaktion mit dem System durchaus nicht umständlich, da die angebotenen Defaults dem Benutzer häufig ausreichen. Nur in selteneren Fällen entsteht ein Mehraufwand, der sich aber meist auf einen zusätzlichen Mausklick beschränkt. Dieser potentielle Mehraufwand wird aber dadurch mehr als ausgeglichen, daß die Dialogsteuerung unter der Kontrolle des Benutzers bleibt.

6.5 Funktionalität des Wissenseditors ZOO

In diesem Abschnitt wird die Funktionalität des Wissenseditors ZOO beschrieben. Es werden die einzelnen Funktionen zur Untersuchung von Wissensbasen und zum Erwerb von Wissen an einem Beispiel verdeutlicht. Im einzelnen wird auf die folgenden Funktionen des Metasystems ZOO eingegangen werden:

1. Visualisieren von Wissensbasen
2. Visualisieren von Wissensbasisobjekten durch Piktogramme
3. Navigieren entlang von Relationen zwischen Wissensbasisobjekten
4. Verändern des graphischen Erscheinungsbilds von Objekten
5. Editieren von Wissensbasisobjekten mittels Formularen
6. Graphische Manipulation von Relationen zwischen Objekten
7. Kopieren und Löschen von Objekten
8. Bearbeiten von konzeptuellen Wissensbasen
9. Erzeugen von Objekten aus Klassen
10. Definition neuer Slotbeschreibungen

Dabei soll von einem Szenario aus dem Bereich der Bürokommunikation ausgegangen werden. Wir werden mit Hilfe des Systems ZOO die Wissensbasen untersuchen, die einem Anwendungssystem aus dem Bürobereich zugrundeliegen. Darüber hinaus werden wir diese Wissensbasen erweitern und dadurch das Anwendungssystem an neue Anforderungen anpassen.

Wir betrachten den Vorgang des Schreibens, Versendens, Empfangens und Lesens von Briefen, der durch ein elektronisches Kommunikationssystem unterstützt wird. Ein Beispiel für die Funktionalität eines derartigen Anwendungs-

---

[36] Diese Art von Unterbrechbarkeit ist mehr als die bloße Unterbrechbarkeit von Betriebssystemprozessen. Die Besonderheit besteht darin, daß die mentale Belastung niedrig gehalten ist, die für den Benutzer entsteht, wenn er die Interaktion mit dem System unterbricht und sich kurzfristig einer anderen Tätigkeit zuwendet.

systems ist das UNIX-Mailsystem [UNIX MAIL 83], allerdings ist dieses System in herkömmlicher Weise mit Hilfe der prozeduralen Programmiersprache C [Kernighan, Ritchie 83] programmiert. Für unseren Zweck ist es jedoch wesentlich, daß das Anwendungssystem wissensbasiert mit Hilfe der objektorientierten Programmiersprache ObjTalk aufgebaut ist. Wir gehen deshalb von einem Postsystem aus, in dem das gesamte Anwendungswissen über die Teilnehmer des Postverkehrs, über Briefe und Postfächer usw. in Form von ObjTalk-Objekten repräsentiert ist. Da es uns nur darum geht, die Funktionsweise des Systems ZOO zu demonstrieren, genügt uns ein Anwendungssystem mit geringem Leistungsumfang.[37] Im folgenden wird gezeigt werden, wie die Fakten und Konzepte aus dem Bereich des Versendens elektronischer Post, die in den Wissensbasen eines derartigen Postsystems repräsentiert sind, mit Hilfe des Systems ZOO inspiziert und modifiziert werden können.

6.5.1 Visualisieren von Wissensbasen

Eine Hauptfunktion des Systems ZOO besteht darin, einen Benutzer beim Untersuchen von Wissensbasen zu unterstützen. Dies geschieht dadurch, daß die Wissensbasis im bereits beschriebenen graphischen Format auf einem ZOO-Window dargestellt wird. Am einfachsten ist die Visualisierung von Wissensbasen, die mit dem System ZOO aufgebaut oder bearbeitet worden sind: Die Darstellungsinformation ist bereits vorhanden und die Wissensbasis läßt sich jederzeit auf einem Bildschirmfenster des Systems ZOO betrachten. Andere Wissensbasen, denen diese Darstellungsinformation fehlt, lassen sich ebenfalls visualisieren, nur erfolgt die Visualisierung nicht vollautomatisch, sondern es muß dazu der Benutzer des Systems einbezogen werden.

Am Beispiel des elektronischen Postsystems lassen sich zwei Arten von Anwendungswissen aufzeigen, die sich zwei verschiedenen Wissensbasen zuordnen lassen:

- Die Wissensbasis *Post-Konzepte* enthält das konzeptuelle Wissen, das die Bedeutung der Begriffe *Brief*, *Teilnehmer*, *Postfach* usw. beschreibt. Dieses Wissen ändert sich bei der normalen Systembenutzung nicht und hat über

---

[37] Der geringe Leistungsumfang des Systems steht nicht im Widerspruch zur Forderung nach Wissensbasiertheit. Wie in Abschnitt 3.3.3 eingeführt, wird der Begriff "wissensbasiertes System" hier in einem technischen Sinn verwendet. Wissensbasierte Systeme sind Systeme, deren Funktionsweise auf Wissensbasen beruht. Wenn die Wissensbasen geringe Komplexität aufweisen, reduziert sich die Funktionalität des Anwendungssystems, das dabei aber nichtsdestoweniger wissensbasiert ist.

viele Sitzungen hinweg Gültigkeit. Die Wissensbasis Post-Konzepte ist vom Systementwerfer mit Hilfe des Systems ZOO aufgebaut worden, sie liegt also bereits in einer graphisch aufbereiteten Form vor. Wir werden auf diese Wissensbasis später zurückkommen, wenn gezeigt werden soll, wie mit dem System ZOO konzeptuelle Wissensbasen aufgebaut und modifiziert werden können.

- In der Wissensbasis *Post-Fakten* ist der aktuelle Zustand des Anwendungsgebiets repräsentiert. Dort sind die dem System bekannten Briefe, Postteilnehmer und Postfächer dargestellt und deren Relationen zueinander beschrieben. Diese Wissensbasis wird durch die normale Benutzung des Postsystems aufgebaut und fortgeschrieben. Durch das Editieren von Briefen mit einem üblichen Texteditor und das Versenden und Verwalten von Briefen über eine menü-, formular- oder kommandobasierte Schnittstelle werden in der Faktenwissensbasis Objekte angelegt und Slots gefüllt, welche den aktuellen Zustand des Anwendungsgebiets beschreiben. Für diese Objekte ist keine für das System ZOO verwertbare Darstellungsinformation angelegt. Im folgenden wird gezeigt werden, wie sich solche Objekte aber mit Hilfe des Systems ZOO in einer graphischen Darstellung inspizieren lassen.

Der Vorgang der erstmaligen Visualisierung von Wissensbasen wird durch Defaults weitgehend unterstützt. Die zur Darstellung von Objekten verwendeten graphischen Symbole werden vom System ermittelt, diese Zuordnung kann aber vom Benutzer verändert werden. Die Piktogramme erscheinen an einer Defaultposition und müssen noch vom Benutzer an einen geeigneten Platz verschoben werden. Eine vollautomatische Berechnung des Layouts wurde bisher noch nicht versucht, da die ästhetischen und semantischen Kriterien, die ein Mensch an eine solche Informationsanordnung anlegt, erst noch erkundet werden müssen.

Zur Visualisierung einer Wissensbasis wird ein ZOO-Window benötigt. Beim Laden des Systems ZOO wird ein solches Fenster automatisch erzeugt (Abbildung 6-9). Wenn mehrere Wissensbasen visualisiert werden sollen, kann der Systembenutzer über die *CREATE*-Funktion des Fenstersystems noch weitere ZOO-Windows erzeugen.

Im System ZOO ist eine einfache Wissensbasisverwaltung integriert. Diese wird aktiviert durch das Anklicken des Funktionssymbols *KB*[38] am Rand eines ZOO-Windows. In dem Verwaltungssystem sind Funktionen zum Selektieren, Abspeichern und Löschen von Wissensbasen vorgesehen.

---

[38]Die Abkürzung KB steht für die Wortverbindung "knowledge base"

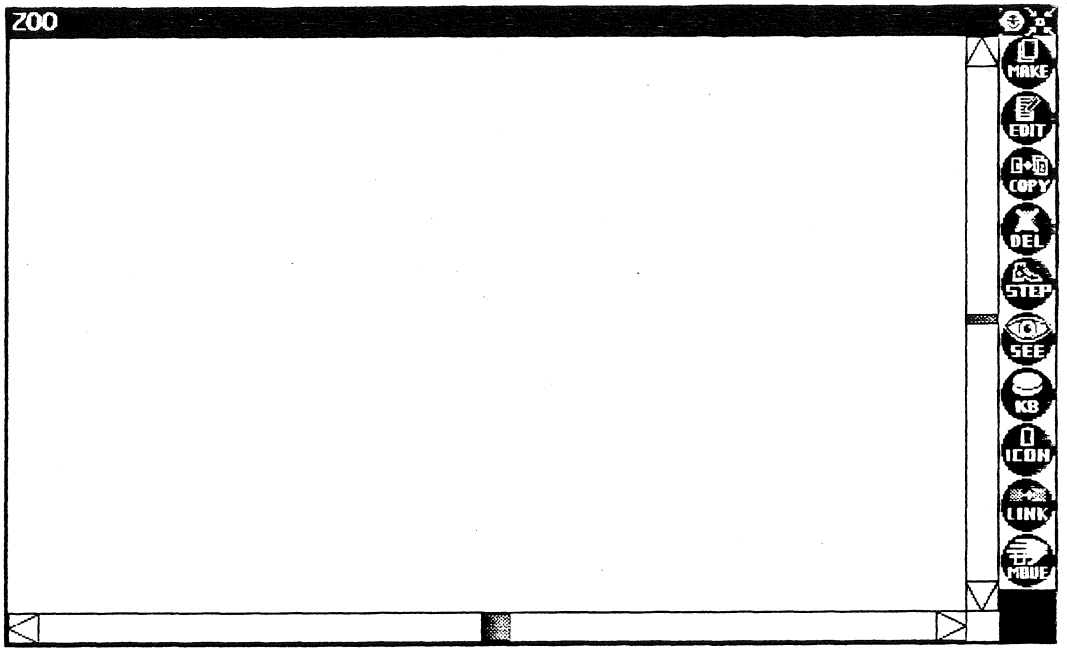


Abbildung 6-9: Ein leeres ZOO-Window

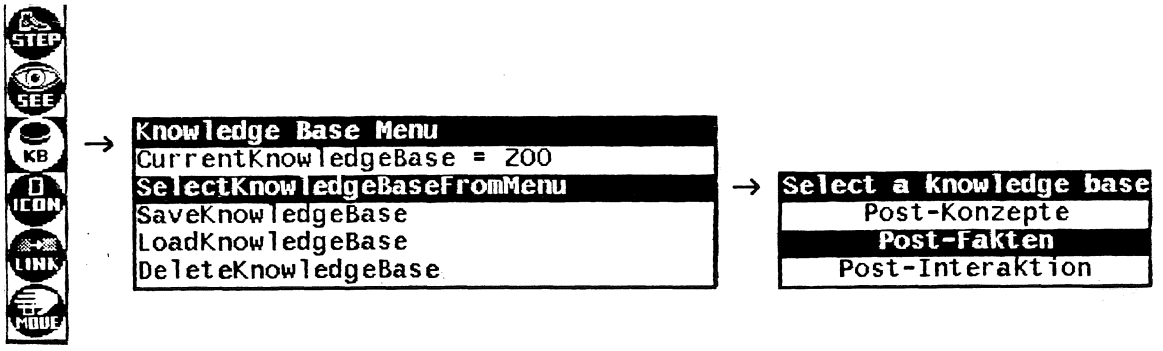


Abbildung 6-10: Selektion der Wissensbasis *Post-Fakten*

Mit Hilfe der Anwenderfunktion *KB* erhalten wir ein Sheet, also eine Mischform aus Menü und Formular, die als Benutzerschnittstelle zur Wissensbasisverwaltung dient (Abbildung 6-10). Wir rufen die Funktion SelectKnowledgeBaseFromMenu auf und erhalten ein Menü der im System vorhandenen Wissensbasen. Nach der Auswahl der Wissensbasis *Post-Fakten* ist das ZOO-Window darauf eingestellt, das Faktenwissen aus unserem Anwendungsgebiet darzustellen. Der Name der Wissensbasis erscheint im Titel des Fensters (siehe Abbildung 6-11).

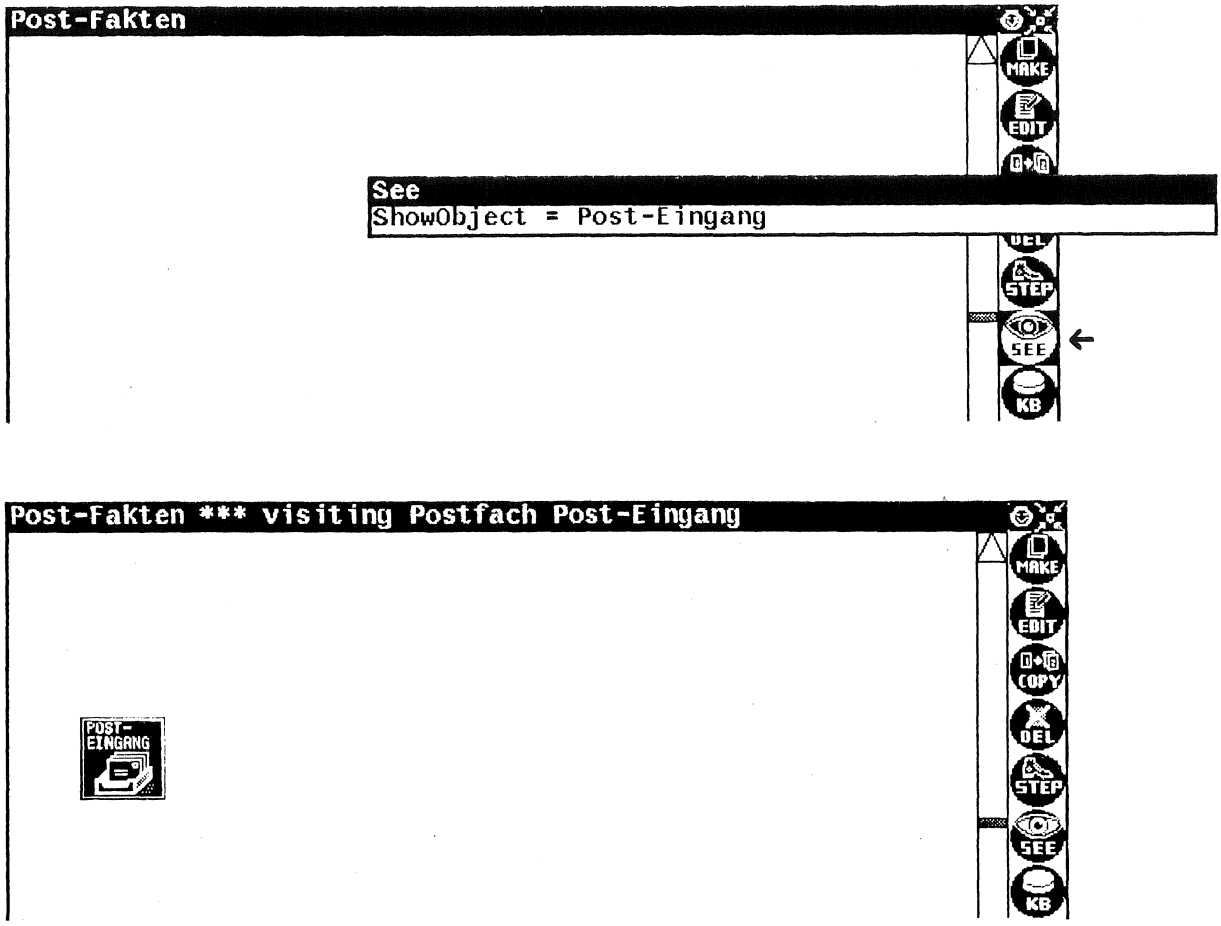


Abbildung 6-11: Visualisieren des Objekts *Post-Eingang*

6.5.2 Visualisieren von Wissensbasisobjekten durch Piktogramme

Im System ZOO kann man auf zwei Arten erreichen, Objekte in Form von Piktogrammen zu besichtigen. Die erste Art hat den Charakter eines Direktzugriffs, der eine beliebige Stelle im Netz der Wissensbasisobjekte zum Ziel haben kann. Die zweite Art, ein Objekt zu visualisieren, beruht auf einem sogenannten Navigationsvorgang. Der Vorgang der Navigation soll im nächsten Abschnitt beschrieben werden.

Für den Direktzugriff auf ein Objekt steht die Anwendungsfunktion *SEE* zur Verfügung. Nach Aufruf der Funktion *SEE* erscheint ein Sheet, mit dessen Hilfe das Objekt bezeichnet werden kann, das als nächstes graphisch dargestellt werden soll. Über den Sheet-Eintrag *ShowObject* = ist es möglich, den Namen des zu visualisierenden Objekts anzugeben. Voraussetzung hierfür ist allerdings,

daß das fragliche Objekt einen eindeutigen Namen besitzt und der Benutzer diesen auch kennt. Eine andere Möglichkeit der direkten Bezeichnung eines Objekts ist die Auswahl aus einem Menü. Abhängig vom Dialogkontext können nach Aufruf der Funktion *SEE* derartige Auswahlen ebenfalls angeboten werden. Sobald das Objekt eindeutig bezeichnet ist, wird es innerhalb des ZOO-Windows in Form eines Piktogramms sichtbar. Das Piktogramm ist dabei als selektiert gekennzeichnet. Mittels der Anwendungsfunktion *MOVE* kann das Piktogramm an eine beliebige Position verschoben werden.

