Kapitel 7

Expertensysteme, Planen und Problemlösen

Kapitelherausgeber: Ipke Wachsmuth

Mit diesem Kapitel wird in zwei Abschnitten ein vertiefter Einstieg in das Gebiet wissensbasierter Informationsverarbeitung mit problemlösenden Systemen gegeben. Das Kapitel führt in das Gebiet der Expertensysteme ein, die das derzeit erfolgreichste Anwendungsgebiet der Künstlichen Intelligenz darstellen. Ein Expertensystem ist ein Computerprogramm, mit dem versucht wird, anhand von erhobenen Gedankengängen und Erfahrungen von Experten eines bestimmten Fachgebiets ein maschinelles System zu konstruieren, das Anwendern Aspekte einer Problemlösungskompetenz zur Verfügung zu stellt.

Der erste Abschnitt über „Expertensysteme und Wissensmodellierung" wendet sich in erster Linie an ein Publikum, das sich nicht nur mit der technischen Seite des Gebietes, sondern auch mit den Hauptthemen bei der Durchführung eines Expertensystem-Projektes vertraut machen möchte. Deshalb behandelt er neben Grundtechniken der Wissensdarstellung und Problemlösung in Expertensystemen auch Vorgehensweisen und Probleme bei der Erhebung und Modellierung von Expertenwissen. Zur Sprache kommen unterschiedliche Eigenarten von Fach- und Erfahrungswissen und auch die Besonderheit von Expertenwissen. Anschließend wird der Prozeß der Expertensystementwicklung mit seinen methodischen Vorgehensweisen dargestellt. Damit versteht sich der Abschnitt auch als Beitrag zur Ausbildung von Knowledge Engineers.

Der zweite Abschnitt über „Planen und Konfigurieren" greift ein Anwendungsthema auf, in dem über die letzten Jahre viel Kompetenz und Erfahrung gewonnen wurde, das sich aber im Gegensatz zur verfügbaren Literatur über Diagnose-Systeme bislang kaum in dieser Geschlossenheit dargestellt findet. Planen bedeutet das Erstellen einer Struktur von Aktionen, deren Ausführung ein gegebenes Problem löst, und Konfigurieren das Erstellen einer Struktur aus Komponenten, die gegebenen Bedingungen genügt. Planen und Konfigurieren sind verwandte Aktivitäten, da sowohl die erzeugten Strukturen als auch die verwendeten Methoden einander ähnlich sind. Im vorliegenden Beitrag werden anhand

eines gemeinsamen Beispiels diese beiden Gebiete skizziert und ihre Gemeinsamkeiten – wie die Verwaltung von Abhängigkeiten zwischen Merkmalen – und Unterschiede – wie die Bedeutung zeitlicher und räumlicher Beziehungen – herausgearbeitet. In beiden Bereichen wird je ein Kernproblem detaillierter behandelt: beim Konfigurieren das Kontrollproblem und beim Planen die Semantik von Planungsformalismen.

Die beiden Abschnitte dieses Kapitels gehen auf Kurse zurück, die auf den jährlich stattfindenden Frühjahrsschulen über Künstliche Intelligenz durchgeführt worden sind. Anregungen und Kritiken von Kursteilnehmern sind weitgehend aufgenommen worden, wodurch wir dem Ziel dieses Kapitels, bei beschränktem Seitenumfang einen tiefergehenden Einblick in das Gebiet zu geben, hoffentlich nahegekommen sind.

7.1 Expertensysteme und Wissensmodellierung

Josef Meyer-Fujara, Frank Puppe und Ipke Wachsmuth

7.1.1 Expertensysteme und Expertenwissen

7.1.1.1 Motivation

Mit Expertensystemen wird versucht, Gedankengänge und Erfahrungen von Experten bestimmter Fachgebiete auf eine Menge von formalisierten, maschinenverarbeitbaren Operationen abzubilden, um Aspekte einer Problemlösungskompetenz zu reproduzieren und Anwendern zur Verfügung zu stellen. Von der Formalisierung von Expertenwissen verspricht man sich:

- Explizierung und Überprüfbarkeit von Fachwissen
- Unterstützung und Ergänzung menschlicher Fachtätigkeit
- Begünstigung einer Wissensevolution
- Vorteile bei der fachlichen Ausbildung
- Möglichkeit einer Konservierung von Fachwissen
- technische Hilfsmittel für die Wissensverarbeitung und -nutzung

Eine schwierige Aufgabe ist die Modellierung von Wissen, welches Experten benutzen. Noch erschwert wird diese Aufgabe dadurch, daß es sich um eine Rekonstruktion von – überwiegend impliziten – Modellen, die Experten bei ihrer Fachtätigkeit benutzen, handelt (siehe Abb. 7.1). Nicht alle Vorgehensweisen der Wissensmodellierung machen das sogenannte mentale konzeptuelle Modell explizit; auf diesen Punkt richten sich neuere modellbasierte Vorgehensweisen, auf die später eingegangen werden wird.