Der Benutzer unseres Postsystems besitzt ein (elektronisches) Postfach mit dem Namen *Post-Eingang*, das die eingehende Post enthält. Mit Hilfe der Funktion *SEE* und nach der Angabe des Namens *Post-Eingang* taucht ein Piktogramm auf dem Bildschirm auf, das ein Ablagefach verbildlicht und mit dem Namen "Post-Eingang" beschriftet ist (Abbildung 6-11). Das Piktogramm erscheint in Inversdarstellung, was bedeutet, daß es selektiert ist, und damit Gegenstand aller objektspezifischen Operationen ist.

6.5.3 Navigieren entlang von Relationen zwischen Wissensbasisobjekten

Eine andere Art der Visualisierung beruht auf dem Vorgang der Navigation durch das Netz der Wissensbasisobjekte. Sie besteht darin, daß man von einem bereits dargestellten Knoten des Wissensnetzes zu einem benachbarten Knoten übergeht, der als nächster auf dem Bildschirm sichtbar gemacht werden soll. Zum Zweck der Navigation bietet das System ZOO die Anwendungsfunktion *STEP* an. Um die Relationen zu untersuchen, in denen das Objekt Post-Eingang zu anderen Objekten steht, lösen wir die Anwendungsfunktion *STEP* aus. Es erscheint ein Menü aller Slots des Objekts *Post-Eingang*. Wir wählen den Slot mit dem Namen "Inhalt" aus. Die Folge ist eine "Navigation" zu einer Bildschirmregion, die den Slot *Inhalt* des Objekts *Post-Eingang* darstellt: Sichtbar wird ein Textfeld, bestehend aus dem Schriftzug "Inhalt", das durch eine gerade Linie mit der graphischen Darstellung des Post-Eingangs verbunden ist. Dabei ist nun das Textfeld als selektiert gekennzeichnet und steht im Mittelpunkt der Betrachtung (Abbildung 6-12 oben). Auch dieses Bildschirmobjekt können wir mit Hilfe der Anwendungsfunktion *MOVE* verschieben; während des Verschiebevorgangs verhält sich die Verbindungslinie zum Post-Eingang-Piktogramm wie ein gespannter Gummifaden, der stets die direkte Verbindung zwischen seinen Endpunkten anzeigt.

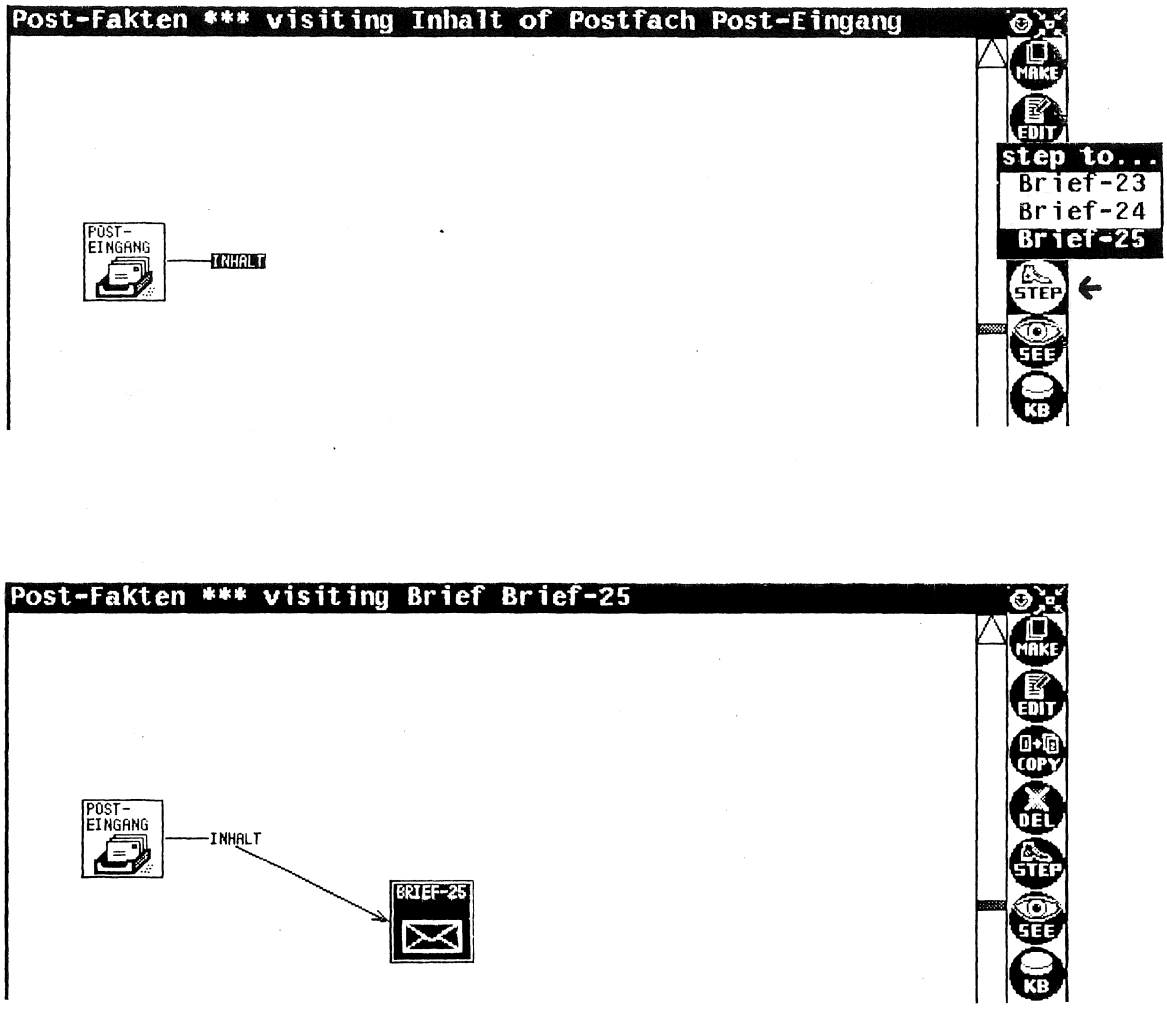


Abbildung 6-12: Navigation zum Objekt *Brief-25* entlang der Relation *Inhalt*

Ein weiterer Aufruf von *STEP* bringt ein Menü aller Briefe im Posteingang zum Vorschein (Abbildung 6-12 oben rechts). Die Auswahl eines Menüeintrags, beispielsweise *Brief-25* bewirkt, daß der betreffende Brief visualisiert wird. Der Brief zeigt sich als Piktogramm, welches das stilisierte Abbild eines Briefes und die Inschrift *Brief-25* trägt. Außerdem erscheint ein Pfeil, welcher das Textfeld *Inhalt* mit dem Piktogramm von *Brief-25* verbindet (Abbildung 6-12 unten).

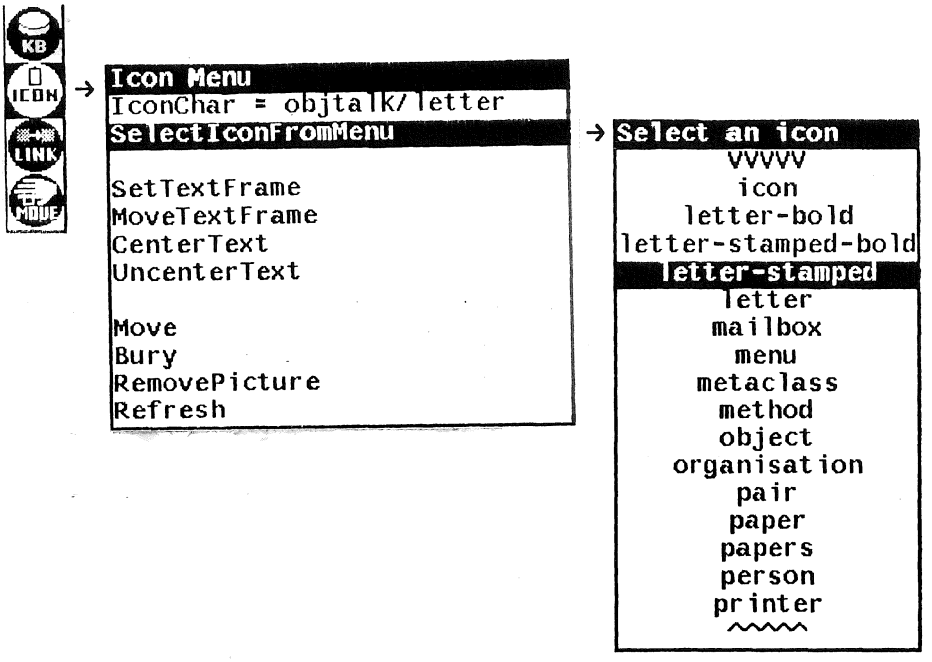


Abbildung 6-13: Dem Objekt *Brief-25* wird das Piktogramm *letter-stamped* zugeordnet

6.5.4 Verändern des graphischen Erscheinungsbilds von Objekten

Das graphische Erscheinungsbild eines Objekts wird vom System ZOO per Default von der graphischen Darstellung der zugehörigen Klasse abgeleitet. Mit Hilfe der Anwenderfunktion *ICON* kann aber die graphische Erscheinung des selektierten Objekts verändert werden.

Das Objekt *Brief-25* besitzt in seiner ZOO-Darstellung ein stilisiertes graphisches Symbol eines Briefes. Dieses Symbol wurde vom System ZOO defaultmäßig vergeben, ohne beim Benutzer rückzufragen. Es wurde aus der Darstellungsinformation der Klasse *Brief* abgeleitet. Offenkundig war das Symbol beim Entwurf der konzeptuellen Wissensbasis *Post-Konzepte* der Klasse Brief zugeordnet worden.

Wir wollen nun das graphische Symbol des Objekts *Brief-25* verändern. Dazu lösen wir die Anwenderfunktion *ICON* aus. Es erscheint das Sheet "Icon Menu", in dem die Manipulationsmöglichkeiten angeboten werden (Abbildung 6-13). Bestimmte Funktionen ermöglichen das Verschieben und Zentrieren des

Textfeldes. Aus dem obersten Eintrag des Sheets geht hervor, daß das verwendete graphische Symbol den Namen "letter" besitzt. Nach Aufruf der Funktion *SelectIconFromMenu* erhalten wir ein Menü aller bereits definierten graphischen Symbole. Wir wählen dort den Namen "letter-stamped" aus. Das bewirkt eine unmittelbare Veränderung des graphischen Symbols auf dem Piktogramm des Briefes. Zuvor war dort die Rückseite eines Briefes dargestellt, nun ist dort eine Brief-Vorderseite sichtbar (siehe Abbildung 6-14).

Neben der Auswahl aus vorhandenen graphischen Objekten besteht auch die Möglichkeit, neue graphische Objekte zu definieren. Dies geschieht mit Hilfe eines Piktogramm-Editors [Maier 85].

6.5.5 Editieren von Wissensbasisobjekten mittels Formularen

Nicht alle Merkmale von Objekten haben den Charakter von Relationen zu anderen Objekten und lassen sich mit Hilfe von beschrifteten Pfeilen zwischen Piktogrammen darstellen. Es gibt auch Objekteigenschaften, die sich lediglich in textueller Form darstellen lassen. Für einen solchen Fall sieht das System ZOO eine Formulardarstellung für Objekte vor. Nach Auslösen der Anwenderfunktion *EDIT* erscheint auf dem Bildschirm ein Formular, das die Merkmale des selektierten Objekts in textueller Form zeigt. Diese Merkmale können editiert werden: sie lassen sich mit Hilfe von Maus und Tastatur überschreiben.

In unserem Beispiel ist das Objekt *Brief-25* selektiert. Nach dem Anklicken des Funktionssymbols *EDIT* werden die Eigenschaften des Briefs in einem Formular dargestellt (Abbildung 6-14). Der *Betreff* des Briefes ist ein einfaches Attribut, das sich gut im Formular darstellen läßt. Der *Inhalt* des Briefes ist zu groß, um sinnvoll im Formular dargestellt zu werden; er erscheint deshalb in abgekürzter Form. Um ihn zu ändern, müßte ein Texteditor aufgerufen werden. *Datum*, *Absender* und *Adressaten* des Briefes sind ObjTalk-Objekte bzw. Listen von Objekten. Objekte werden in den Formularfeldern durch ihren Namen dargestellt; eine besondere Kennzeichnung (durch die Zeichen #.) macht deutlich, daß hinter diesen Namen Objekte stehen.

Wenn wir jetzt mit der Maus die Betreff-Zeile im Formular anklicken, können wir den *Betreff* des Briefs mit einem anderen Text überschreiben. Diese Veränderung würde sichtbar werden, wenn wir später den Brief im Postsystem betrachten. Da wir aber an dem Objekt nichts ändern wollen, klicken wir ein-

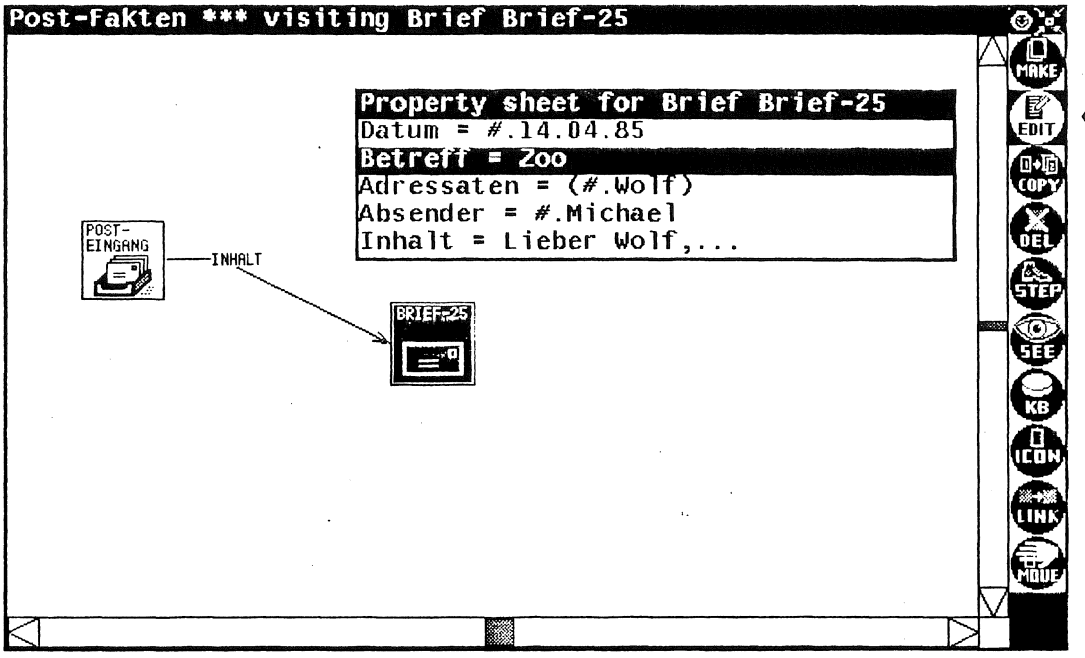


Abbildung 6-14: Formulardarstellung des Objekts *Brief-25*

fach mit der Maus auf einen Bereich außerhalb des Formulars. Das Formular verschwindet und der Editiervorgang ist ohne Folgen abgebrochen.

6.5.6 Graphische Manipulation von Relationen zwischen Objekten

Relationen zwischen Objekten können durch direkte Manipulation verändert werden. Es ist möglich, Pfeilverbindungen zwischen den Bildschirmrepräsentationen von Slots und Objekten durch Zeigeaktionen herzustellen und zu lösen. Für diesen Zweck hält das System ZOO die Anwenderfunktion *LINK* bereit.[39]

Als Ergebnis mehrerer Navigationsaktionen, die wir überspringen wollen, weil sie uns keine neue Funktionalität demonstrieren, zeigt sich die Faktenwissensbasis Post-Fakten wie in Abbildung 6-15 oben. Sichtbar ist, daß die Person *Wolf* zu den Adressaten des Briefs *Brief-25* gehört und daß als Absender des Briefs eine Person namens *Michael* fungiert. Um zu zeigen, wie diese Relationen mit Hilfe von ZOO verändert werden können, wollen wir nun die Rollen der beiden

[39]Da diese Funktion sowohl zum Herstellen von bisher nicht bestehenden Relationen als auch zum Lösen von bestehenden Relationen verwendet werden kann, müßte sie eigentlich den Namen "LINK/UNLINK" oder "TOGGLE-LINK" tragen.