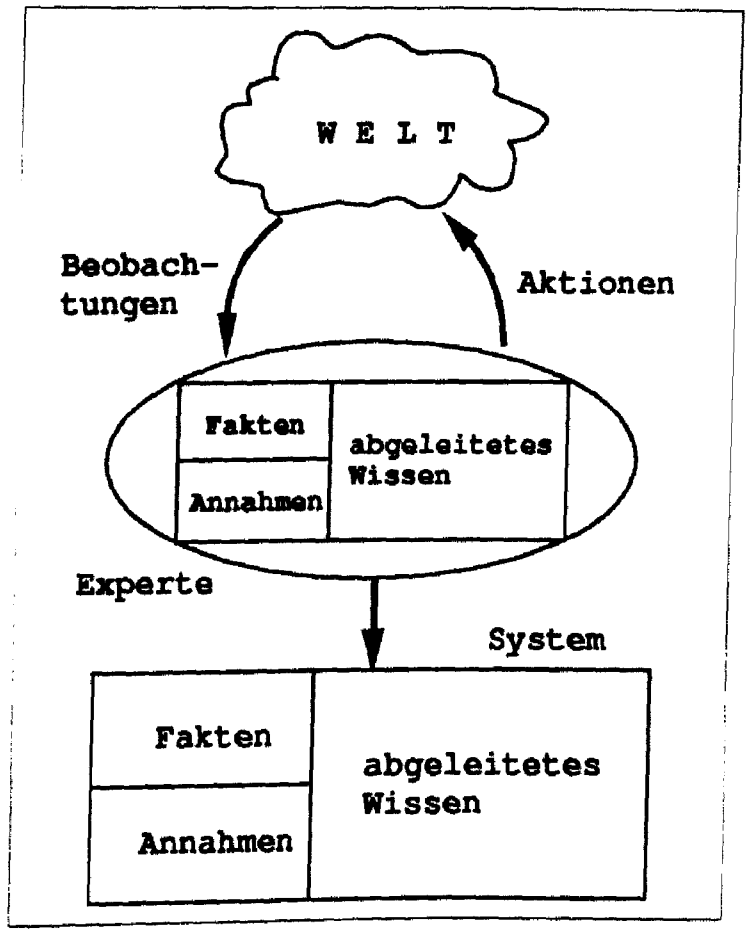

Abbildung 7.1: Mentale konzeptuelle Modelle, über welche Experten für Problemlösungen in ihrer Anwendungswelt verfügen, sind in operationale Modelle für Expertensysteme zu überführen.

7.1.1.2 Einordnung und Abgrenzung

Wissensmodellierung ist zunächst eine Teilaufgabe der *Wissensrepräsentation.* Unter einer Wissensrepräsentation versteht man ein symbolisch dargestelltes Modell eines Wissensbereichs aus Objekten, Fakten und Regeln in operationaler Form für einen Handlungsträger mit symbolverarbeitender Kompetenz. Das *Gebiet* „Wissensrepräsentation" umfaßt zweierlei: (1) Die Entwicklung von Repräsentationsformalismen, die Beschreibungsmittel für die symbolische Darstellung des Wissens bestimmter Fachgebiete und Weltbereiche bereitstellen und die automatische aufgabengesteuerte Berechnung von Inferenzen über der Menge kodierten Wissens ermöglichen; (2) die Bereichsmodellierung (Modellierung von Wissensbereichen), bei der insbesondere folgende Fragen zu beantworten sind:

- *Welche* Kategorien sind zur Einordnung von Objekten und Ereignissen eines Wissensbereichs zu wählen?
- *Welche* Eigenschaften und Annahmen sind ihnen zuzuordnen?
- *Wie* werden sie untereinander in Beziehung gesetzt?
- *Welche* Folgerungen sollen im Kontext bestimmter Annahmen möglich sein?

Im Kontext von Expertensystemen wird die *Wissensmodellierung* dem sog. *Knowledge Engineering* zugerechnet. Es stellt sich als hauptsächliche Anforderung, das Wissen eines eingegrenzten Aufgabenbereichs von Fachleuten zu erheben und in einem Repräsentationsformalismus so auszudrücken, daß sich Aufgaben des Anwendungsbereichs mit maschinellen Systemen bearbeiten und lösen lassen. Wissensmodellierung ist ein hochgradig interdisziplinäres Gebiet, für das Nachbargebiete der KI (kognitive Psychologie, Linguistik, ...) ebenso relevant sind wie Methoden der KI und der „klassischen" Informatik. Das hauptsächliche Problem besteht darin, ein adäquates Modell des Anwendungsbereichs und des dort relevanten Problemlösungsprozesses zu finden, das sich mit dem mentalen Modell des Experten deckt. Aufgrund dieser Sachlage ist bereits erkennbar, daß dieses Problem nicht auf einer programmiertechnischen Ebene lösbar ist.

Was sicherlich heute und auch in Zukunft nicht geht, ist das nachstehend persiflierte Vorgehen: Finde einen Experten oder eine Expertin, schreibe nieder, was diese Person weiß, und tippe das in eins der vielen Software-Werkzeuge zur Erstellung wissensbasierter Systeme ein. Die Komplikationen einer solchen Vorgehensweise sind mehrdimensional: Es erweist sich als ausgesprochen schwierig, menschliches Wissen in einer formalen symbolischen Notation zu repräsentieren, und es mangelt immer noch an hinreichend geklärten Methoden zur formalen Erhebung des Wissens von Experten. Die Hauptschwierigkeit liegt sicherlich in der Spanne der zu erbringenden Aufgaben: Einerseits ist ein Umgang mit einem hohen Grad an Formalität in der Wissensrepräsentation erforderlich, andererseits vor allem aber auch entwickelte Fertigkeiten in der Kommunikation mit Experten. Wenige Leute bringen von ihrer beruflichen Qualifikation her beides von vornherein mit.

Dieser Abschnitt versteht sich als Beitrag zur Ausbildung von Knowledge Engineers, was als ergänzende Qualifikation von Fachexperten oder von informatisch qualifizierten Fachleuten verstanden werden sollte. Dementsprechend liegt - neben einer Einführung ins Gebiet Expertensysteme - das Hauptgewicht auf Vorgehensweisen und Problemen bei der Modellierung von Wissen für Expertensysteme (etwa zwischen Wissenserhebung und „pre-software environment"). Ein Ziel ist es vor allem, mit den Rahmenbedingungen der Aufgabe vertraut zu machen. Der Beitrag wendet sich damit in erster Linie an ein Publikum, das sich nicht nur mit der technischen Seite des Gebiets Expertensysteme, sondern auch mit den Hauptthemen bei der Durchführung eines Expertensystem-Projektes vertraut machen möchte. Die behandelten „Mitspieler" werden zunächst kurz vorgestellt:

EXPERTEN: Hierunter verstehen wir Personen, die berufliche Aufgaben bewältigen, für die man sowohl eine lange Fachausbildung als auch praktische Erfahrung benötigt. Zu den Fähigkeiten von Experten gehört, daß sie Probleme erkennen und verstehen, Probleme lösen, die Lösung erklären, die eigene Kompetenz einschätzen, Randgebiete ihres Fachs überschauen sowie Wissen erwerben und strukturieren können. Es ist möglich, daß Experten starke, aber fehlerhafte Annahmen darüber haben, wie sie zu Urteilen gelangen. Ihr Wissen kann unbewußt sein und nicht mit Worten ausdrückbar. Mit der Besonderheit von Experten und ihrem Wissen werden wir uns noch eingehender befassen.

WISSEN: Unter „Wissen" versteht sich einerseits ein Begriff für einen individuellen kognitiven Inhalt („Ermöglichungsgrund für Handlungen"), andererseits ein Begriff für ein soziales und damit kulturelles Phänomen. Mit beiden Aspekten hat man bei Expertensystemen zu tun. Arten von Wissen, die für unseren Zweck im weiteren unterschieden werden, sind theoretisches Fachwissen, Erfahrungswissen, Expertenwissen sowie Allgemein- oder Alltagswissen. Dies soll aber keine erschöpfende Taxonomie sein; es gibt weitere Dimensionen zur Unterscheidung von Wissensarten.

EXPERTENSYSTEME sind eine Form wissensbasierter Systeme (also Programme), die spezifisches Wissen und Schlußfolgerungsfähigkeiten qualifizierter Fachleute nachbilden. Ihr Anwendungsfeld ist beim Stand der Technik auf scharf abgegrenzte, schmale Gebiete beschränkt. Ihre Arbeitsweise beruht im Regelfall auf interaktiver oder automatischer Lösung von Problemen durch Symbolmanipulation in mehrstufigen Such- und Entscheidungsprozessen. Expertensysteme und grundlegende Techniken für ihre Realisierung sind Gegenstand von Teil 7.1.2 des Beitrags; die Entwicklungsmethodik wird in Teil 7.1.5 behandelt.

KNOWLEDGE ENGINEERS („Wissensingenieure") sind die Leute, die Wissen von erfahrenen Fachleuten auf ein Computersystem übertragen, welches in der Lage ist, einen Bereich zu repräsentieren, darin Schlüsse durchzuführen und in diesem Bereich (Routine-) Probleme zu lösen. Wissensingenieure haben bei der Entwicklung von Expertensystemen etwa die Funktion von Systemanalytikern bei klassischen Software-Systemen. Die Spannweite ihrer Tätigkeiten ist enorm; sie umfaßt die Analyse und Bewertung der kognitiven Fähigkeiten von Experten, einblickhaftes Verständnis des jeweiligen Anwendungsbereichs, die Erstellung eines Wissensmodells und schließlich systemnahe Entscheidungen bei der Im-

plementierung der Wissensbasis. Bei entsprechender Schulung kann die Rolle des Knowledge Engineers auch vom Domänenexperten selbst übernommen werden. Mit Methodologien zur Wissensmodellierung werden wir uns in Teil 7.1.3 des Beitrags auseinandersetzen.

EXPERTENSYSTEMWERKZEUGE sind Software-Werkzeuge zur Erstellung von Expertensystemen, die Verallgemeinerungen und Formalisierungen bewährter Konzepte von Expertensystemen bereitstellen. Insbesondere die Ansätze und Möglichkeiten verschiedener sog. hybrider Expertensystemwerkzeuge, die sich teils auf eigene Erfahrungen stützen, sind Gegenstand von Teil 7.1.5 des Beitrags; Spezialwerkzeuge für die Wissensakquisition werden in Teil 7.1.3 angesprochen.

7.1.1.3 Expertensysteme – Einsatzfelder und Arbeitsprinzip

Die wichtigsten Einsatzfelder für Expertensysteme liegen in der heuristischen Diagnostik (etwa für Wartung und Reparatur komplizierter Geräte, für die Prozeßüberwachung) und in der Konfigurierung, also dem Zusammensetzen komplexer Apparate aus Komponenten. Mittlerweile werden etliche solche Systeme routinemäßig eingesetzt. Außer im technischen Bereich werden Einsatzmöglichkeiten vor allem auch in der Medizin (z.B. zur Unterstützung der medizinischen Entscheidungsfindung), in der Analysetechnik (z.B. bei Meßgeräten, die ihre Daten selbst interpretieren) und im Bereich von Dienstleistung und Ausbildung gesehen. Mit Ausnahme von hochspezialisierten Systemen ist jedoch der Sprung in die Praxis bisher nicht gelungen.

Leistungsfähige realisierte Vorzeigesysteme können u.a. folgendes vorweisen: Das System R1/XCON [McDermott, 1982; McDermott, 1984] konfiguriert seit langem die VAX-Computer von DEC nach Kundenvorgaben; QMR/INTERNIST [Miller *et al.*, 1982] hat Wissen über den gesamten Bereich der inneren Medizin und bietet Schutz vor dem Übersehen von Diagnosen; es ist auch als einfaches tutorielles System verfügbar. Eine Übersicht über betriebliche Expertensysteme gibt [Mertens *et al.*, 1990]. Sachkenner sind sich heute aber weitgehend darin einig, daß kein System einen Experten wirklich ersetzen können wird, da das nötige Allgemeinwissen und (besonders in der Medizin) die Fähigkeit zur ganzheitlichen Beobachtung und Beurteilung von Fällen fehlt.

Die Anwendungsbereiche von Expertensystemen sind „diffus“ im Vergleich zu denen klassischer datenverarbeitender Systeme, es gibt statt einer einheitlichen Theorie viel fragmentarisches, empirisches „heuristisches“ Wissen, das abhängig von den aktuellen Daten eingesetzt wird. Die Systeme verwenden deshalb allgemeine Lösungsstrategien, die durch Wissen aus dem Anwendungsbereich gesteuert werden. Ein solches *mustergesteuertes Inferenzsystem* (pattern-directed inference system [Jackson, 1986]) besteht aus

1. einer Sammlung relativ unabhängiger „Module“ (z.B. Regeln, Frames, Constraints, Logik-Klauseln), die für passende aktuelle Daten aktiviert werden können,
2. einer dynamischen Datenstruktur, die durch diese Module inspiziert oder verändert werden kann und

3. einer zyklischen Steuerung für die Aktivierung der Module (Interpreter).