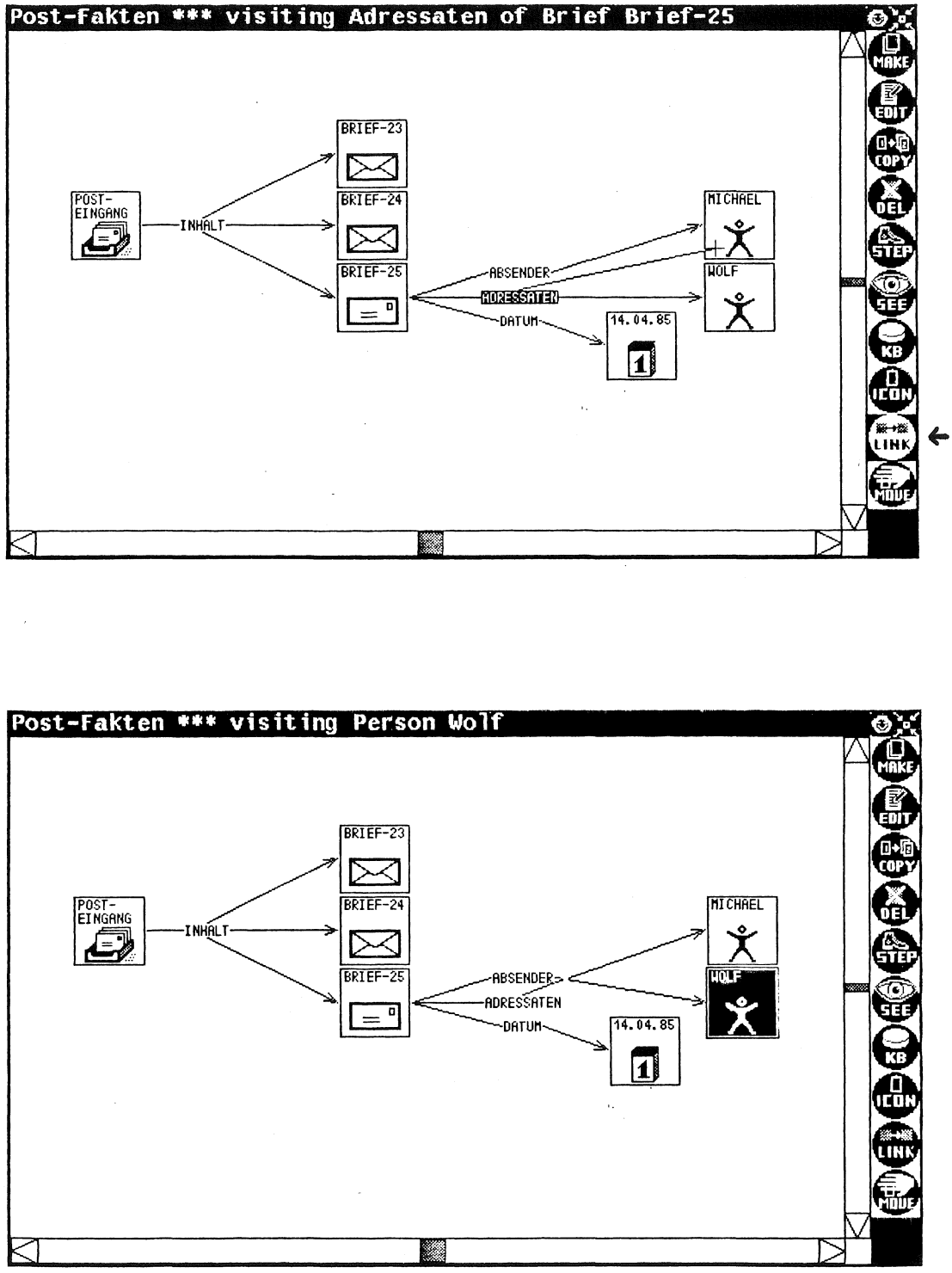


Abbildung 6-15: Vertauschen von *Adressat* und *Absender* des Objekts *Brief-25*

Personen als Absender und Adressat des Briefes auf graphischem Weg vertauschen.

Aktuell selektiert ist der Slot *Adressaten*. Wir klicken das Funktionssymbol *LINK* an und erhalten einen von dem Bild des Slots ausgehenden "Gummifaden", dessen Ende den Mausbewegungen folgt. Wenn wir nun auf die Bildschirmrepräsentation von *Michael* klicken, wird Michael auch zu einem Adressaten des Briefes, von der Bildschirmdarstellung dieses Slots gehen nun zwei Pfeile aus. Offenkundig kann der Slot Adressaten multiple Werte annehmen.

Um Wolf aus der Adressatenliste zu entfernen, gehen wir genau gleich vor: Zwei Zeigeaktionen, erst auf das Funktionssymbol *LINK* und dann auf die Person *Wolf*, lösen die bestehende Relation von Wolf als Adressat des Briefes. Wenn wir nun Wolf selektieren und mit einem weiteren Aufruf von *LINK* mit dem *Absender* des Briefs verbinden, wird zum einen Wolf als Absender des Briefs eingetragen, und zum andern, da es in unserem Postsystem immer nur einen Absender eines Briefes geben kann, wird Michael automatisch als Absender ausgetragen und der entsprechende Pfeil verschwindet. Das Resultat unserer Manipulationen ist in Abbildung 6-15 unten zu sehen.

Wenn wir bei späterer Benutzung des Postsystems den Brief versenden wollten, würde er an seinen ursprünglichen Absender *Michael* zurückgeschickt werden.

6.5.7 Kopieren und Löschen von Objekten

Durch Kopieren können neue Objekte erzeugt werden. Zu diesem Zweck steht die Funktion *COPY* bereit. Zum Löschen von Objekten gibt es die Funktion *DELETE*:

Selektieren der Person *Wolf* und anschließendes Auslösen der Funktion *COPY* bewirkt, daß auf dem Schirm das Piktogramm einer weiteren Person erscheint, das mit dem Namen *<some—Person>* beschriftet ist. Nach dem Aufruf von *MAKE* können wir der erzeugten Person einen Namen geben, z.B. *Thomas*. Unserem Postsystem ist so der Name eines weiteren Postteilnehmers bekannt geworden (Abbildung 6-16). Mit der *DELETE*-Funktion können wir das erzeugte Objekt *Thomas* wieder aus der Wissensbasis entfernen (Abbildung 6-17).

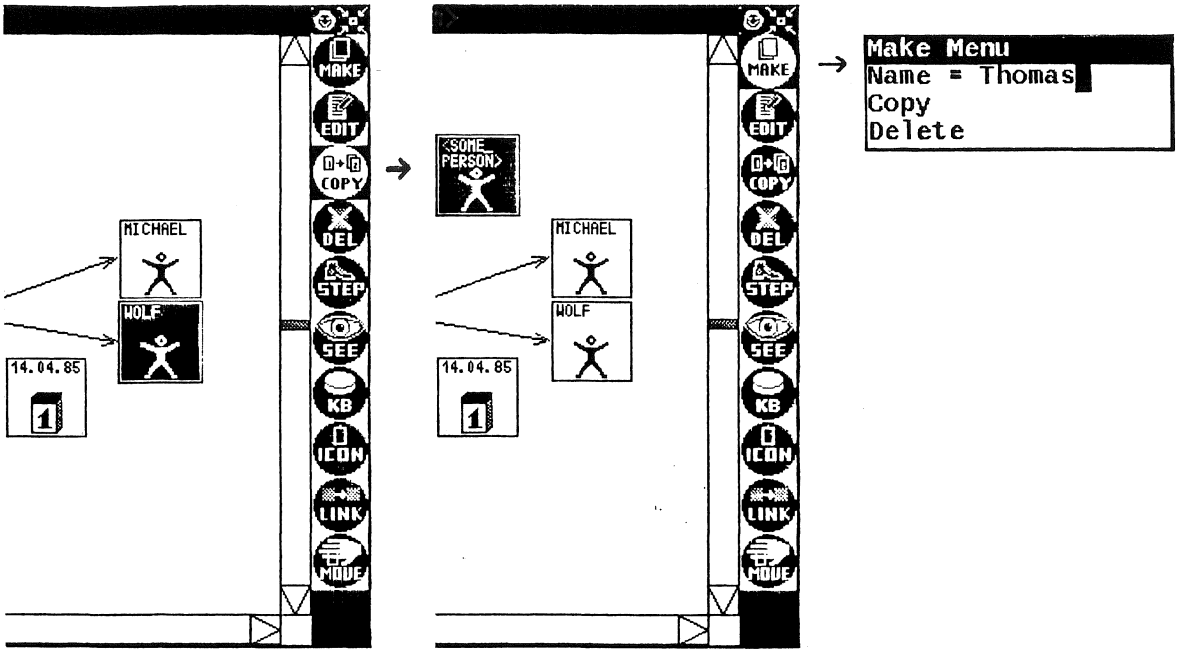


Abbildung 6-16: Durch Kopieren wird die Person *Thomas* erzeugt

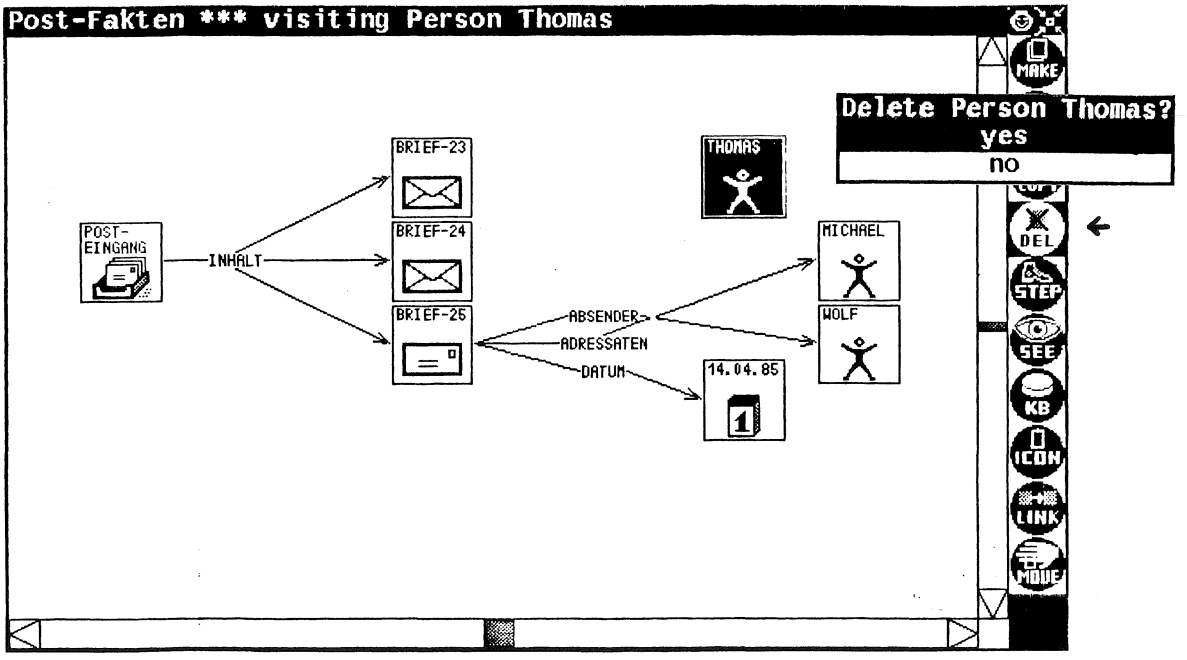


Abbildung 6-17: Die Person *Thomas* wird aus der Wissensbasis gelöscht

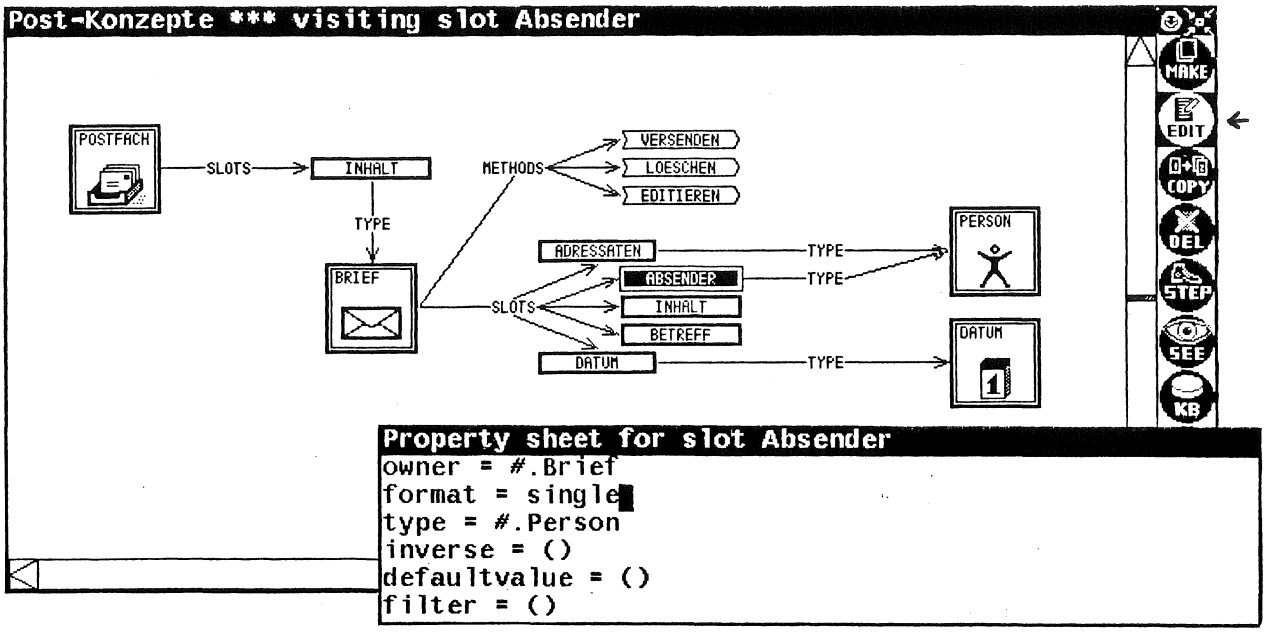


Abbildung 6-18: Darstellung der konzeptuellen Wissensbasis *Post-Konzepte*

6.5.8 Bearbeiten von konzeptuellen Wissensbasen

Genauso wie Faktenwissensbasen können auch konzeptuelle Wissensbasen visualisiert werden. Konzeptuelle Wissensbasen enthalten Klassen, Slotbeschreibungen, Methoden, Regeln usw. sowie deren Relationen untereinander. Konzeptuelle Wissensbasen spielen die Rolle von "Programmen" in herkömmlichen Systeme, sie werden bei der normalen Benutzung des Systems nicht angetastet. Nur wenn das Verhalten des Anwendungssystems verändert werden soll, wird man die konzeptuelle Wissensbasis modifizieren wollen. Mit dem Anwendungssystem ist dies normalerweise[40] nicht möglich, für diese Wissenserwerbsaufgabe bietet sich der ZOO Wissenseditor an.

Um die konzeptuelle Wissensbasis unseres Postsystems zu betrachten, erzeugen wir mit Hilfe der *CREATE*-Funktion des Fenstersystems[41] ein neues ZOO-Window und lassen uns über die Funktion *KB* die Wissensbasis *Post-Konzepte* zeigen. Diese Wissensbasis ist schon einmal mit dem ZOO Wissenseditor bear-

[40] Eine Ausnahme bilden Systeme, die über eine Metakomponente verfügen, wie die bereits in früheren Kapiteln genannten Systeme D&I oder MYCIN

[41] diese ist beschrieben in [Fabian 84]

beitet worden; denn die Darstellungsinformation ist bereits vorhanden, so daß die Wissensbasis wie in Abbildung 6-18 erscheint. In der graphischen Darstellung erkennt man, in welchen Beziehungen die Konzepte des Postsystems zueinander stehen. Sichtbar sind die Klassen *Postfach*, *Brief*, *Person* und *Datum*, sowie deren Slotbeschreibungen *Inhalt*, *Adressaten*, *Absender* usw. Des weiteren werden die Methoden visualisiert, die für Briefe verfügbar sind, nämlich *Versenden*, *Löschen* und *Editieren*.

Wir erkennen, daß der *Absender* eines Briefs vom Typ *Person* sein muß. Wenn wir die Slotbeschreibung *Absender* selektieren und die Funktion *EDIT* aufrufen, werden in einer Formulardarstellung weitere Einzelheiten erkennbar (Abbildung 6-18 rechts unten): Beispielsweise hat der Absender eines Briefes das *format* "single", was bedeutet, daß nur eine einzelne Person als Absender eines Briefes möglich ist. Wenn wir den Eintrag in "list" ändern, würde es in unserem Postsystem künftig möglich sein, auch mehrere Personen als Absender eines Briefes einzutragen.

6.5.9 Erzeugen von Objekten aus Klassen

Es gibt verschiedene Möglichkeiten, um aus Klassen neue Objekte zu erzeugen. Wenn im System ZOO eine Klasse selektiert ist, gibt es über die Funktion *COPY* die Möglichkeit, eine Klasse zu kopieren, sowie über die Funktion *MAKE* die Möglichkeit, Subklassen und Instanzen[42] der Klasse zu bilden.

Das System ZOO operiert hierbei wie auch sonst "am lebenden Objekt": Die mit Hilfe des Systems definierten Objekte werden sofort in die Wissensbasis eingetragen und sind unmittelbar verfügbar. Insbesondere kann eine im System ZOO erzeugte Klasse ohne weiteren Compilationsvorgang wiederum instantiiert werden.

Wir wollen die Funktionalität des Postsystems erweitern, indem wir die Möglichkeit vorsehen, Briefe in sogenannten *Ablagen* aufzubewahren. Eine Ablage soll etwas ähnliches sein, wie ein Postfach, daher wollen wir sie als eine Subklasse von *Postfach* definieren. Dazu selektieren wir die Klasse *Postfach* und rufen über das Symbol *MAKE* die Funktion *MakeSubclass* auf. Es entsteht in der Spezialisierungshierarchie eine Subklasse von *Postfach*. Wir können ih-

---

[42] Auch Klassen können durch Instantiieren gebildet werden, und zwar durch Instantiieren der Metaklasse class.

ren Namen in *Ablage* ändern und ihr Piktogramm an eine geeignete Stelle auf dem ZOO-Window "Post-Konzepte" plazieren. *Ablage* erbt als Subklasse von Postfach dessen graphisches Symbol (Abbildung 6-19 oben).

Von der Klasse *Ablage* können wir sofort eine Instanz machen. Mit Hilfe der Funktion MakeInstance (ebenfalls erreichbar über das Funktionssymbol *MAKE*) erzeugen wir eine Instanz von Ablage, der wir den Namen *Privat* geben. Dort sollen künftig Privatbriefe abgelegt werden (Abbildung 6-19 unten).

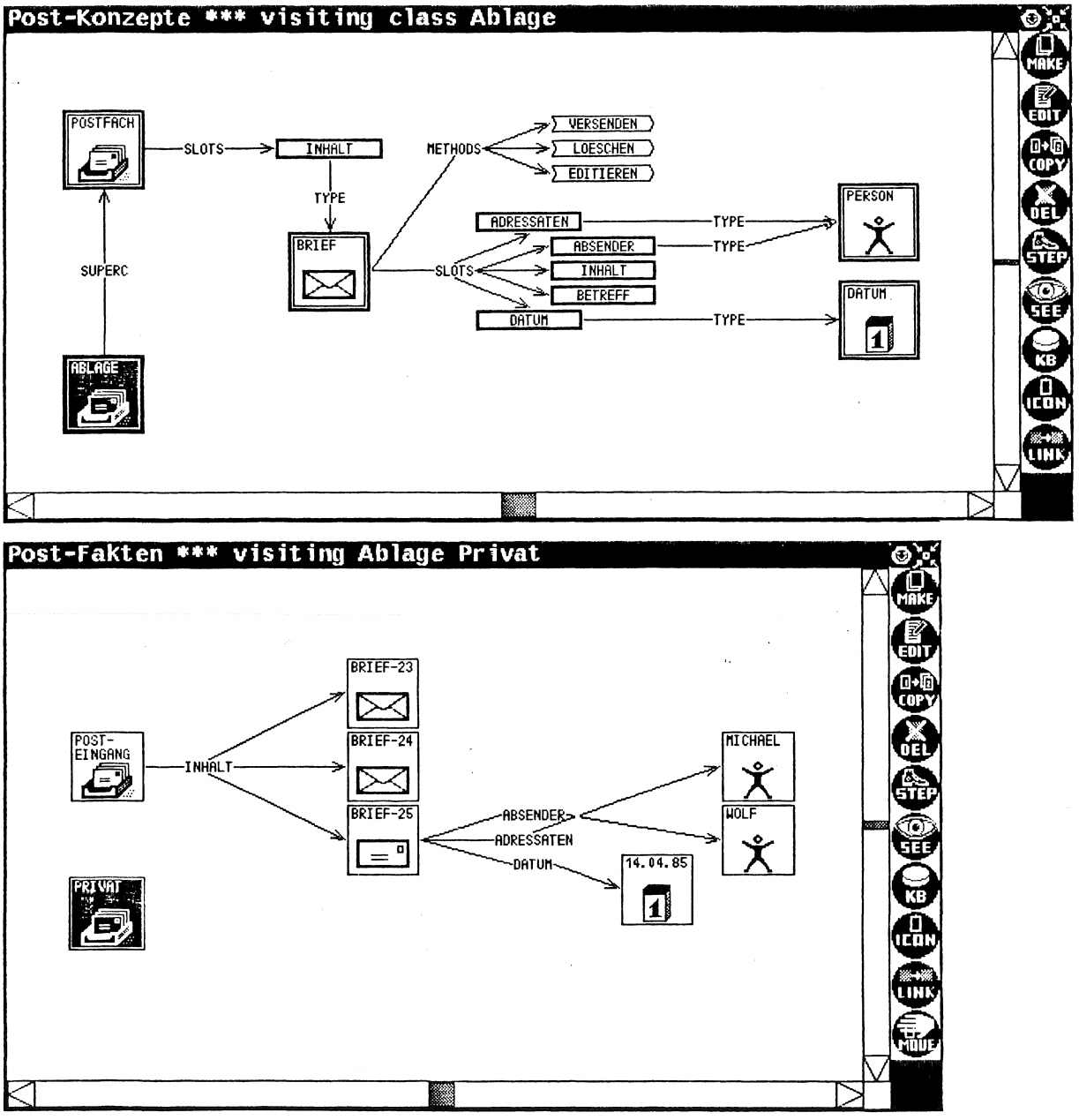


Abbildung 6-19: Die Wissensbasen nach dem Erzeugen der Klasse *Ablage* und deren Instanz *Privat*

6.5.10 Definition neuer Slotbeschreibungen

ObjTalk-Klassen können über Slotbeschreibungen verfügen. Diese legen fest, welche Slots eine Instanz der betreffenden Klasse besitzen kann. Wenn für eine Klasse von Instanzen ein neuer Slot benötigt wird, muß man der Klasse eine neue Slotbeschreibung zuordnen. Slotbeschreibungen können durch Kopieren[43] erzeugt werden.

Um in unserem Postsystem einen Brief einer Ablage zuordnen zu können, brauchen wir im Brief einen Slot, in welchen die zugeordnete Ablage eingetragen werden kann. Wir müssen eine Slotbeschreibung erzeugen, die der Klasse Brief zugeordnet ist. Eine derartige Slotbeschreibung können wir uns durch Kopieren einer vorhandenen erzeugen. Wir geben ihr den Namen "Abgelegt". Ab sofort besitzt jeder Brief im System den zusätzlichen Slot *Abgelegt*. Der Slot *Abgelegt* soll in einer Inversbeziehung zum Slot *Inhalt* einer Ablage stehen. Durch weiteres Kopieren von Slotbeschreibungen und durch Ziehen von Relationspfeilen gelangen wir zu einer neuen Beschreibung der konzeptuellen Welt, wie sie in Abbildung 6-20 oben zu sehen ist.

Wenn wir jetzt mit Hilfe der Funktion *STEP* das Menü der Relationen abrufen, die für den Brief *Brief-25* definiert sind, so erscheint dort jetzt auch der Name der Relation *Abgelegt* (Abbildung 6-20 unten). Wir können nun unseren *Brief-25* der Ablage *Privat* zuordnen. Da der Slot *Abgelegt* als invers zum Slot *Inhalt* von Ablage definiert ist, ist das System zur Bildung einer Inferenz fähig: Ohne weiteres Zutun des Benutzers wurde der *Brief-25* automatisch im *Inhalt* der Ablage *Privat* eingetragen. Das Ergebnis ist in Abbildung 6-21 zu betrachten.

Die Wissensnetze in einem ZOO-Window haben also eine Syntax. Es können nur solche Relationen zwischen Objekten hergestellt werden, die mittels entsprechender Slotbeschreibungen definiert sind. Die Syntax einer Faktenwissensbasis ist durch die zugeordnete konzeptuelle Wissensbasis gegeben.[44] Durch die Manipulation einer konzeptuellen Wissensbasis läßt sich die Syntax der Faktenwissensbasis mit unmittelbarer Wirkung verändern. Eine derartige syntaxverän-

---

[43]Eine zweite Art, Slotbeschreibungen zu erzeugen, besteht darin, die vordefinierte Klasse slot zu instantiieren.

[44]Dementsprechend ergibt sich die Syntax einer konzeptuellen Wissensbasis aus der übergeordneten Metawissensbasis.

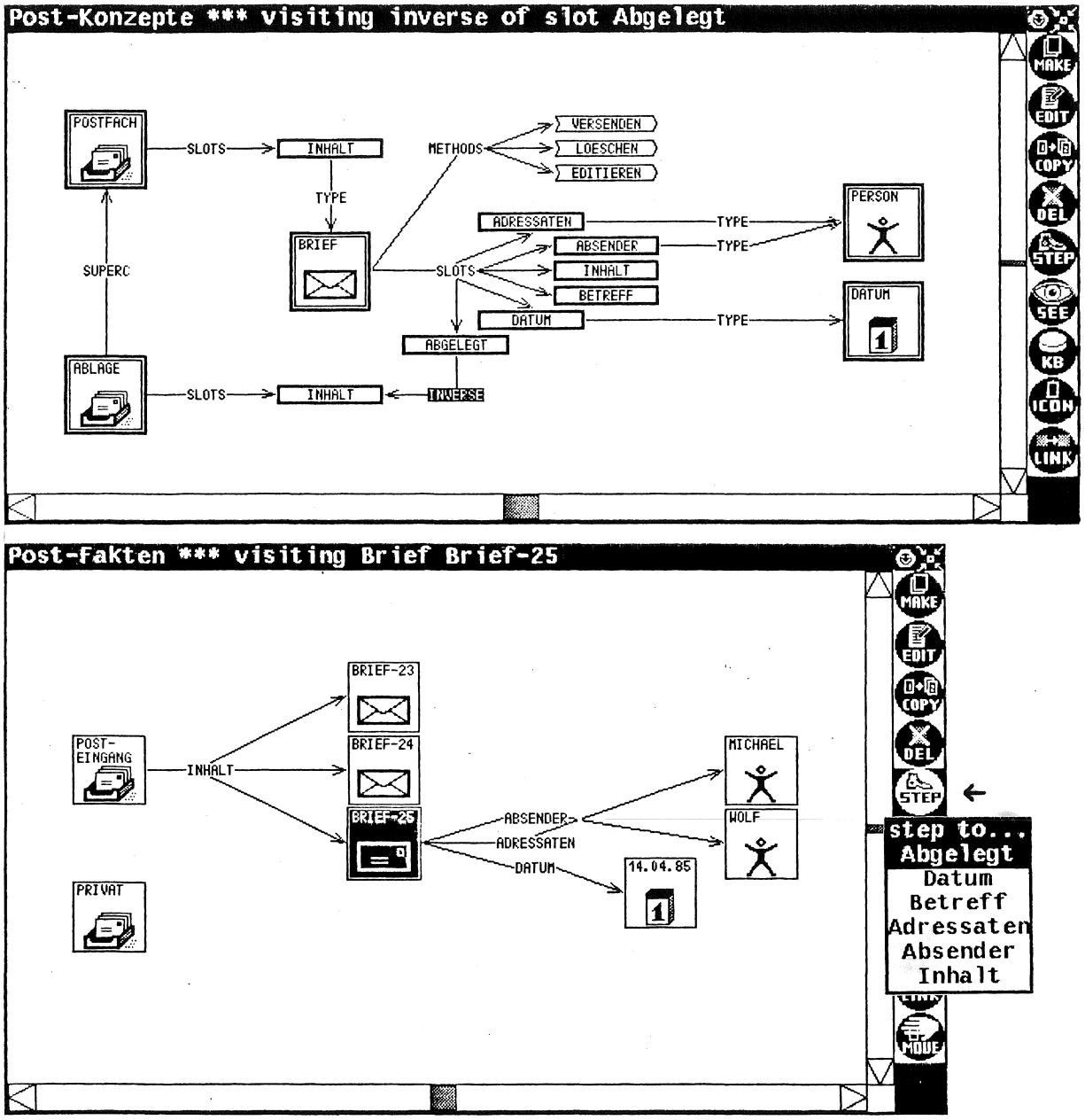


Abbildung 6-20: Die Wissensbasen nach dem Erzeugen der Slotbeschreibung *Abgelegt*

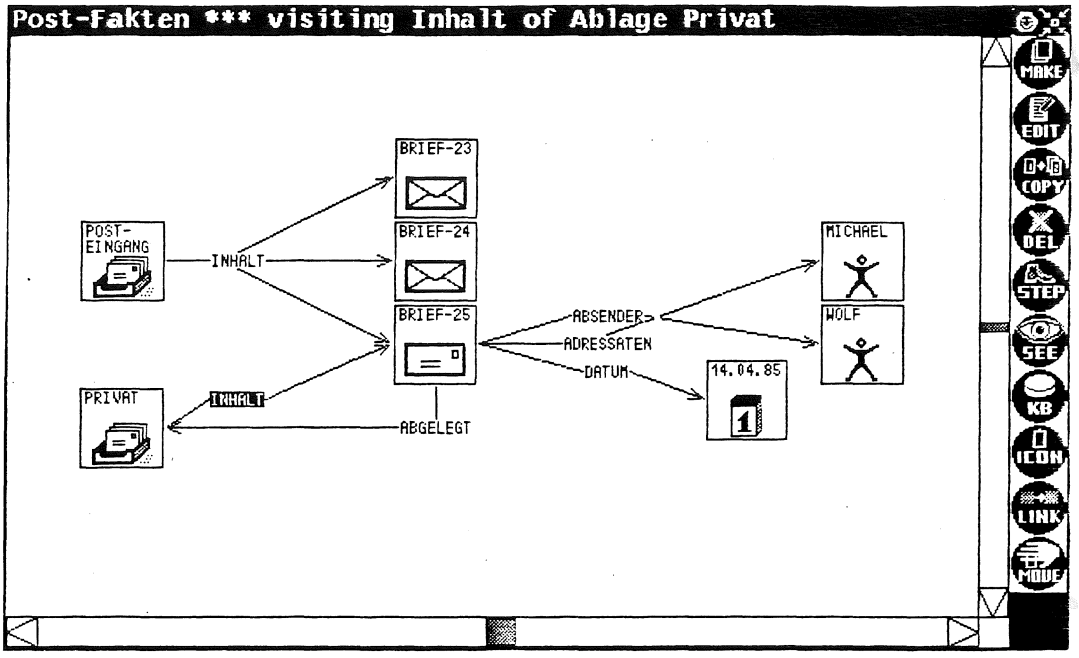


Abbildung 6-21: Der Brief *Brief-25* ist der Ablage *Privat* zugeordnet

dernde Manipulation einer konzeptuellen Wissensbasis ist von der Interaktionsform her nicht unterschiedlich zur Art der Bearbeitung von Faktenwissensbasen.

6.5.11 Folgerungen

Unsere Beispielsitzung ist nun beendet. Wir waren in der Lage, sowohl die Faktenwissensbasis *Post-Fakten* als auch die konzeptuelle Wissensbasis Post-Konzepte zu bearbeiten und dadurch das Verhalten eines Anwendungssystems zu verändern. Insbesondere die Änderungen im konzeptuellen Bereich wären bei einem nicht-wissensbasierten System (wie etwa UNIX-Mail) nur durch Programmänderungen möglich gewesen.

Was jetzt noch fehlt, ist die Anpassung der Benutzerschnittstelle des Systems. Beispielsweise wenn die Slots eines Briefes in unserem Postsystem durch Formularfelder dargestellt sind, muß jetzt ein weiteres Formularfeld definiert werden, in das die Ablage eines Briefes eingetragen werden kann. Die Liste aller vorhandenen Ablagen sowie die in einer Ablage vorhandenen Briefe könnten jeweils durch ein Menü dargestellt werden. Wenn die Benutzerschnittstelle ebenfalls wissensbasiert dargestellt ist, z.B. in Form einer Wissensbasis Post-Interaktion, kann wiederum das System ZOO eingesetzt werden, um die erforderlichen Formular- und Menüobjekte zu erzeugen.

Die Demonstration der Funktionen des Systems ZOO zeigt neue Möglichkeiten, die Fakten und Konzepte eines Anwendungssystems in einer anschaulichen Darstellungsform zu untersuchen. Modifikationen des Anwendungssystems erfolgen durch die direkte Manipulation von Graphiken; Programmierkenntnisse sind nicht erforderlich. Der Wissenseditor ZOO macht so beispielhaft deutlich, wie mit Hilfe von graphischen Schnittstellen zum Erwerb von Wissen und zur Untersuchung von Wissensbasen die Durchschaubarkeit und Adaptierbarkeit von Softwaresystemen für den Endbenutzer in entscheidender Weise erhöht werden kann.

Das System ZOO nutzt das bereits im Anwendungssystem vorhandene Wissen, um den Benutzer bei seiner Arbeitsaufgabe zu unterstützen: Das Untersuchen von Wissensbasen wird erleichtert durch Hilfen, die sich aus den vorhandenenen Wissensstrukturen ableiten, so etwa durch automatisch erzeugte Menüs zur Navigation und durch defaultmäßig vorgegebene Piktogramme für die darzustellenden Objekte. Der Vorgang des Wissenserwerbs erfolgt ebenfalls wissensbasiert: Das Erzeugen von Objekten und das Herstellen von Beziehungen zwischen Objekten wird unterstützt durch konzeptuelles Wissen. Vom System wird die Konstruktion von genau solchen Objekten und Relationen angeboten, die von Art und Syntax her dem vorhandenen konzeptuellen Wissen entsprechen.

6.6 ZOO-Wissensbasen

Die Verwaltung der graphischen Objekte eines ZOO-Windows geschieht nicht an einer zentralen Stelle, vielmehr erfolgt sie in objektorientierter Weise verteilt auf viele aktive Instanzen: Jedes Piktogramm, jeder Pfeil, jede Pfeilbeschriftung ist durch ein ObjTalk-Objekt repräsentiert. Diese ObjTalk-Objekte sollen hier einem Vorschlag von Herczeg folgend als *Interaktionsobjekte* bezeichnet werden [Herczeg 85a]. Interaktionsobjekte stellen die Kopplung her zwischen internen Wissensbasisobjekten und ihren graphischen Darstellungen auf dem Bildschirm. In den Interaktionsobjekten sind daher zwei Arten von Wissen repräsentiert:

- Interaktionsobjekte besitzen Darstellungswissen: Jedes Interaktionsobjekt beschreibt die Lage und Gestalt eines auf dem Bildschirm erscheinenden graphischen Objekts. Dieses Wissen ist gleichermaßen von Bedeutung für das optische Erscheinungsbild des Interaktionsobjekts wie für die Möglichkeit der Aktivierung desselben mit Hilfe eines Zeigeinstruments.