Den Kontrollfluß bestimmt der Interpreter in Abhängigkeit von den aktuellen Daten; der Experte legt nur fest, was in bestimmten Situationen zu tun ist.

Als Beispiele für solche Module führen wir hier zwei unterschiedlich geartete *Regeln* (mit Implikationen bzw. mit Handlungen) an:

1. *Wenn* 1. Nackensteife und
   2. hohes Fieber und
   3. Bewußtseinstrübung
   zusammentreffen,
   dann besteht Verdacht auf Meningitis.
2. *Wenn* Verdacht auf Meningitis besteht,
   dann bestimme Liquorstatus.

Zusammenfassend gesagt arbeitet ein Expertensystem mit einer viele solcher Module umfassenden *Wissensbasis*, die bereichsbezogenes Expertenwissen, fallspezifisches Wissen sowie Zwischenergebnisse und Problemlösungen darstellt. Eine *Problemlösungskomponente* interagiert mit diesen Wissensteilbeständen und erzeugt und verwaltet die Zwischenergebnisse und Lösungen.

7.1.1.4 Wissen – Fachwissen – Erfahrungswissen

Angesichts der häufig formulierten Kritik über die Adäquatheit der Wissensauffassung, die sich in realisierten Expertensystemen widerspiegelt, wird sicherlich mit einigem Recht behauptet, daß man sich in der Forschung über Künstliche Intelligenz in unzureichendem Maße mit natürlicher Intelligenz auseinandergesetzt habe. Dies gibt Anlaß, hier vor allen Anläufen zur Wissensmodellierung die Natur von Expertenwissen eingehender zu betrachten. Expertenstudien in der psychologischen Problemlöseforschung[1] sind u.a. für die Gebiete Physik, der Medizin und auch der Wissensvermittlung (Lehrer) durchgeführt worden. Diese Studien sind gekennzeichnet durch die Verwendung von Problemen, zu deren Lösung Fach- und Hintergrundwissen über Realitätsbereiche erforderlich ist. Grob zusammengefaßt stellte sich heraus: Gegenüber weniger erfahrenen Fachleuten haben Experten qualitativ unterschiedliches Wissen; dies kann betreffen:

- Inhalte des Wissens (was für Wissen)
- Quantität (mehr Wissen)
- sachliche Richtigkeit und Angemessenheit
- Abstraktheit der Begriffe

[1] Die zusammenfassenden Darstellungen in diesem Abschnitt profitieren wesentlich aus einer Reihe von Seminaren über Expertenwissen, die 1989/90 an der Universität Bielefeld vom dritten Autor zusammen mit dem Psychologen R. Bromme durchgeführt worden sind. Eine gute Zusammenstellung der Expertenstudien findet sich in [Bromme, 1992].

- Kohärenz des Wissens
- fall- oder aufgabenbezogene Organisation des Wissens
- Prozeduralisierung von Wissen
- Verknüpfung des Wissens über Sachverhalte mit dem Wissen über Lösungsschritte

Bei den Expertenstudien zeigte sich auch, daß zwischen dem in Büchern niedergeschriebenen *Fachwissen* und der Expertise, die sich als Folge langjähriger beruflicher *Erfahrung* ausbildet, kein einfacher Zusammenhang besteht: Gleiche Erfolge in der Bewältigung einer Problemlösung sind auch bei unterschiedlichem theoretischem Fachwissen möglich, andererseits sind deutliche Zusammenhänge zwischen Fachwissen und Problembearbeitung nachweisbar. Expertenwissen umfaßt einerseits weniger, andererseits mehr als das theoretische Kernwissen des Fachs. Und es ist oft anders strukturiert. Im Vergleich zu guten „Novizen" zeigt sich eine andere Qualität des Fachwissens von Experten, was die Exaktheit, Detailliertheit und Differenziertheit anbelangt.

Untersucht wurde ferner die Frage, wie *spezifisch* Expertenwissen ist. Am Beispiel der diagnostischen Problemlösung in der Medizin zeigte sich dabei, daß unabhängig vom Expertise-Grad sich alle Mediziner nach dem hypothetisch-deduktiven Verfahren richten, d.h. aufgrund weniger Symptome Hypothesen generieren und dann versuchen, diese durch weitere Daten zu untermauern. Ein anderes Grundprinzip ist das differential-diagnostische Vorgehen, bei dem jeweils wenige Hypothesen vergleichend berücksichtigt werden. Darüber hinaus scheint es keine Verallgemeinerung medizinischer Argumentation zu geben, die ganz ohne bereichsspezifisches Wissen auskommen würde.

Die als *Erfahrungswissen* bezeichnete Art von Wissen wird durch praktische Tätigkeit, das Erleben von vielen Situationen und Verläufen erworben. Am Beispiel der Medizin zeigt sich etwa, daß durch lange klinische Erfahrung die interne Struktur der Expertenmodelle von Krankheiten sich auf die natürliche Variation der Befunde einstellt. Eine Vielzahl von Situationsmerkmalen kann ganzheitlich erfaßt werden und führt zu raschen Einordnungen („klinischer Blick"). Es findet eine anforderungsspezifische Umorganisation des Lehrbuchwissens statt. Fälle spielen hierbei eine wichtige Rolle. Zum Bestand des Fachwissens kommt ein Bestand an in der Tätigkeit demonstrierbaren Fertigkeiten hinzu (Prozeduralisierung).