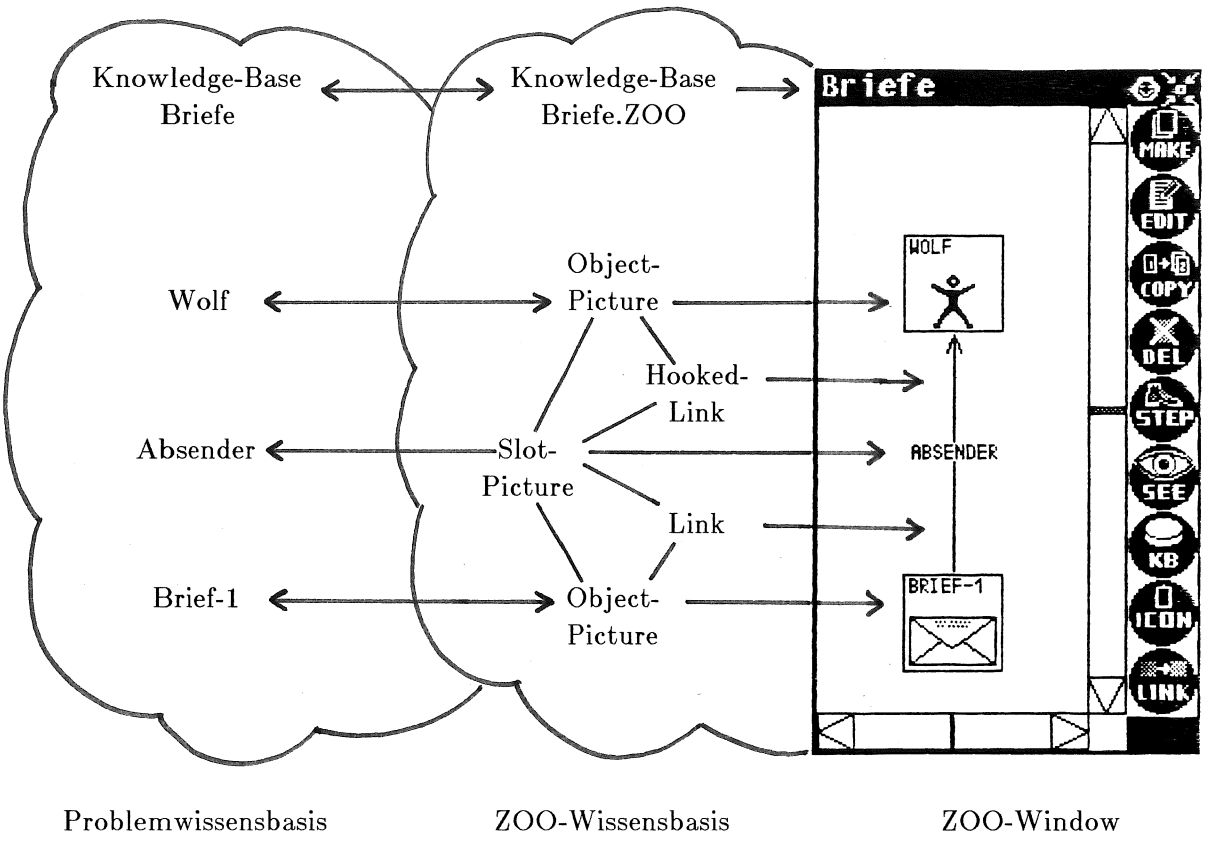


Abbildung 6-22: Problemwissensbasis, ZOO-Wissensbasis und ZOO-Window

- Interaktionsobjekte besitzen Wissen über Wissensbasisobjekte: Jedes Interaktionsobjekt weiß um die Existenz oder um die Eigenschaften des Wissensbasisobjekts, das es nach außen hin vertritt. Darüberhinaus stellen Interaktionsobjekte ein Medium dar zur Manipulation der internen Wissensbasisobjekte.

Die hauptsächliche Funktion des Systems ZOO, nämlich die Darstellung des Inhalts einer Wissensbasis mittels direkt manipulierbarer graphischer Objekte in einem Bildschirmfenster, läßt sich somit folgendermaßen darstellen: Die Aufgabe des Systems ZOO besteht darin, zu einer Wissensbasis über einen bestimmten Problemraum eine zweite Wissensbasis zu generieren, die aus Interaktionsobjekten besteht. Diese zweite Wissensbasis wird als ZOO-Wissensbasis bezeichnet. Eine solche ZOO-Wissensbasis stellt gleichermaßen eine Beschreibung der internen Objekte einer Problemwissensbasis und der externen graphischen Objekte auf einem ZOO-Window dar (siehe Abbildung 6-22).

Die im System ZOO zur Darstellung von Wissensbasen verwendeten Interaktionsobjekte sind im wesentlichen Spezialformen sogenannter Icons aus dem Icons-Package [Herczeg 85b]. Icons sind Interaktionsobjekte, die auf dem Bildschirm in der Regel als beschriftete Piktogramme erscheinen. Icons verfügen über eine sogenannte Aktion, die ausgeführt wird, wenn das Icon mit der Maus aktiviert wird, sowie über einen Pointer auf das von ihnen repräsentierte Objekt. Im Icons-Package gibt es eine Vielzahl von Icon-Klassen und Icon-Eigenschaften, mit welchen das für die jeweilige Verwendung benötigte Icon maßgeschneidert werden kann.

Im System ZOO wurden mit Hilfe des Icon-Packages drei Icon-Klassen definiert:

Object-Pictures Diese Icons dienen zur Darstellung von gewöhnlichen Objekten einer Wissensbasis. Auf dem Bildschirm erscheinen sie als gewöhnliche Icons, die mit einem Piktogramm und einer Beschriftung versehen sind. Object-Pictures verfügen über die notwendigen Methoden, um die Anwendungsfunktionen des Systems ZOO zur Visualisierung und Manipulation von gewöhnlichen Wissensbasisobjekten zu ermöglichen.

Class-Pictures Diese Icons dienen zur Darstellung von Klassen. Auf dem Bildschirm unterscheiden sie sich von Object-Pictures durch das Vorhandensein einer Umrahmung. Class-Pictures verfügen über eine größere Anzahl von Visualisierungs- und Manipulationsmethoden als die Object-Pictures, da Klassen speziellere Gebilde als die gewöhnlichen Instanzen sind.

Slot-Pictures Diese Icons dienen zur Darstellung der Slots von Objekten. Auf dem Bildschirm erscheinen sie als Beschriftung von Pfeilen. Slot-Pictures besitzen kein Piktogramm, sie sind lediglich rechteckige Textbereiche. Die Methoden der Slot-Pictures legen die Visualisierungs- und Manipulationsmöglichkeiten der Slots von Objekten fest.

Außerdem gibt es zwei Klassen von Linienelementen zum Darstellung von Pfeilen, die dem Nets-Package [Riekert 86a] entstammen: sogenannte *Hooked-Links* zum Zeichnen von Pfeilspitzen und *Links* zum Zeichnen von Pfeilenden. Instanzen dieser Objekte gehören ebenfalls der ZOO-Wissensbasis an, es handelt sich bei diesen aber nicht um Interaktionsobjekte im eigentlichen Sinne; denn sie repräsentieren nur graphisches Wissen, aber kein Wissen um Objekte der

Problemwissensbasis. Der Bezeichnungsweise von Herczeg folgend können diese Linienelemente daher am ehesten als *Darstellungsobjekte* bezeichnet werden [Herczeg 85a].

Auch die Funktionssymbole auf dem Rand eines ZOO-Windows sind durch spezielle Icons repräsentiert. Diese Icons dienen aber nicht direkt zur Visualisierung von Wissensbasisobjekten. Vielmehr haben sie die Aufgabe, die Anwendungsfunktionen des Systems ZOO zu visualisieren. Die meisten Anwendungsfunktionen haben das selektierte Objekt zum Gegenstand und sind in Form von Methoden der Object-Pictures, Class-Pictures oder Slot-Pictures definiert. Die Aktion, die mit dem Anklicken eines Funktionssymbols verbunden ist, besteht daher lediglich im Versenden einer Botschaft an das selektierte Icon im Innern des ZOO-Windows. Die Reaktion auf diese Botschaft ist abhängig davon, ob ein Object-Picture, ein Class-Picture oder ein Slot-Picture selektiert ist, da diese Icons auf die gleichen Botschaften in unterschiedlicher Weise reagieren. So kommt das System ZOO mit einer relativ kleinen Anzahl von Anwendungsfunktionen aus, durch die Typabhängigkeit des Systemverhaltens ist die Benutzerschnittstelle dennoch hinreichend mächtig.

Außer Icons werden im System ZOO noch weitere Interaktionsobjekte verwendet: Auch die Menüs, welche die Funktionen und Auswahlparameter als mit der Maus selektierbare Textzeilen eines Bildschirmfensters visualisieren und die Sheets, die unter anderem das Editieren eines Wissensbasisobjekts auf einem Bildschirmformular ermöglichen, sind durch Interaktionsobjekte repräsentiert.

Zu Beginn diese Kapitels haben wir festgestellt, daß das System ZOO im Grunde nichts anderes ist, als eine Benutzerschnittstelle zu ObjTalk-Wissensbasen. Wie man sieht, ist diese Benutzerschnittstelle aber selbst durch eine Wissensbasis dargestellt. Alles Wissen des Gesamtsystems ist also mit Hilfe desselben fundamentalen Konzepts, des Objekts, repräsentiert.

6.7 Implementation

Das System ZOO ist in der auf ObjTalk und Franz Lisp aufbauenden Software-Entwicklungsumgebung der Forschungsgruppe INFORM entstanden. Als Basis-Software für die Realisierung der Mensch-Maschine-Schnittstelle diente das Fenstersystem *WLISP* [Fabian 84]. Zusätzlich wurden die folgenden Softwarepakete der Forschungsgruppe INFORM verwendet: Das *Icons-Package* zur Visualisierung von Objekten in Form von Piktogrammen [Herczeg 85b], das

Nets-Package zur Darstellung und Manipulation von Netzstrukturen [Riekert 86a] und das *Sheets-Package* zur formularbasierten Interaktion [Riekert 86b] mit dem System. Im Rahmen der Arbeit am System ZOO wurden außerdem kompatible ObjTalk-Erweiterungen vorgenommen, welche die objektorientierte Darstellung von Slotbeschreibungen und Wissensbasen ermöglichen. Als Bildschirmgerät findet ein *BitGraph-Terminal* der Firma BBN Computer Verwendung. Das Terminal besitzt einen Bitmap-Raster-Bildschirm mit 768 mal 1024 Bildelementen und ermöglicht die Verwendung einer Maus als Eingabeinstrument. Das verwendete Rechnersystem ist eine VAX 11/780 mit Betriebssystem UNIX.

6.8 Zusammenfassung

Die Berücksichtigung von Wissensaspekten ermöglicht eine neue Sichtweise der Mensch-Computer-Kommunikation. Das Problem, verständliche Benutzerschnittstellen zu konstruieren, stellt sich dar als die Frage nach einer benutzergerechten externen Wissensrepräsentation und ist eng verbunden mit dem Prinzip der direkten Manipulation.

Direkte Manipulation beruht auf der externen Repräsentation von Wissen, die mit der internen Wissensdarstellung eng gekoppelt sein muß. Große Vorteile bietet hier eine integrierte objektorientierte Systemarchitektur, die den Systemkern und die Benutzerschnittstelle gleichermaßen umfaßt. Eine objektorientierte Benutzerschnittstelle ermöglicht den Einsatz von Interaktionsobjekten, die sich nach außen hin als sensitive Bildschirmobjekte zur externen Wissensrepräsentation zeigen, nach innen hin aber aktive Datenstrukturen sind, die mit den internen Objekten zur Wissensrepräsentation kommunizieren. Mit einer objektorientierten internen Wissensrepräsentation haben die Benutzerschnittstellenobjekte eine eindeutige interne Entsprechung und eine enge Kopplung beider Darstellungen wird möglich.

Das Objekt als fundamentales Konzept der Programmierung und Wissensrepräsentation ermöglicht die einfache Konstruktion von Wissenseditoren und Metasystemen, mit denen Fakten und Konzepte des Systems visualisiert und vom Benutzer modifiziert werden können. Die Anpassung von Softwaresystemen an neue Anforderungen erfolgt in Form eines Wissenserwerbsvorgangs, für den keine besonderen Programmierkenntnisse erforderlich sind.

Der graphische Wissenseditor ZOO erlaubt die direkte Manipulation von Wissensbasen und operiert "am lebenden Objekt". Nicht etwa eine Textrepräsentation wird editiert, sondern die aktiven Wissensbasisobjekte eines Anwendungssystems in geladenem Zustand. Dadurch kann die Interaktion mit dem System wissensgesteuert erfolgen: Das bereits im System repräsentierte Wissen wird genutzt, um die Vorgänge des Wissenserwerbs und der Visualisierung von Wissensstrukturen zu unterstützen.

Die Benutzerschnittstelle des Wissenseditors ZOO ist gekennzeichnet durch eine ausgiebige Nutzung von Graphik, durch eine weitgehend modusfreie Interaktion dank der angewandten Noun-Verb-Syntax, durch generische Operationen, die sich auf alle Arten von Objekten anwenden lassen, durch unmittelbares Feedback zu allen Benutzeraktionen ("what you see is what you get") und durch eine konsequente Einknopfbedienung der Maus. Deshalb ist das System sehr einfach erlernbar und gleichzeitig sehr leistungsfähig.

7. Zusammenfassung

Das Paradigma der Wissensverarbeitung führt zu einer neuen Betrachtungsweise der Rechnernutzung. Die Bedeutung von Daten und Prozeduren tritt zurück, im Blickfeld stehen stattdessen Fakten und Konzepte, diein Wissensbasen repräsentiert sind. Diese Fakten und Konzepte können von wissensbasierten Systemen und Systemkomponenten auf vielfältige Weise aufgebaut und genutzt werden.

Dabei ändert sich der Charakter vieler am Computer verrichteter Tätigkeiten, die nun als Vorgänge des Wissenserwerbs und der Modellierung von Wissen gesehen werden können:

- Die Datenerfassung im herkömmlichen Sinn wird abgelöst durch einen Vorgang der Assimilation von Faktenwissen, der durch das im System repräsentierte konzeptuelle Wissen gesteuert wird. Aus diesem konzeptuellen Wissen leiten sich Hilfe und Unterstützungsfunktionen für den Benutzer ebenso ab wie die erforderlichen Modifikationen der Wissensbasis zur konsistenten Einbettung der neu eingebrachten Fakten. Das D&I Anwendungssystem gibt hierfür ein Beispiel ab.

- Die Tätigkeit der Einrichtung und Umgestaltung eines Softwaresystems nimmt eine neue Gestalt an und erscheint als ein Vorgang der Akkommodation von konzeptuellem Wissen. Hierbei kann aus dem Metawissen, das der Anwendung zugrundeliegt, Nutzen gezogen werden und der Systemverwalter in ähnlicher Weise wie beim der Assimilation von Faktenwissen unterstützt werden. Für diese Aufgabenstellung verwendet das D&I Metasystem sogar dieselbe Benutzerschnittstelle wie das D&I Anwendungssystem.

- Die Entwicklung von Softwaresystemen geschieht, indem Wissensbasen aufgebaut werden, die das Anwendungsgebiet, die Benutzerschnittstelle, das Benutzermodell und den Systemzustand beschreiben. Der Vorgang der Programmierung eines Rechnersystems wird so überlagert durch die Modellierung von Fakten und Konzepten mit Hilfe eines Wissensrepräsentationsformalismus. Der beschriebene Wissenseditor ZOO unterstützt diesen Vorgang für Systeme, die auf der objektorientierten Repräsentationssprache ObjTalk basieren.

Der Wandel in den Tätigkeiten zieht auch eine Veränderung des Benutzerbildes nach sich: Dank der besseren Unterstützung kann nun ein Sachbearbeiter komplexere Sachverhalte als bisher auf dem Computer darstellen und bearbeiten. Die Umgestaltung von Anwendungssystemen kann in vielen Fällen von einem programmierunkundigen Anwendungsexperten vorgenommen werden, es ist kein Softwareexperte mehr erforderlich. Der Entwurf von wissensbasierten Sy-

stemen bleibt zwar nach wie vor eine schwierige Aufgabe, der Systementwerfer ist aber von vielem programmtechnischen Ballast befreit und kann sich auf die Konzepte des Anwendungsgebiets konzentrieren. Neue Tätigkeitsbezeichnungen wie *Knowledge Worker* oder *Knowledge Engineer* tragen den veränderten Aufgabenprofilen Rechnung.

Die beschriebenen Systeme, das D&I Anwendungssystem, das D&I Metasystem und der Wissenseditor ZOO gehen in dieser Reihenfolge zunehmend mit abstrakterem Wissen um und werden von ihrer Anwendung her gleichzeitig immer universeller. Im gleichen Maße nimmt aber auch die Unterstützungsfunktion ab, die diese Systeme bieten können.

- Das D&I Anwendungssystem, das auf eine kleine konzeptuelle Welt beschränkt ist, kann am meisten Hilfe bieten, da die Konzepte der darstellbaren Welt in einer Wissensbasis repräsentiert sind und bei der Assimilation genutzt werden können.
- Die Akkommodation der Konzepte des D&I Systems mit Hilfe seiner Metakomponente kann durch das Metawissen unterstützt werden, das in den Metakonzepten TYP, ASPEKT, SCRIPT, MERKMAL usw. repräsentiert ist. Diese Metakonzepte sind teilweise noch anwendungsspezifisch, indem sie sich auf Aufgaben der Dokumentation und Information beziehen. Allerdings besteht hier bereits eine große Freiheit, wie neues Gebietswissen in Konzepte umgesetzt werden kann.
- Die Modellierung von Wissen in einer Wissensbasis mit Hilfe des Wissenseditors ZOO ist der allgemeinste betrachtete Fall. Die Verwendung einer so erstellten Wissensbasis ist nicht begrenzt auf irgendein vorhersehbares Anwendungsgebiet. Beim erstmaligen Aufbau von konzeptuellem Wissen ist infolgedessen auch keine anwendungsspezifische Wissensbasis vorhanden. Von Anfang an vorhanden ist aber die Metawissensbasis des ObjTalk-Systems, welche die Metaobjekte *class*, *object*, *Method*, *slot* usw. enthält. Diese Metaobjekte repräsentieren anwendungsunabhängige Muster und Schemata der Wissensrepräsentation, die vom Wissenseditor ZOO zur Unterstützung und Steuerung des Modelliervorgangs genutzt werden können. Bei der Erweiterung und Modifizierung eines Anwendungssystems kann der Wissenseditor ZOO bereits ähnlich wie das D&I Metasystem anwendungsbezogene Hilfen anbieten, da dann schon problemspezifische Wissensbasen aufgebaut sind.

In den Systemen D&I und ZOO werden wichtige Aspekte des Erwerbs von Wissen deutlich. Die Wissenserwerbsvorgänge der Assimilation und der Akkommodation erscheinen unter einem neuen Blickwinkel und stellen sich als gleichgeartete Vorgänge heraus. In beiden Fällen ist der Erwerb von Wissen einer konkreteren Stufe möglich in Form der Nutzung von Wissen einer höheren Stu-

fe. Dies gilt für die Assimilation von Faktenwissen als Vorgang der Nutzung von konzeptuellem Wissen ebenso, wie für den Erwerb von konzeptuellem Wissen durch Nutzung von Metawissen. Diese Stufung von Wissen muß sich nicht endlos fortsetzen, sie läßt sich nach oben abschließen durch eine Form von Metawissen, das nicht nur niederere Formen von Wissen, sondern auch sich selbst beschreibt. Zumindest dieses Metawissen muß durch einen Bootstrapprozeß als initiales Wissen vorhanden sein. Dann aber kann selbst dieses Metawissen noch durch Akkommodationsvorgänge verändert werden.

Für die künftige Fortentwicklung der beschriebenen Systeme gibt es zwei wichtige weiterführende Zielsetzungen:

- Die beschriebenen Techniken der Wissensdarstellung haben zum Ziel, alle Funktionalität eines Softwaresystems auf deskriptives Wissen zurückzuführen. Dies ist aber in der Praxis oft nicht möglich, selbst wenn man sich auf das eigentliche Anwendungswissen beschränkt. An die Stelle von exaktem, deskriptivem Wissen treten Heuristiken, die sich in Form von Regeln mit Seiteneffekten ausdrücken. Häufig sogar kommt man auch nicht umhin, das Wissen prozedural in Form einer Methode darzustellen. Ein wichtiges Ziel bei der Weiterentwicklung von Wissenseditoren wird daher sein, benutzernahe Darstellungsformen für Regeln und Methoden zu entwickeln, die besser sind als die herkömmlichen Textrepräsentationen.
- Beim Vorgang der Wissensübertragung hin zum Computer kommt dem Benutzer der vorgestellten Systeme eine aktive Rolle zu. Dies garantiert den Werkzeugcharakter und die damit verbundene Kontrollierbarkeit der Systeme. Die Freiheit, die der Mensch bei der Modellierung von Wissen hat, bedeutet aber auch, daß der Benutzer zu vielen Überlegungen und Entscheidungen gezwungen ist, die ihm das System nicht abnimmt. Ein wichtiges Ziel für künftige Forschungen auf dem Gebiet des computerunterstützten Wissenserwerbs wird deshalb sein, auch noch andere Methodiken zu erproben, bei denen das System in stärkerem Maße Initiative übernimmt, indem es beispielsweise den Benutzer befragt oder aus vom Benutzer vorgelegten konkreten Beispielen selbständig abstrakte Konzepte ableitet.

Der objektorientierte Ansatz erwies sich bei der Implementation der beschriebenen Systeme als sehr vorteilhaft. Alle objektorientiert dargestellten Teile eines Systems liegen in einer operationalisierten Form vor, sie sind zugreifbar und veränderbar. Vermittels einer ebenfalls objektorientierten Benutzerschnittstelle präsentieren sich die internen Objekte auf dem Bildschirm eines Computerterminals, wo sie vom Benutzer durch direkte Manipulation erzeugt und bearbeitet werden können. Da Objekte aktive Strukturen sind, können diese Manipulationen des Benutzers Inferenzmechanismen auslösen, welche die konsistente Einbettung der neuen Informationen in das Netz der Wissensbasisobjekte bewirken.

Die Schlüsselidee bei den vorgestellten Systemen liegt in der Kombination einer wissensbasierten Systemarchitektur mit fortgeschrittenen Interaktionstechniken, wobei Anwendungswissen und Benutzerschnittstelle in einer einheitlichen Weise repräsentiert sind. Dadurch kann zu dem im System repräsentierten Anwendungswissen ein interaktiver Zugang geschaffen werden. Da die Benutzerschnittstelle selbst durch eine Wissensbasis dargestellt ist und den Inhalt weiterer Wissensbasen nutzt, geschieht darüber hinaus die Interaktion selbst wissensbasiert. Beides zusammen ermöglicht die Konstruktion besser durchschaubarer und in weiten Teilen vom Benutzer adaptierbarer wissensverarbeitender Softwaresysteme und leistet so einen wichtigen Beitrag zur Gestaltung benutzergerechter Computersysteme.

Anhang A. Merkmalsbeschreibungen im System D&I

Merkmale und Merkmalsbeschreibungen besitzen eine Schlüsselfunktion im System D&I. Viele der Inferenzmechanismen des Systems werden durch Zugriffe auf Merkmale aktiviert. Außerdem bestimmen Merkmalsbeschreibungen die Präsentation von Merkmalswerten an der Benutzerschnittstelle des Systems. Im nächsten Abschnitt werden die Sprachelemente der D&I-Merkmalsbeschreibungen vorgestellt. Der darauffolgende Abschnitt enthält den Lisp-Code des etwas vereinfachten Interpreters für die Verarbeitung von Botschaften zum Zugriff auf Merkmale im System D&I.

A.1 Merkmalsdeskriptoren

Merkmalsbeschreibungen bestehen aus einzelnen Merkmalsdeskriptoren. Für das D&I Anwendungssystem erscheinen Merkmalsbeschreibungen als Assoziationslisten; Merkmalsdeskriptoren sind die Elemente dieser Assoziationslisten. Im D&I-Metasystem sind Merkmalsbeschreibungen durch Wissensbasisobjekte vom Typ MERKMAL repräsentiert; Merkmalsdeskriptoren sind dann in den Merkmalen derartiger Objekte dargestellt. Im folgenden werden die verschiedenen Arten von Merkmalsdeskriptoren des Systems D&I beschrieben. Das Merkmal, das durch die einzelnen Deskriptoren beschrieben wird, wird im folgenden als *Merkmal* bezeichnet.

Höhere Beschreibungselemente

default *Default* — Merkmalsdeskriptor

Deklaration eines Defaultwerts für *Merkmal*.

typ *Typ* — Merkmalsdeskriptor

Typdeklaration: Als Werte von *Merkmal* kommen nur Wissensbasisobjekte vom Typ *Typ* in Frage.

format *Format* — Merkmalsdeskriptor

Dieser Deskriptor beschreibt das Format der Werte von *Merkmal*. Es gibt die Formate *Liste* und *Einzelwert*.

invers *Inverses-Merkmal* Merkmalsdeskriptor

Inverses-Merkmal wird als invers zu *Merkmal* deklariert. Aus diesem Deskriptor werden X-Link-Konfigurationen abgeleitet.

llink *Funktion* Merkmalsdeskriptor

Deklaration einer Funktion zur Bestimmung einer abhängigen Merkmalsbesetzung: *Funktion*, angewandt auf einen *Wert* von *Merkmal*, ergibt eine abhängige Merkmalsbesetzung, die durch eine Liste aus drei Elementen (Abhängiges-Objekt *Abhängiges-Merkmal Abhängiger-Wert*) dargestellt ist. Diese Merkmalsbesetzung wird solange aufrechterhalten, als *Wert* gültig ist. Aus diesem Deskriptor werden L-Link-Konfigurationen abgeleitet. Falls das Merkmal multiple Werte (Listen) annimmt, wird die *Funktion* auf jedes einzelne Listenelement angewandt, ansonsten wirkt sie auf den Gesamtwert.

Triggers beim Modifizieren eines Merkmals

if-set *If-Set-Trigger* Merkmalsdeskriptor

Deklaration einer Triggerfunktion: Wenn *Merkmal* mit einem neuen *Wert* besetzt wird, wird die Funktion *If-Set-Trigger* mit dem Parameter *Wert* versorgt und aufgerufen.

if-added *If-Added-Trigger* Merkmalsdeskriptor

Deklaration einer Triggerfunktion: Wenn durch einen modifizierenden Zugriff zu *Merkmal* ein neuer *Wert* hinzugefügt wird, wird die Funktion *If-Added-Trigger* mit dem Parameter *Wert* versorgt und aufgerufen. Falls *Merkmal* multiple Werte (Listen) annimmt, wird für jedes hinzukommende Listenelement die Funktion *If-Added-Trigger* aufgerufen, ansonsten wirkt die Funktion auf den Gesamtwert.

if-removed *If-Removed-Trigger* Merkmalsdeskriptor

Deklaration einer Triggerfunktion: Wenn durch einen modifizierenden Zugriff von *Merkmal* ein *Wert* entfernt wird, wird die Funktion *If-Removed-Trigger* mit dem Parameter *Wert* versorgt und aufgerufen. Falls *Merkmal* multiple Werte (Listen) annimmt, wird für jedes entfernte Listenelement die Funktion *If-Removed-Trigger* aufgerufen, ansonsten wirkt die Funktion auf den alten Gesamtwert.

Triggers beim Lesen eines Merkmals

if-needed *If-Needed-Trigger* Merkmalsdeskriptor
Deklaration einer Triggerfunktion: Bei einem lesenden Zugriff wird der Wert des Funktionsaufrufs *(If-Needed-Trigger)* zurückgegeben.

if-not-present *If-Not-Present-Trigger* Merkmalsdeskriptor
Deklaration einer Triggerfunktion: Bei einem lesenden Zugriff wird der Wert des Merkmals zurückgegeben, sofern dieses ungleich NIL ist. Andernfalls wird das Merkmal auf den Wert *(If-Not-Present-Trigger)* gesetzt und dieser Wert zurückgegeben.

Konvertierung zwischen interner und externer Darstellung von Merkmalswerten

if-filled *If-Filled-Konversion* Merkmalsdeskriptor
Ein mit Hilfe der Botschaft `fill:` an *Merkmal* geschickter *Wert* wird mit der Funktion *If-Filled-Konversion* konvertiert und anschließend in *Merkmal* eingetragen. Falls *Merkmal* multiple Werte (Listen) annimmt, wird für jedes hinzukommende Listenelement die Funktion *If-Filled-Konversion* aufgerufen, ansonsten wirkt die Funktion auf den Gesamtwert.

if-printed *If-Printed-Konversion* Merkmalsdeskriptor
Ein mit Hilfe der Botschaft `print:` zur Ausgabe anstehender Wert wird über die normale Lesefunktion für Merkmale abgerufen und anschließend mit der Funktion *If-Printed-Konversion* konvertiert. Falls *Merkmal* multiple Werte (Listen) annimmt, wird die Funktion *If-Printed-Konversion* auf jedes auszugebende Listenelement angewandt, ansonsten wirkt die Funktion auf den Gesamtwert.

Deskriptoren, die von der Benutzerschnittstelle genutzt werden

name *Externer-Merkmalsname* Merkmalsdeskriptor
Mit dem Namen *Externer-Merkmalsname* wird *Merkmal* extern dargestellt.

feldhoehe *Feldhöhe* Merkmalsdeskriptor
Platzbedarf zur externen Darstellung von *Merkmal*

argumente *Lisp-Ausdruck* Merkmalsdeskriptor

Der Wert von *Lisp-Ausdruck* ergibt eine Liste von möglichen Werten für *Merkmal*. Wenn kein *Lisp-Ausdruck* spezifiziert ist, wird die Botschaft `argumente:` an den *Typ* von *Merkmal* geschickt.

hilfe *Lisp-Ausdruck* Merkmalsdeskriptor

Der Wert von *Lisp-Ausdruck* ergibt einen Beschreibungstext für *Merkmal*. Wenn kein *Lisp-Ausdruck* spezifiziert ist, wird die Botschaft `hilfe:` an den *Typ* von *Merkmal* geschickt.

syntax *Lisp-Ausdruck* Merkmalsdeskriptor

Der Wert von *Lisp-Ausdruck* ergibt ein Syntaxmuster für die Werte von *Merkmal*. Wenn kein *Lisp-Ausdruck* spezifiziert ist, wird die Botschaft `syntax:` an den *Typ* von *Merkmal* geschickt.

protection *Schutzcode* Merkmalsdeskriptor

Der *Schutzcode* wird von der Benutzerschnittstelle verwandt, um Schreib- und Leseberechtigung für *Merkmal* zu überprüfen.[45]

restriction *Prädikat* Merkmalsdeskriptor

Deklaration einer Überprüfungsfunktion für Merkmalswerte: Das *Prädikat*, angewandt auf einen einzutragenden *Wert* muß `t` (*true*) ergeben. Falls *Merkmal* multiple Werte (Listen) annimmt, wird jedes hinzukommende Listenelement einzeln überprüft, ansonsten gilt das *Prädikat* für den Gesamtwert.[46]

---

[45] In der implementierten D&I-Version wird die Zugriffsberechtigung in etwas unterschiedlicher Weise spezifiziert.

[46] Von der implementierten D&I-Version wird dieser Deskriptor derzeit nicht genutzt.

A.2 Botschafteninterpreter für Merkmale

Das nachfolgende Lisp-Programm ist ein Interpreter zur Bearbeitung von Botschaften zum Zugriff auf Merkmale. Der Code des Interpreters wurde zum Zweck einer besseren Verständlichkeit gegenüber dem im System D&I verwendeten Original leicht vereinfacht. Der Interpreter nutzt einige der im letzten Abschnitt beschriebenen Merkmalsdeskriptoren. Es werden folgende Arten von Botschaften bearbeitet:

Merkmal	Lesender Zugriff auf *Merkmal*
Merkmal = *Wert*	Schreibender Zugriff auf *Merkmal*
Merkmal < *Wert*	Hinzufügen eines Werts zu *Merkmal*
Merkmal > *Wert*	Entfernen eines Werts von *Merkmal*
Merkmal `print:`	Aufbereiten des Inhalts von Merkmal zur Ausgabe durch die Benutzerschnittstelle.
Merkmal `fill:` *Wert*	Eintrag eines Werts von Seiten der Benutzerschnittstelle.

Programmcode

```
;*** Fundamentale Methode zum Zugriff auf Merkmale
;
;    <name>.... Name des Merkmals (= 1. Element der Botschaft)
;    <rest>.... Rest der Botschaft
;    <descr>... Merkmalsbeschreibung in Form
;               einer Assoziationsliste
;
;    Die Merkmale des Objekts sind realisiert als Assoziations-
;    listen, die an das Symbol self gebunden sind.


(def xslotmeth
  (lambda (<name> <descr> <rest>)
    (selectq (car <rest>)
             (nil (xget <name> <descr> <rest>))
             (= (xset <name> <descr> <rest>))
             (< (xpush <name> <descr> <rest>))
             (> (xdrop <name> <descr> <rest>))
             (fill: (xfill <name> <descr> <rest>))
             (print: (xprint <name> <descr> <rest>)))))
```

```
;*** Abrufen eines Merkmalswertes

(def xget
  (lambda (<name> <descr> <rest>)
    (selectq (caar <descr>)
             (if-needed (mapn (cdar <descr>)))
             (if-not-present
              (x-if-not-present
               (cdar <descr>) <name> (cdr <descr>) <rest>))
             (nil (cadr (assq <name> self)))
             (xget <name> (cdr <descr>) <rest>))))

(def x-if-not-present
  (lambda (foos <name> <descr> <rest>)
    (let ((value (xget <name> <descr> <rest>)))
         (cond (value)
               ((xset <name>
                      <descr>
                      `(= ,(mapn foos)))
                (xget <name> <descr> <rest>))))))



;*** Besetzen eines Merkmals mit einem neuen Wert

(def xset
  (lambda (<name> <descr> <rest>)
    (selectq (caar <descr>)
     (if-set (x-if-set (cdar <descr>) <name> (cdr <descr>) <rest>))
     (if-added (x-if-added (cdar <descr>) <name> (cdr <descr>) <rest>))
     (if-removed (x-if-removed (cdar <descr>) <name> (cdr <descr>) <rest>))
     (llink (x-llink (cdar <descr>) <name> (cdr <descr>) <rest>))
     (invers (x-invers (cdar <descr>) <name> (cdr <descr>) <rest>))
     (nil (let ((ass (assq <name> self)))
            (cond (ass (rplacd ass (cadr <rest>)))
                  (t (nconc self (list (cons <name> (cadr <rest>))))))))
     (xset <name> (cdr <descr>) <rest>))))

(def x-if-set
  (lambda (foos <name> <descr> <rest>)
    (prog1 (xset <name> <descr> <rest>)
           (mapc (function (lambda (foo) (funcall foo (cadr <rest>))))
                 foos))))
```

```
(def x-if-added
  (lambda (foos <name> <descr> <rest>)
    (let ((oldfiller (xget <name> <descr> nil)) (newfiller (cadr <rest>)))
         (xset <name> <descr> <rest>)
         (mapdiff (function
                   (lambda (added-element)
                     (mapc (function
                            (lambda (foo) (funcall foo added-element)))
                           foos)))
                  newfiller
                  oldfiller)
         newfiller)))

(def x-if-removed
  (lambda (foos <name> <descr> <rest>)
    (let ((oldfiller (xget <name> <descr> nil)) (newfiller (cadr <rest>)))
         (xset <name> <descr> <rest>)
         (mapdiff (function
                   (lambda (removed-element)
                     (mapc (function
                            (lambda (foo) (funcall foo removed-element)))
                           foos)))
                  oldfiller
                  newfiller)
         newfiller)))

(def x-invers
  (lambda (linkslots <name> <descr> <rest>)
    (x-llink (mapcar (function
                      (lambda (linkslot)
                        `(lambda (el)
                           (list el
                                 ,(list 'quote linkslot)
                                 ,(list 'quote self)))))
                     linkslots)
             <name>
             <descr>
             <rest>)))
```

```
(def x-llink
  (lambda (pathfoos <name> <descr> <rest>)
    (let ((oldfiller (xget <name> <descr> nil))
          (newfiller (cadr <rest>)))
         (xset <name> <descr> <rest>)
         (mapc (function
                (lambda (pathfoo)
                  (mapdiff '(lambda (added-element)
                              ((lambda (path)
                                 (ask ,(car path) ,(cadr path)
                                                  < ,(caddr path)))
                               (,pathfoo added-element)))
                           newfiller
                           oldfiller)
                  (mapdiff '(lambda (removed-element)
                              ((lambda (path)
                                 (ask ,(car path) ,(cadr path)
                                                  > ,(caddr path)))
                               (,pathfoo removed-element)))
                           oldfiller
                           newfiller)))
               pathfoos)
         newfiller)))

;*** Hinzufügen eines Merkmalswertes

(def xpush
  (lambda (<name> <descr> <rest>)
    (cond ((liste? <descr>)
           (let ((oldfiller (makl (xget <name> <descr> nil))))
                (let ((newfiller oldfiller))
                     (do ((to-push (makl (cadr <rest>)) (cdr to-push)))
                         ((null to-push) newfiller)
                         (cond
                          ((not (memq (car to-push) newfiller))
                           (setq newfiller
                                 (cons (car to-push) newfiller)))))
                     (cond
                      ((not (eq newfiller oldfiller))
                       (xset <name> <descr> '(= ,newfiller))))
                     newfiller)))
          ((xset <name> <descr> <rest>)))))
```

```
;*** Entfernen eines Merkmalswerts

(def xdrop
  (lambda (<name> <descr> <rest>)
    (let ((oldfiller (xget <name> <descr> nil)))
         (cond ((liste? <descr>)
                (setq oldfiller (makl oldfiller))
                (let ((newfiller (copy oldfiller)))
                     (do ((to-drop (makl (cadr <rest>)) (cdr to-drop)))
                         ((null to-drop) newfiller)
                         (cond
                          ((memq (car to-drop) newfiller)
                           (setq newfiller (delq (car to-drop) newfiller)))))
                     (cond ((not (equal newfiller oldfiller))
                            (xset <name> <descr> '(= ,newfiller)))
                           (t newfiller))))
               ((equal (cadr <rest>) oldfiller)
                (xset <name> <descr> '(= nil)))
               (t oldfiller)))))


;*** Aufbereiten eines Merkmalswerts zum Zweck der Ausgabe

(def xprint
  (lambda (<name> <descr> <rest>)
    (let ((if-printeds (cdr (assq 'if-printed <descr>))))
         (cond ((null if-printeds) (xget <name> <descr> <rest>))
               ((liste? <descr>)
                (mapcar (function
                         (lambda (to-print)
                           (xprint1 to-print if-printeds)))
                        (xget <name> <descr> <rest>)))
               (t (xprint1 (xget <name> <descr> <rest>) if-printeds))))))

(def xprint1
  (lambda (to-print if-printeds)
    (cond ((null if-printeds) to-print)
          (t (xprint1 (funcall (car if-printeds) to-print)
                      (cdr if-printeds))))))
```

```
;*** Besetzen eines Merkmals durch den Benutzer

(def xfill
  (lambda (<name> <descr> <rest>)
    (let  ((class (cadr (assq 'typ <descr>))))
          (cond ((liste? <descr>)
                 (let ((oldfiller (makl (xprint <name> <descr> nil)))
                       (xset <name>
                             <descr>
                             (list '=
                                   (mapcar (function
                                             (lambda (fillerelement)
                                              (cond ((member fillerelement
                                                             oldfiller)
                                                     fillerelement)
                                                    (t (xfill1 fillerelement
                                                               class
                                                               <descr>)))))
                                           (makl (cadr <rest>)))))))
                ((or (atom (cadr <rest>)) (cdadr <rest>))
                 (cond ((equal (xprint <name> <descr> nil) (cadr <rest>))
                        (cadr <rest>))
                       ((xset <name>
                              <descr>
                              (list '=
                                    (xfill1 (cadr <rest>)
                                            class
                                            <descr>))))))
                (t (cond ((equal (xprint <name> <descr> nil) (caadr <rest>))
                          (caadr <rest>))
                         ((xset <name>
                                <descr>
                                (list '=
                                      (xfill1 (caadr <rest>)
                                              class
                                              <descr>)))))))))

(def xfill1
  (lambda (filler class <descr>)
    (cond (class
           (ask ,class
                identify:
                ,(xfill2 filler (cdr (assq 'if-filled <descr>)))))
          ((xfill2 filler (cdr (assq 'if-filled <descr>)))))))

(def xfill2
  (lambda (filler if-filleds)
    (cond (if-filleds
           (xfill2 (funcall (car if-filleds) filler) (cdr if-filleds)))
          (filler))))
```

```
;*** Hilfsfunktionen

(def liste?
  (lambda (description)
    (= (cadr (assq 'format description)) 'list)

(def makl
  (lambda (cand)
    (cond ((listp cand) cand) ((list cand)))))

(def mapdiff
  (lambda (foo more less)
    (setq more (makl more) less (makl less))
    (cond
     ((not (equal more less))
      (mapc (function
             (lambda (el)
               (cond ((not (memq el less)) (funcall foo el)))))
            more)))))

(def mapn
  (lambda (foos)
    (cond
      ((null foos) nil)
      ((cdr foos) (funcall (car foos)) (mapn (cdr foos)))
      ((funcall (car foos))))))
```