7.1.1.5 Besonderheit von Expertenwissen

Die Besonderheit von Expertenwissen betrifft zum einen die bei der Aufgabenbewältigung eingenommene *Perspektive*: Einstellungen und Wertungen führen zu einer gerichteten Wahrnehmung von Situationen. Bereits hierdurch, nicht erst in der Anwendung des Wissens auf eine – wie immer geartete – interne Repräsentation der aktuellen Problemsituation wird ein Teil der Problemlösung erreicht. Andererseits zeigt sich eine *kognitive Gliederung*

des Wissens: Neben dem taxonomischen Aufbau des Fachwissens gibt es andere Kristallisationspunkte, um die herum Wissen organisiert ist: In der Medizin sind dies erlebte Fälle, diagnostische und therapeutische Situationen. Unter welchen Gesichtspunkten diese verschiedenen Informationen zusammengefaßt sind, ist noch empirisch ungeklärt. Jedenfalls umfaßt medizinisches Wissen mehr als Beziehungsregeln zwischen Symptomen und Krankheitsbegriffen: Es wird ein mehr oder weniger kohärentes Bild über die jeweilige Krankheit bzw. den jeweiligen Fall entwickelt (sog. disease models).

Das implizite Wissen von Experten, oder besser die Besonderheit des Expertenhandelns, äußert sich in der Beobachtung von Experten vor allem in der Geschwindigkeit und der Qualität (Flüssigkeit): Das rasche Handeln setzt Wissen voraus, aber im Moment des Handels wird kein bewußter Bezug auf das Wissen erlebt. Hier stößt man an eine methodische Schwierigkeit der Expertenforschung: Experten handeln „gekonnt", können aber das Wissen, das dem Können zugrunde liegt, häufig nicht angeben. Lassen sich ihr Faktenwissen und ihre theoretische Kenntnisse noch relativ leicht explizieren, so sind ihre handlungsleitenden Einstellungen, Annahmen und Werte erst im Gebrauch zu erkennen und müssen begrifflich rekonstruiert werden. Diese Schwierigkeiten äußern sich ebenfalls bei der Wissenserhebung für Expertensysteme.

An dieser Stelle muß auf die von Dreyfus & Dreyfus formulierten Thesen verwiesen werden (siehe [1987, Kap. 1]), die ein Fünf-Stufen-Modell des Fertigkeitenerwerbs postulieren, das von der Stufe des Anfängers über abgestufte Grade von Kompetenz mit analytischer Bewältigung von Problemsituationen bis zum Niveau des Experten mit erfahrungsbasierter, intuitiver Entscheidungsfindung führt. Ihre zentrale These besagt, daß es einen prinzipiellen Unterschied zwischen Wissen („know-that") und Können („know-how") gibt, was bedeutet, Können ist nicht unbewußtes Wissen. Da mit derzeitigen Expertensystemen nur Wissen verarbeitet werden könne, seien sie prinzipiell nicht in der Lage, volle Problemlösefähigkeit von Experten zu simulieren. Als Gegenposition dazu wird formuliert, Können sei unbewußtes Wissen und könne im Prinzip mit den gleichen Techniken repräsentiert werden. Im Falle, daß die Dreyfus-Thesen zutreffen, wäre es – jedenfalls mit den Methoden der „klassischen" symbolverarbeitenden KI – nicht möglich, daß Expertensysteme über das Kompetenz-Niveau hinausgelangen.

7.1.1.6 Arten von Expertenwissen

Auch bei spezifischem Expertenwissen sind verschiedene Arten unterschieden worden. Dazu gehören assoziatives (heuristisches) Wissen, kausales (modellbasiertes) Wissen, statistisches und schließlich fallvergleichendes Wissen. Es ist wichtig einzusehen, daß die Unterscheidung solcher *Wissensarten* von vornherein nichts mit den zur Wissensmodellierung herangezogenen Repräsentationsformalismen zu tun hat (also nicht etwa: Regeln stehen für assoziatives Wissen etc.). „Gelbfärbung der Augen deutet auf Leberkrankheit" ist eine empirisch gerechtfertigte Heuristik zur diagnostischen Hypothesenfindung, während die Aus-

sage „Lebererkrankung [erhöht den Bilirubinspiegel und] bewirkt [so] eine Gelbverfärbung der Augen" ein Schritt zur kausalen Modellierung ist. Gleichwohl können beide Arten von Zusammenhängen in Regelform ausgedrückt werden. Die Art des in einem Expertensystem eingesetzten Wissens kann von der wahrgenommenen Teilaufgabe bestimmt sein: für schnelle Hypothesenfindung eignet sich heuristisches Wissen besser als kausales, welches wiederum für Erklärungen einer Problemlösung vorzuziehen ist.

Expertenwissen des Menschen ist dabei im Vergleich zu Expertensystemen in viele Schichten von zunehmend allgemeinem Wissen und Erfahrungen eingebettet; wesentliche der anfangs genannten Fähigkeiten von Experten (z.B. Einschätzung der eigenen Kompetenz) sind ohne dieses Rahmenwissen kaum vorstellbar. Ein in der Künstlichen Intelligenz noch weitgehend ungelöstes Problem ist die Modellierung des von Menschen auch bei der Lösung fachspezifischer Aufgaben implizit herangezogenen Allgemein- und Alltagswissens (Wissen über Raum und Zeit, über physikalische Zusammenhänge in erfahrbaren Phänomenbereichen, über Motive und Überzeugungen anderer etc.). Bisher sind hier keine nennenswerten Erfolge erzielt worden; dort, wo Expertensystemprojekte erfolgreich gewesen sind, gelang dies durch Beschränkung auf ein Spezialgebiet. Der auch bei hoher spezifischer Problemlöseleistung sich ergebende scharfe Kompetenzabbruch bei nicht eindeutig für das System spezifizierten Aufgabenstellungen (*„Kliff-und-Plateau-Effekt"*) ist ein grundsätzliches Problem der Expertensystemtechnik. Diesen – in Abb. 7.2 veranschaulichten – Effekt zu überwinden, ist allenfalls bei gelungener Modellierung riesiger Wissensmengen vorstellbar – wenn überhaupt der Ansatz der symbolverarbeitenden Künstlichen Intelligenz hierfür die richtige Ebene darstellt.