Anhang B. Die Benutzerschnittstelle des Systems ZOO

Alle Anwendungsfunktionen des Systems ZOO werden durch Anklicken sensitiver Bildschirmbereiche mit Hilfe des linken Mausknopfs ausgelöst. Man kann drei Arten von derartigen Bildschirmbereichen unterscheiden:

- die Icons im Innenbereich des Zoofensters,
- der freie Innenbereich des Zoofensters,
- die Funktionssymbole am rechten Rand des Zoo-Fensters

Das Anklicken der Funktionssymbole in der Titelzeile und der Scroll-Bars mit dem linken Mausknopf, sowie das Anklicken des Fensters mit dem rechten Mausknopf lösen keine Anwendungsfunktionen aus, sondern bewirken universelle Funktionen des Fenstersystems, die an anderer Stelle beschrieben sind [Fabian 84; Bauer J. 85].

B.1 Die Icons im Innenbereich eines Zoo-Fensters

Im Innenbereich des Zoo-Fensters ist das Wissensnetz dargestellt, das mit dem System ZOO editiert werden kann. Die Icons, aus denen das Wissensnetz gebildet, stellen sensitive Bereiche dar, welche mit Hilfe der Maus angesprochen werden können. Es gibt drei Arten von solchen Icons:

1. Icons, welche Klassen visualisieren, erkennbar durch dick umrandete Piktogramme (sogenannte Class-Pictures),
2. Icons, welche die übrigen Objekte visualisieren (gewöhnliche Instanzen, Slotbeschreibungen, Methoden usw.), diese sind ebenfalls als Piktogramme sichtbar (sogenannte Object-Pictures),
3. Text-Icons, welche die Beschriftung von Pfeilen bilden und für die Relationen zwischen den Objekten stehen (sogenannte Slot-Pictures).

Alle drei Arten von Icons halten eine Selektier-Funktion bereit, die man durch Drücken des linken Mausknopfes erhält:

Select Anwendungsfunktion

Nach der Aktivierung der Funktion erscheint das betreffende Bildschirmobjekt in Inversdarstellung. Das invers dargestellte Objekt bzw. Slot gilt als selektiert, das heißt es ist das Argument aller objektspezifischen Anwendungsfunktionen, die man über die Funktionssymbole auf dem rechten Fensterrand auswählen kann.

B.2 Freier Innenbereich eines Zoo-Fensters

Ein Zoo-Fenster hält in seinem freien Innenbereich folgende Funktionalität bereit, die man durch Drücken des linken Mausknopfes erhält:

Move Anwendungsfunktion

Die Form des Maus-Pointers verändert sich zum sogenannten *Move-Cross*, daraufhin läßt sich das aktuell selektierte Bildschirmobjekt bewegen und durch einen zweiten Klick an einer anderen Position plazieren.

B.3 Funktionssymbole am rechten Rand des Zoo-Fensters

Der rechte Rand eines Zoo-Fensters enthält eine Reihe von Icons, welche die Anwendungsfunktionen des Systems ZOO visualisieren. Durch das Anklicken eines derartigen Icons mit Hilfe des linken Mausknopfs wird die betreffende Anwendungsfunktion ausgelöst. Die meisten derartigen Anwendungsfunktionen beziehen sich auf das selektierte Objekt. Diese Anwendungsfunktionen sind generische Operationen, sie haben abhängig vom Typ des selektierten Objekts unterschiedliche Wirkung. In Einzelfällen sind sie auf das selektierte Objekt nicht anwendbar, es ertönt dann die Terminalhupe.[47]

Edit Anwendungsfunktion

Diese Funktion hat unterschiedliche Wirkung, je nachdem ob ein Objekt (ungeachtet ob Klasse oder Instanz) über sein Icon oder ein Slot über eine Pfeilbeschriftung selektiert ist.

- Für ein selektiertes Objekt erscheint ein Property-Sheet, in dem die Slots des Objekts dargestellt sind und editiert werden können. Diejenigen Slots, die in der zugehörigen Slotbeschreibung beim Merkmal *filter* das Symbol *zoo* eingetragen haben, werden im Property-Sheet nicht visualisiert.
- Für ein selektiertes Slot erscheint ein einzeiliges Property-Sheet, in welchem der Wert des Slots editiert werden kann.

---

[47] Dies ist keine voll befriedigende Lösung, was man sich eigentlich wünscht, ist, daß in einem derartigen Fall die betreffende Anwendungsfunktion gar nicht angeboten wird, bzw. durch eine besondere graphische Gestaltung als inaktiv gekennzeichnet wird. In einer künftigen ZOO-Version soll dieser Wunsch berücksichtigt werden.

Make Anwendungsfunktion

Diese Funktion ist nur bedeutungsvoll, wenn eine Klasse oder eine Instanz selektiert ist. Es erscheint ein Menu aus folgenden Funktionen:

Name = Wird hier eine Zeichenfolge <name> eingetragen, so hat dies zwei Wirkungen: Der *pname* des Objekts wird auf <name> gesetzt und das Objekt wird an das Symbol <name> gebunden.

MakeInstance Es wird eine Instanz der dargestellten Klasse erzeugt und an einer Default-Position auf dem Bildschirm visualisiert. Die entstandene Instanz wird zum selektierten Objekt (Diese Funktion erscheint nur für Klassen).

MakeSubclass Es wird eine Subklasse der dargestellten Klasse erzeugt und an einer Default-Position auf dem Bildschirm visualisiert. Die entstandene Subklasse wird zum selektierten Objekt (Diese Funktion erscheint nur für Klassen).

Copy Anwendungsfunktion

Diese Funktion ist nur für Objekte (Klassen oder Instanzen) sinnvoll. Es wird eine Kopie des betreffenden Objekts erzeugt und an einer Default-Position auf dem Bildschirm visualisiert. Die entstandene Kopie wird zum selektierten Objekt.

Delete Anwendungsfunktion

Falls ein Objekt selektiert ist, wird dieses (mittels der Botschaft *kill:*) nach positivem Bestätigungsdialog gelöscht. Falls ein Slot eines Objekts selektiert ist, wird sein Inhalt (mittels der Slotbotschaft *forget:*) vergessen.

Step Anwendungsfunktion

Diese Funktion hat unterschiedliche Wirkung, je nachdem ob ein Objekt (d.h. eine Klasse oder eine Instanz) oder ein Slot selektiert ist.

- Für ein selektiertes Objekt erscheint ein Menü, in dem die Slots des Objekts aufgelistet sind, wobei diejenigen Slots, die in der zugehörigen Slotbeschreibung beim Merkmal *filter* das Symbol *zoo* eingetragen haben, im Menü unterdrückt werden.
- Für ein selektiertes Slot erscheint ein Menü, das die Objekte enthält, die in diesem Slot eingetragen sind.

Nach Auswahl eines Menüeintrags wird das betreffende Slot bzw. Objekt auf dem Bildschirm an einer Defaultposition dargestellt (sofern es nicht schon auf

dem Bildschirm sichtbar ist) und zum selektierten Bildschirmobjekt gemacht. Es ist mit dem bisherigen selektierten Bildschirmobjekt durch eine Linie (Bestandteil eines Pfeils) verbunden.

See Anwendungsfunktion

Die Funktion See erlaubt die Selektion von Objekten, die von dem aktuell selektierten Objekt oder Slot aus nicht direkt über die Step-Funktion erreichbar sind. Ansonsten ist die Wirkung ähnlich wie bei der Step-Funktion. Nach der Aktivierung des See-Buttons erscheint ein Menü von Wahlmöglichkeiten:

ShowObject = Nach Auswahl dieser Menüzeile kann der Name eines Objekts eingegeben werden, das als nächstes dargestellt und selektiert wird.

SeeClass Die Klasse des selektierten Objekts wird als nächste dargestellt und selektiert. (Nicht anwendbar für Slots)

SeeSlot es kann aus einer Liste von Slotbeschreibungen eine ausgewählt werden, die als nächste dargestellt und selektiert wird. (Nur anwendbar für Klassen)

SeeMethod es kann aus einer Liste von Methoden eine ausgewählt werden, die als nächste dargestellt und selektiert wird. (Nur anwendbar für Klassen)

SeeSuperclass es kann aus einer Liste von Superklassen eine ausgewählt werden, die als nächste dargestellt und selektiert wird. (Nur anwendbar für Klassen). Analog wirken *SeeSubclass*, *SeeHierarchie* und *SeeInstance*.

KnowledgeBase Anwendungsfunktion

Diese Funktion bringt ein Menü von Kommandos zur Verwaltung von Wissensbasen auf den Bildschirm:

CurrentKnowledgeBase =
Es kann der Name einer Wissensbasis eingegeben werden, die zur aktuellen Wissensbasis gemacht werden soll. Alle ab jetzt vom Benutzer innerhalb dieses ZOO-Fensters erzeugten Objekte gehören der aktuellen Wissensbasis an. Falls die angegebene Wissensbasis noch nicht existiert, werden sie und die sie visualisierende ZOO-Wissensbasis erzeugt. Sofern die Wissensbasis oder die zugehörige ZOO-Wissensbasis eine externe Filerepräsentation besitzen, werden diese geladen.

SelectKnowledgeBaseFromMenu
Unter den im System bekannten Wissensbasen kann eine ausgewählt und zur aktuellen gemacht werden.

SaveKnowledgeBase
Die aktuelle Wissensbasis wird in einer externen Textrepräsentation auf einer (im Normalfall) gleichnamigen UNIX-Datei ausgelagert. Die zugehörige ZOO-Wissensbasis wird auf einer Datei abgelegt, welche die zusätzliche Namens-Extension ".zoo" trägt.

LoadKnowledgeBase
Die aktuelle Wissensbasis und/oder die zugehörige ZOO-Wissensbasis werden von ihrer externen Filerepräsentation überladen.

DeleteKnowledgeBase
Die aktuelle Wissensbasis und alle ihr angehörigen Objekte sowie die zugehörige ZOO-Wissensbasis und deren Interaktions- und Darstellungsobjekte werden gelöscht.

Icon Anwendungsfunktion

Diese Funktion ermöglicht es, die Darstellung eines Icons zu verändern. Es erscheint ein Menü von Manipulationsmöglichkeiten. Für Objekte und Slots gibt es unterschiedliche Menüs, da sich viele der Kommandos auf die Gestaltung von Piktogrammen beziehen und daher nur für Objekte sinnvoll sind.

IconChar = Das verwendete graphische Symbol kann durch Eingabe eines Namens verändert werden. Voraussetzung ist, daß ein graphisches Symbol dieses Namens mit Hilfe des Character-Editors [Maier 85] erzeugt wurde.

SelectIconFromMenu
Ein graphisches Symbol kann aus einem Menü gewählt werden.

SetTextFrame Das Textfeld eines Piktogramms kann durch Angabe seiner Eckpunkte mit Hilfe der Maus definiert werden.

MoveTextFrame Das Textfeld eines Piktogramms mit Hilfe der Maus innerhalb des Piktogramms bewegt werden.

CenterText Die Inschrift des Piktogramms erscheint zentriert im Textfeld.

UncenterText Die Inschrift des Piktogramms erscheint linksbündig im Textfeld.

RemovePicture Das Icon wird gelöscht. Das von ihm dargestellte Objekt existiert jedoch weiter.

Link Anwendungsfunktion

Mit Hilfe dieser Funktion kann man Relationen zwischen Objekten definieren. Die Form des Maus-Cursors verändert sich zu einem Zielkreuz. Vom aktuell selektierten Icon aus zur momentanen Position des Maus-Cursors ist eine gerade Linie, ein sogenannter Gummi-Faden gezogen. Ein zweiter Klick auf einem anderen Icon erzeugt einen Pfeil, der beide Icons verbindet. Die beiden beteiligten Icons müssen unterschiedlicher Art sein, das eine muß ein Slot, das andere ein Objekt visualisieren.

Move Anwendungsfunktion

Die Form des Maus-Pointers verändert sich zum sogenannten *Move-Cross*, daraufhin läßt sich das aktuell selektierte Bildschirmobjekt bewegen und durch einen zweiten Klick an einer anderen Position plazieren.

Literatur

[ABC Mathematik 78] H. Gellert, H. Kästner, S. Neubert: *"Fachlexikon ABC Mathematik"*. Harri Deutsch, Thun und Frankfurt/Main, 1978.

[Arning 86] A. Arning: *"ASY - ein Ablagesystem"*. WISDOM-Forschungsbericht FB-INF-86-8, Forschungsgruppe INFORM, Universität Stuttgart, 1986.

[Backus et al. 63] J.W. Backus et al.: *"Revised Report on the Algorithmic Language ALGOL 60"*. *Communications of the ACM* 6(1), January, 1963.

[Bauer D. 84] D. Bauer: *"STRUPPI - ein Struktur-Pretty-Printer zur Darstellung von Lisp-Ausdrücken"*. Studienarbeit Nr. 385, Institut für Informatik, Universität Stuttgart, 1984.

[Bauer D. 86] D. Bauer: *"Wissensbank für Objekte - ein Werkzeug zur Verwaltung von ObjTalk-Wissensbasen"*. WISDOM-Forschungsbericht FB-INF-86-10, Forschungsgruppe INFORM, Universität Stuttgart, 1986.

[Bauer J. 85] J. Bauer: *"INFORM-Manual: Lift-Mixin"*. Institut für Informatik, Universität Stuttgart, 1985.

[Bauer, Goos 71] F.L. Bauer, G. Goos: *"Informatik - eine einführende Übersicht, Teil 1"*. Springer-Verlag, New York - Heidelberg - Berlin, 1971.

[Bobrow, Stefik 81] D.G. Bobrow, M. Stefik: *"The LOOPS Manual"*. Technical Report KB-VLSI-81-83, Knowledge Systems Area, Xerox Palo Alto Research Center (PARC), 1981.

[Bobrow, Winograd 77] D.G. Bobrow, T. Winograd: *"An Overview of KRL, a Knowledge Representation Language"*. *Cognitive Science* 1(1), pp 3-46, 1977.

[Brachmann 78] Brachmann. R. et al.: *"KL-ONE Reference Manual"*. BBN-Report 3848, BBN, July, 1978.

[Buchanan 78] B.G. Buchanan, E.A. Feigenbaum: *"DENDRAL and Meta-DENDRAL: Their applications dimensions"*. *Artificial Intelligence*, November, 1978.

[Buchanan, Shortliffe 84] B.G. Buchanan, E.H. Shortliffe: *The Addison-Wesley Series in Artificial Intelligence: "Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project"*. Addison-Wesley, Reading, Ma., 1984.

[Böcker 86] H.-D. Böcker: *"Visualisierungstechniken"*. In G. Fischer, R. Gunzenhäuser (editor), *Methoden und Werkzeuge zur Gestaltung benutzergerechter* Computersysteme. Verlag Walter de Gruyter & Co., Berlin - New York, 1986. Kapitel 7.

[Böcker, Fischer, Nieper 86] H.-D. Böcker, G. Fischer, H. Nieper: *"The Enhancement of Understanding through Visual Representations"*. In *Proceedings of the Conference on Human Factors in Computing Systems CHI '86*, pp 44-50. ACM, Boston, April, 1986.

[Clark 83] W. Clark: *"Schlaglicht"*. *Stuttgarter Zeitung*, 12. November, 1983.

[Clocksin, Mellish 84] W.F. Clocksin, C.S. Mellish: *"Programming in Prolog"*. Springer-Verlag, Berlin - Heidelberg - New York, 1984. Second Edition.

[Codd 70] E.F.Codd: *"A Relational Model of Data for Large Shared Data Banks"*. *CACM 13* (6), Juni, 1970.

[Csima 83] F. Csima: *"Gestaltung der Benutzerschnittstelle eines Informationssystems zur inhaltlichen Verarbeitung von Texten"*. Dissertation, Universität Stuttgart, Februar, 1983.

[Csima, Riekert 83a] F. Csima, W.-F. Riekert: *"D&I - Ein computerunterstütztes System zum Wissenserwerb"*. *Office* Management, pp 53-55, Sonderheft, 1983. Vortrag bei der Arbeitstagung Mensch-Maschine-Kommunikation der GMD, Bad Honnef, November 1982.

[Csima, Riekert 83b] F. Csima, W.-F. Riekert: *"Die Benutzerschnittstelle des Expertensystems D&I"*. In H. Balzert (editor), *Software-Ergonomie*. Teubner, Stuttgart, April, 1983. Vortrag bei der Software-Ergonomie-Tagung I des German Chapter of the ACM.

[Date 75] C.J. Date: *The Systems Programming Series: "An Introduction to Database Systems"*. Addison-Wesley, Reading, Massachusetts, 1975.

[Davis 82] R. Davis: *"Expert Systems: Where Are We? And Where Do We Go From Here?"*. *AI Magazine*, pp 3-22, Spring, 1982.

[Davis, Lenat 82] R. Davis, D.B. Lenat: *Advanced Computer Science Series: "Knowledge Based Systems in Artificial Intelligence"*. McGraw-Hill, New York, 1982.

[Dittrich et al. 79] K.R. Dittrich, R. Hüber, P.C. Lockemann: *"Methodenbanksysteme: Ein Werkzeug zum Maßschneidern von Anwendersoftware"*. *Informatik-Spektrum* 2, pp 194-203, 1979.

[Dröge 72] F. Dröge: *"Wissen ohne Bewußtsein"*. Athenäum, Frankfurt, 1972.

[Fabian 84] F. Fabian: *"Benutzungsanleitung für das Bitgraph-Fenstersystem"*. INFORM-Memo, Institut für Informatik, Universität Stuttgart, Februar, 1984.

[Fabian 86] F. Fabian: *"Fenster- und Menü-Systeme für die MCK"*. In G. Fischer, R. Gunzenhäuser (editor), *Methoden und Werkzeuge zur Gestaltung benutzergerechter Computersysteme*. Verlag Walter de Gruyter & Co., Berlin - New York, 1986. Kapitel 5.

[Fabian, Rathke C. 83] F. Fabian, C. Rathke: *"Menüs: Einsatzmöglichkeiten eines Fenstersystems zur Unterstützung des Mensch-Maschine-Dialogs"*. *Office Management* 31(Sonderheft), pp 42-44, April, 1983.

[Feigenbaum 77] E.A. Feigenbaum: *"The art of artificial intelligence: Themes and case studies in knowledge engineering"*. In *Proc. of the Fifth IJCAI*. Pittsburgh, PA, 1977.

[Fischer 83] G. Fischer: *"Navigationswerkzeuge in wissensbasierten Systemen"*. *Office Management* 31(Sonderheft), pp 49-52, 1983.

[Fischer 86] G. Fischer: *"Wissensbasierte Systeme und Mensch-Computer Kommunikation"*. Verlag Walter de Gruyter & Co., Berlin - New York, 1986. In Vorbereitung.

[Fischer et al. 78] G. Fischer, R. Burton, J.S. Brown: *"Analysis of Skiing as a Success Model of Instruction: Manipulating the learning environment to enhance skill acquisition"*. In *Proceedings of the Second National Conference of the Canadian Society for Computational Studies of* Intelligence. Canadian Society for Computational Studies of Intelligence, 1978.

[Fischer, Lemke, Schwab 85]
G. Fischer, A. Lemke, T. Schwab: *"Knowledge-based Help Systems"*. In L. Borman, B. Curtis (editor), *CHI-85, Human Factors in Computing Systems Conference Proceedings*, pp 161-167. ACM SIGCHI/HFS, New York, April, 1985.

[Fischer, Schneider 84a]
G. Fischer, M. Schneider: *"Knowledge-based Communication Processes in Software Engineering"*. In *Proceedings of the 7th International Conference on Software Engineering*, pp 358-368. Orlando, Florida, March, 1984.

[Fischer, Schneider 84b]
G. Fischer, M. Schneider;: *"Computer-supported Program Documentation Systems"*. In *Proceedings of INTERACT '84, IFIP Conference on Human-Computer Interaction*. IFIP, London, September, 1984.

[Frege 79]
G. Frege: *"Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens"*. Verlag von Louis Nebert, Halle a/S, 1879.

[Freksa et al. 84]
C. Freksa, U. Furbach, G. Dirlich: *"Cognition and Representation - an Overview of Knowledge Representation Issues in Cognitive Science"*. In Joachim Laubsch (editor), *8th German Workshop on AI*, pp 119 - 144. GWAI, Wingst/Stade, 1984.

[Goldberg 84]
A. Goldberg: *"SMALLTALK-80, The Interactive Programming Environment"*. Addison-Wesley, Reading, Ma., 1984.

[Gunzenhäuser 84]
R. Gunzenhäuser: *"Lernen als Dimension der Mensch-Maschine-Kommunikation"*. In H.Schauer, M.J.Tauber (editor), *Psychologie der Computerbenutzung*, pp 226-252. Oldenbourg Verlag, Wien - München, 1984. Schriftenreihe der Österreichischen Computer Gesellschaft, Bd. 22.

[Hanakata 80]
K. Hanakata: *"An Intelligent Digester for Interactive Text Processing"*. Tokio.COLING 80, 1980

[Herczeg 85a]
M. Herczeg: *"Konzeption einer objektorientierten Benutzerschnittstelle"*. Diskussionsunterlagen für den WISDOM-Arbeitskreis MCK, 1985.

[Herczeg 85b]
M. Herczeg: *"INFORM-Manual: Icons"*. Institut für Informatik, Universität Stuttgart, 1985.

[Herczeg et al. 85] M. Herczeg, D. Maier, C. Rathke, W.-F. Riekert: *"Vom Dialogsystem zur wissensbasierten Mensch-Computer-Kommunikation"*. WISDOM-Forschungsbericht FB-INF-85-25, Institut für Informatik, Universität Stuttgart, 1985.

[Hewitt 77] C. Hewitt: *"Viewing Control Structures as Patterns of Passing Messages"*. *Artificial Intelligence Journal* 8, pp 323-364, 1977.

[Hutchins et al. 86] E.L. Hutchins, J.D. Hollan, D.A. Norman: *"Direct Manipulation Interfaces"*. In D.A. Norman, S. Draper (editors), *User Centered System Design: New Perspectives on Human-Computer Interaction.* Lawrence Erlbaum Associates Ltd., 1986.

[Häfner 86] K. Häfner: *"Desoxyribonukleinsäure - Genialer Informationsspeicher des genetischen Informationsverarbeitungssystems"*. *Computer Magazin* (5), pp 44-45, Mai, 1986.

[IBM 74] IBM: *"HIPO - A Design Aid and Documentation Technique"*. Technical Report GC20-1851-0, IBM White Plains, 1974.

[Ingalls 81] D.H.H. Ingalls: *"Design Principles Behind Smalltalk"*. *BYTE* 6(8), pp 286-298, August, 1981.

[KEE 85] *"KEE Software Development System User's Manual"*. IntelliCorp, 1985.

[Kernighan, Ritchie 83] B.W. Kernighan, D.M. Ritchie: *"Programmieren in C"*. Carl Hanser Verlag edition, 1983.

[Lemke 85] A. Lemke: *"ObjTalk84 Reference Manual"*. Technical Report CU-CS-291-85, University of Colorado, Boulder, 1985.

[Liebermann, Hewitt 80] H. Liebermann, C. Hewitt: *"A Session With TINKER: Interleaving Program Testing With Program Writing"*. Artificial Intelligence Laboratory and Laboratory for Computer Science Massachusetts Institute of Technology.1980

[Maier 85] D. Maier: *"INFORM-Manual: Character-Editor, Version 3.1"*. WISDOM-Forschungsbericht FB-INF-85-26, Institut für Informatik, Universität Stuttgart, 1985.

[Marx 83] Dr. Marx: *"Informationssysteme der 5ten Computergeneration"*. Bonn, 25. Mai, 1983.

[Michalski, Carbonell, Mitchell 83] R.S. Michalski, J.G. Carbonell, T.M. Mitchell (editors). *Machine Learning*. Tioga Publishing Company, P.O.Box 98, Palo Alto, CA 94302, 1983.

[Minsky 75] M. Minsky: *"A Framework for Representing Knowledge"*. In P.H. Winston (editor), *The Psychology of Computer Vision*, pp 211-277. McGraw Hill, New York, 1975.

[Moore, Newell 74] J. Moore, A. Newell: *"How can MERLIN understand?".L.W. Gregg(ed.): Knowledge and Cognition.* Erlbaum, Potomac, 1974

[Moritz 83] H. Moritz: *"Umsetzung wahrnehmungspsychologischer Erkenntnisse für die Informationsgestaltung am Bildschirm (Maskengestaltung)"*. In H. Balzert (editor), *Software-Ergonomie*, pp 98 - 113. April, 1983.

[Nake 85] F. Nake: *"Computer - Daten - Wissen - Menschen"*. Studie im Auftrag des Projekts INFORM der Universität Stuttgart, 1985.

[Nassi, Shneiderman 73] I. Nassi, B. Shneiderman: *"Flowcharting Techniques for Structured Programming"*. *ACM SIGPLAN Notices* 8(8), pp 12-26, 1973.

[Neuhold 77] E. Neuhold: *"Informationssysteme I"*. Universität Stuttgart, Institut fr Informatik.üVorlesungsskript, 1977

[Newell 69] A. Newell: *"Heuristic programming; ill structured problems"*. In Aronofsky (editor), *Progress in Operations Research, vol. 3*, pp 362-414. Wiley, New York, 1969.

[Nieper 83] H. Nieper: *"KÄSTLE: Ein graphischer Editor für LISP-Datenstrukturen"*. Studienarbeit Nr. 347, Institut für Informatik, Universität Stuttgart, 1983.

[Oberquelle et al. 83] H. Oberquelle, I. Kupka, S. Maaß: *"A view of human-machine communication and co-operation"*. *Int. J. of Man-Machine Studies* 19, pp 309-333, 1983.

[Piaget 72] Piaget: *"The Psychology of Intelligence"*. Littlefield, Adams & Co., Inc., Totowa, N.J., USA, 1972.

[Quillian 68] M.R. Quillian: *"Semantic Memory"*. In M. Minsky (editor), *Semantic Information Processing*, chapter 4, pp 227-270. The MIT Press, Cambridge, Ma., 1968.

[Rathke C. 86] C. Rathke: *"ObjTalk. Repräsentation von Wissen in einer objektorientierten Sprache"*. Dissertation, Fakultät für Mathematik und Informatik der Universität Stuttgart, Oktober, 1986.

[Riekert 86a] W.-F. Riekert: *"INFORM-Manual: Nets"*. WISDOM-Forschungsbericht FB-INF-86-7, Institut für Informatik, Universität Stuttgart, 1986. in Vorbereitung.

[Riekert 86b] W.-F. Riekert: *"INFORM-Manual: Sheets"*. WISDOM-Forschungsbericht FB-INF-86-6, Institut für Informatik, Universität Stuttgart, 1986.

[Riekert 86c] W.-F. Riekert: *"Der graphische Wissenseditor ZOO - ein Metasystem zur Visualisierung und Manipulation von Wissensbasen"*. WISDOM-Forschungsbericht FB-INF-86-9, Institut für Informatik, Universität Stuttgart, 1986.

[Riekert 86d] W.-F. Riekert: *"Systemkomponenten zum Wissenserwerb"*. In G. Fischer, R. Gunzenhäuser (editor), *Methoden und Werkzeuge zur Gestaltung benutzergerechter Computersysteme*. Verlag Walter de Gruyter & Co., Berlin - New York, 1986. Kapitel 9.

[Roberts, Goldstein 77] R.B. Roberts, I.P. Goldstein: *"The FRL-Manual"*. Technical Report MIT-AI Memo 409, MIT, 1977. Cambridge, Ma.

[Robertson et al. 81] G. Robertson, D. McCracken, A. Newell: *"The ZOG Approach to Man-Machine Communication"*. *International Journal of Man-Machine Studies* 14, pp 461-488, August, 1981. Carnegie-Mellon University, Computer Science Department, Pittsburgh.

[Rosenberg 77] S.T. Rosenberg: *"Frame based Text Processing"*. AI-Memo 431, MIT, 1977.

[Schank, Abelson 77] R.C. Schank, P.R. Abelson: *"Scripts, Plans, Goals and Understanding"*. Lawrence Erlbaum, Hillsdale, New York, 1977.

[Schwab 84] T. Schwab: *"AKTIVIST: Ein aktives Hilfesystem für den bildschirmorientierten Editor BISY"*. Diplomarbeit Nr. 232, Institut für Informatik, Universität Stuttgart, 1984.

[Shneiderman 83] B. Shneiderman: *"Direct Manipulation: A Step Beyond Programming Languages"*. *IEEE Computer* 16(8), pp 57-69, August, 1983.

[Shortliffe 76] E.H. Shortliffe: *Artificial Intelligence Series.* Volume 2: *"Computer-Based Medical Consultations: MYCIN"*. Elsevier, New York - Amsterdam, 1976.

[Simon 73] H.A. Simon: *"The Structure of ill-structured Problems"*. *Artificial Intelligence* (4), 1973.

[Simon 81] H.A. Simon: *"The Sciences of the Artificial"*. MIT Press, Cambridge, Ma., 1981.

[Smith et al. 82] D.C. Smith, Ch. Irby, R. Kimball, B. Verplank: *"Designing the Star User Interface"*. *BYTE*, April, 1982.

[Staufer 85] M. J. Staufer: *"Piktogramme für Bürocomputer"*. WISDOM-Forschungsbericht FB-TA-85-6, Triumph-Adler AG, Basisentwicklung, Juni, 1985.

[Stefik 79] Stefik, M.: *"An examination of a frame-structured representation system"*. In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence IJCAI-79*, pp 845-852. Tokyo, Japan, 1979.

[Stefik, Bobrow 85] M. Stefik, D.G. Bobrow: *"Object-Oriented Programming: Themes and Variations"*. *AI Magazine* 6(4), pp 40-62, Winter, 1985.

[Stelovsky 84] J. Stelovsky: *"XS-2: The User Interface of an Interactive System"*. Dissertation 7425, ETH Zürich, 1984.

[Tesler 81] L. Tesler: *"The Smalltalk Environment"*. *BYTE* 6(8), pp 90-147, August, 1981.

[Tou, Williams 82] F.N. Tou, M.D. Williams: *"RABBIT: An Intelligent Database Assistant"*. Technical Report, Xerox Palo Alto Research Center, Palo Alto, California, 1982.

[UNIX MAIL 83] 4th Berkeley Distribution: *"MAIL(1)"*. UNIX Programmer's Manual edition, University of Berkeley, Ca, 1983.

[Wertheimer 23] M. Wertheimer: *"Untersuchungen zur Lehre von der Gestalt"*. *Psychol. Forsch.* 4, pp 301-350, 1923.

[Zloof 75] M.M. Zloof: *"Query-by-Example"*. Proceedings of the International Conference on Very Large Databases.1975

Lebenslauf

Name:	Wolf-Fritz Johannes Riekert	
Adresse:	Rotenwaldstraße 19 7000 Stuttgart 1	
Geboren:	12. Juni 1950 in Stuttgart	
Familienstand:	verheiratet seit 10. April 1984 mit Elisabeth Riekert, geb. Wehe	
Schule:	1957-1961	Mühlbachhof-Grundschule Stuttgart
	1961-1969	Eberhard-Ludwigs-Gymnasium Stuttgart
	1969	Abitur (ebenda)
Studium:	1969	Immatrikulation an der Universität Stuttgart in Mathematik und Physik
	1971	Zwischenprüfung Lehramt (Mathematik und Physik)
	1973	Diplomvorprüfung Mathematik
	1976	Diplomhauptprüfung Mathematik
	Thema der Diplomarbeit aus dem Gebiet der Topologie: "Limesräume von Funktionen und der Satz von Arzela und Ascoli"	
Beruf:	1977-1983	Angestellter bei der Firma Informatik GmbH, Stuttgart; Systemsoftware-Entwicklung für Mikroprozessoren, zuletzt als Gruppenleiter
	seit 1984	Wissenschaftlicher Angestellter in der Abteilung Dialogsysteme, Institut für Informatik, Universität Stuttgart

Index