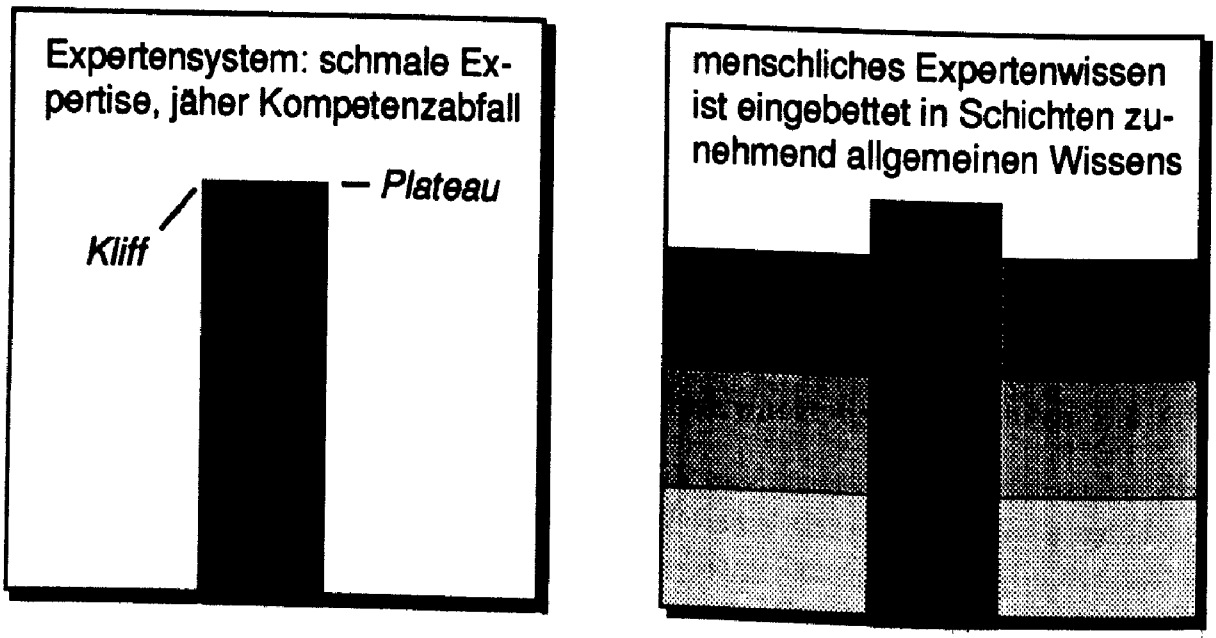

Abbildung 7.2: Veranschaulichung des Kliff-und-Plateau-Effektes bei Expertensystemen

Eine Haupterkenntnis aus den oben zitierten Expertenstudien ist allerdings folgende: Das

als „Expertenwissen" bezeichnete Spezialwissen ist (mindestens teilweise) aufgaben-, fall- bzw. zielorientiert organisiert, hat also eine bestimmte „Ausrichtung". Im Gegensatz dazu ist Allgemeinwissen, oder auch das Weltwissen für das Verstehen natürlicher Sprache, „unspezifisch relational". Ähnliches gilt allerdings auch für das Hintergrundwissen von Experten (z.B. von Ärzten über die allgemeinen Lebensumstände von Patienten). Soweit sich „ausgerichtetes", auf die Lösung bestimmter Problemklassen zugeschnittenes Expertenwissen identifizieren läßt, bestehen beim gegenwärtigen Stand der Kunst der Expertensystemtechnik Chancen, es für zweckgerichteten Einsatz zu modellieren.

7.1.2 Wissensrepräsentationen und Ableitungsstrategien

In diesem Abschnitt werden die in Expertensystemen häufig benutzten Grundtechniken der Wissensrepräsentationen mit ihren zugehörigen Ableitungsstrategien im Abriß dargestellt. Eine umfassende Darstellung enthält [Puppe, 1991], und eine allgemeine Diskussion von Wissensrepräsentationen findet sich in [Struß, 1991]. Die Abbildungen in diesem Abschnitt sind [Puppe, 1991] entnommen.

7.1.2.1 Regeln

Regeln sind die am weitesten verbreitete Wissensrepräsentation in Expertensystemen. Sie bestehen aus einer Vorbedingung und einer Implikation bzw. einer Aktion, z.B. „Wenn A, dann gilt B", oder „Wenn A, dann führe B aus". Im Gegensatz zur konventionellen Programmierung legt der Experte mit einer Regel nur fest, was in einem beschriebenen Zusammenhang getan wird, während die Reihenfolge der Regelausführung dem Regelinterpreter überlassen bleibt. Die beiden Haupttypen der Regelverknüpfung sind *Vorwärts-* und *Rückwärtsverkettung.* Bei der Vorwärtsverkettung geht der Regelinterpreter von der vorhandenen Datenbasis aus und wählt sich aus den anwendbaren Regeln mittels einer Konfliktlösungsstrategie eine Regel aus, deren Aktionsteil ausgeführt wird und die Datenbasis verändert. Dieser „Recognize-and-Act"-Zyklus wird solange wiederholt, bis ein Terminierungskriterium erfüllt oder keine Regel mehr anwendbar ist. Häufig benutzte Konfliktlösungsstrategien sind die Auswahl nach der *Reihenfolge* (Trivialstrategie), nach *Spezifität* (z.B. ist die Regel „A & B → C" spezifischer als die Regel „A → D"), nach *Aktualität* (eine Regel ist umso aktueller, je neuer die Aussagen ihrer Vorbedingung in der Datenbasis sind) oder aufgrund von *Zusatzwissen* (statische Regelprioritäten oder Meta-Regeln). Bei der Rückwärtsverkettung geht der Regelinterpreter von einem vorgegebenen Ziel aus und überprüft alle Regeln, deren Aktionen das Ziel enthalten. Wenn die Gültigkeit der Aussagen ihrer Vorbedingungen unbekannt sind, werden rekursiv Unterziele zur Bestimmung der Wahrheitswerte dieser Aussagen generiert, die entweder mit anderen Regeln abgeleitet oder vom Benutzer erfragt werden. Ein Beispiel für einen vorwärtsverkettenden Regelinterpreter ist OPS5 [Brownston et al., 1985] und für einen rückwärtsverkettenden EMYCIN [van Melle, 1981].

Die Größe von Expertensystemen wird häufig in Abhängigkeit von der Anzahl der Regeln angegeben. Man muß dabei jedoch auch die Ausdrucksstärke des Regelformalismus berücksichtigen, z.B. ob Variable instantiiert werden können.

Im einfachsten Fall besteht die Vorbedingung einer Regel aus nur einer Aussage. Bei komplexeren Regeln mit vielen Aussagen ist es vorteilhaft, die Aussagen nach ihrer Bedeutung zu unterscheiden, z.B. die Kernbedingung, den allgemeinen Kontext und spezielle Ausnahmen. Gleichfalls ist es oft zweckmäßig, die Regelaktionen zu typisieren, z.B. danach, ob sie zur Herleitung von Eigenschaften bestimmter Arten von Objekten oder zur Dialogsteuerung dienen. Ein Beispiel für solche Strukturierungsmöglichkeiten stellt der Regelformalismus von MED2 [Puppe, 1987] dar.

7.1.2.2 Objekte und Frames

Regeln beziehen sich auf Objekte in der Datenbasis. Während im einfachsten Fall die Datenbasis eine unstrukturierte Menge von Fakten beinhaltet, ist es häufig möglich, die Menge der Fakten zu strukturieren. Der erste Schritt dazu ist die Zusammenfassung aller Aussagen über ein Objekt in einer Datenstruktur, wie z.B. Records in PASCAL, Property-Listen in LISP oder Relationen einer Datenbank. Ein Beispiel wäre etwa ein Objekt „Motor" mit Aussagen über Zylinderanzahl, Verdichtungswert und Leistung. In Frames (auch Begriffe wie Schemata, Konzepte, Objekte sind gebräuchlich) wird die normalerweise passive Objektdarstellung durch Vererbungshierarchien, zugeordnete Prozeduren, Methoden, Erwartungswerte oder einen Mechanismus zur automatischen Klassifikation unbekannter Objekte aufgrund deren Eigenschaften erweitert.

In *Vererbungshierarchien* ererbt ein Frame allgemeine Eigenschaften von seinem Vorgänger in der Hierarchie, so daß nur die individuellen Eigenschaften bei dem Frame selbst abgespeichert werden müssen. Zur Flexibilitätssteigerung verwendet man auch Vererbungsheterarchien, bei denen ein Frame Eigenschaften von mehreren Vorgängern übernimmt. Bei größeren Expertensystemen kann die Übersicht häufig verbessert werden, indem die Vererbung bestimmter Eigenschaften gezielt unterdrückt wird oder sich der Name und Wertebereich einer Eigenschaft bei der Vererbung verändert (z.B. wird der Verdichtungswert des Motors mit numerischem Wertebereich zur Benzinsorte des Autos mit dem Wertebereich „Super" oder „Normal" transformiert). Vielfältige Vererbungsmechanismen bietet z.B. CRL, die Wissensrepräsentationssprache des hybriden Expertensystemwerkzeugs Knowledge Craft (siehe Teil 7.1.5).

Zugeordnete Prozeduren sind kleine Programme, die einer Eigenschaft eines Frames zugeordnet sind und bei einem Lese- oder Schreibzugriff auf dessen Wert ausgeführt werden, z.B. um den Wert zu berechnen oder Konsequenzen aus einer Wertänderung zu propagieren. Damit ließe sich z.B. die Rückwärts- und Vorwärtsverkettung eines Regelinterpreters nachbilden. Auch wenn ein Expertensystem Wertänderungen mittels „aktiver Werte" automatisch auf dem Bildschirm anzeigt, beruht das auf der Ausführung einer zugeordneten

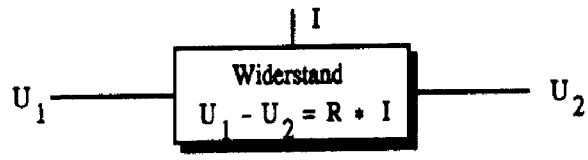

Abbildung 7.3: Beispiel eines Constraints für einen Widerstand mit Widerstandswert R

Prozedur.

Erwartungswerte (Defaultwerte) sind Vorbelegungen von Werten, die als Standardannahme meistens, aber nicht immer stimmen, z.B. „alle Vögel können fliegen". Erwartungswerte gelten nur dann, wenn kein genaueres Wissen verfügbar ist. Da sie gegebenfalls korrigiert werden müssen, wird bei Verwendung von Erwartungswerten ein System zum nicht-monotonen Schließen (s. Abschnitt 7.1.2.5) benötigt.

Wenn man ein gut strukturiertes Framesystem mit Vererbungsheterarchien hat, ist es ein naheliegender Gedanke, daß das System ein neues Objekt gemäß seiner Eigenschaften selbständig in die richtige Position in der Heterachie einfügt. Das erfordert eine strenge Definition, inwiefern ein Frame ein Unterframe eines anderen Frames ist, und natürlich das Verbot von Erwartungswerten, da sie willkürlich überschrieben werden können. Dieser Gedanke einer *automatischen Klassifikation* unbekannter Objekte ist in der *KL-ONE-Sprachfamilie* realisiert, deren Kernstück der „Classifier" zur Einordnung neuer Frames ist. Eine Übersicht dazu findet sich z.B. in [von Luck und Owsnicki-Klewe, 1991].

7.1.2.3 Constraints

Mit Constraints können beliebige Relationen zwischen Variablen repräsentiert werden. So beschränkt z.B. ein elektrischer Widerstand die Werte für Spannungs- und Stromvariablen U_1, U_2, I nach dem Ohmschen Gesetz (Abb. 7.3).

Im Unterschied zu Regeln können Constraints ungerichtet sein, d.h. wenn in dem Constraint in Abb. 7.3 zwei beliebige Variablen bekannt sind, kann die dritte Variable bestimmt werden. Viele Probleme lassen sich natürlich als ein Constraint-Netz beschreiben, d.h. als eine Menge von Constraints, die durch gemeinsame Variablen verbunden sind. Ein Constraint-Problem ist dann eine Anfangsbelegung einiger Variablen eines Constraint-Netzes, und seine Lösung besteht darin, möglichst eindeutige Werte für die übrigen Variablen zu finden. Ein Beispiel eines Constraint-Netzes für einen elektrischen Schaltkreis mit Widerständen, Verzweigungen, Dioden, Kondensatoren und Transistoren als Constraints und Spannungen und Strömen als Variablen findet sich in [Stallman und Sussman, 1977].

Ähnlich wie Regeln basieren Constraints auf einer allgemeinen Idee, für die es vielfältige Realisierungen gibt. So kann man Constraints als Tabellen, Funktionen oder als charakteristische Prädikate definieren und z.B. mit Datenbanken (für Tabellen), mit Regeln oder unmittelbar als Programmcode implementieren. Die Grundidee zur Lösung von Constraint-

$$P(D_i/S_1\&\ldots\&S_m) = \frac{P(D_i) * P(S_1/D_i) * \ldots * P(S_m/D_i)}{\sum_{j=1}^{n} P(D_j) * P(S_1/D_j) * \ldots * P(S_m/D_j)}$$

Abbildung 7.4: Die zur Bewertung von Alternativen hauptsächlich benutzte Form des Theorems von Bayes

Problemen besteht darin, daß, ausgehend von der Anfangsbelegung der Variablen, alle damit verbundenen Constraints aktiviert werden; bei deren Auswertung erhalten dann weitere Variablen einen Wert, was zur Aktivierung neuer Constraints führt, bis keine weitere Wertzuweisung an eine Variable mehr möglich ist. Dies Verfahren heißt *Propagierung.* Während im einfachsten Fall nur feste Werte für eine Variable propagiert werden können, sind leistungsfähigere Constraintsysteme auch in der Lage, Wertemengen, Intervalle oder symbolische Ausdrücke zu propagieren. Weiterhin kann die Funktionalität eines Constraint-Systems dadurch gesteigert werden, daß es plausible Annahmen für einen Variablenwert macht und diesen im Falle eines Widerspruches (der Zuweisung der leeren Menge an eine Variable) wieder zurückziehen kann. Falls ein Constraint-Problem unlösbar ist, kann das System versuchen, durch Zurücknahme von Constraints (d.h. durch Aufgeben der entsprechenden Anforderung) eine Lösung zu finden. Eine formale Beschreibung von Constraints findet sich z.B. in [Voß und Voß, 1989], und ein Beispiel eines Constraint-Interpreters, der feste Werte und Wertemengen propagieren kann, in [Güsgen, 1989].

7.1.2.4 Probabilistisches Schließen

In vielen Anwendungsbereichen sind die Fakten bzw. das Wissen inhärent unsicher. Zur Behandlung von *unsicherem Wissen* kann man die Wissensrepräsentation um Wahrscheinlichkeiten oder Evidenzwerte erweitern, die den Frame-Eigenschaft-Wert-Tripeln und den Regeln (seltener den Constraints) zugeordnet werden, z.B. „Wenn A, dann B mit 70% Wahrscheinlichkeit". Um Schlußfolgerungen herzuleiten, muß ein Verfahren zur Verknüpfung der Evidenzwerte angeben werden. Unter bestimmten Voraussetzungen sind dafür statistische Verfahren wie das *Theorem von Bayes* [Charniak und McDermott, 1985, Kap. 8.2] geeignet. Da in vielen Anwendungsbereichen von Expertensystemen diese Voraussetzungen jedoch nicht erfüllt sind, werden meist andere Techniken verwendet, die sich in der Praxis mehr oder weniger gut bewährt haben. Der Leitgedanke ist, ein möglichst einfaches Verrechnungsschema zu benutzen und keine größere Genauigkeit vorzutäuschen als aufgrund der Ausgangsdaten erreichbar ist. Der Basisalgorithmus zur Bewertung von Alternativen aufgrund unsicherer Informationen besteht aus drei Schritten:

1. starte mit der Apriori-Wahrscheinlichkeit der Alternativen
2. modifiziere bei jeder neuen Information die Wahrscheinlichkeit der Alternativen entsprechend den Evidenzwerten und dem Verrechnungsschema
3. wähle die am besten bewertete Alternative aus

Neben dem Theorem von Bayes sind zwei verbreitete Verrechnungschemata in Expertensystemen das MYCIN-Schema [Shortliffe und Buchanan, 1975] und das INTERNIST-Schema [Miller *et al.*, 1982].

Das Theorem von Bayes eignet sich dazu, aus den Apriori-Wahrscheinlichkeiten $P(D_i)$ einer Menge von n Diagnosen und aus den bedingten Wahrscheinlichkeiten $P(S_j/D_i)$, d.h. der statistischen Häufigkeit des Auftretens der Symptome bei gegebenen Diagnosen, die bedingte Wahrscheinlichkeit $P(D_i/S_1\&\ldots\&S_m)$ der Diagnosen bei gegebenen Symptomen gemäß der Formel in Abb. 7.4 zu berechnen.

Die Voraussetzungen zur Anwendung des Theorems von Bayes sind die Unabhängigkeit der Symptome untereinander, die Vollständigkeit und der wechselseitige Ausschluß der Diagnosen sowie gute und große Statistiken.

Das MYCIN- und INTERNIST-Schema unterscheiden sich vom Theorem von Bayes vor allem durch die getrennte Bewertung von positiver und negativer *Evidenz* (also von Fakten, die für bzw. gegen eine Diagnose sprechen), durch gröbere Bewertungskategorien, durch die Einteilung der Diagnosen in Gruppen konkurrierender Diagnosen (INTERNIST) und durch die Berücksichtigung von Symptomkombinationen (MYCIN).

7.1.2.5 Nicht-monotones Schließen

Im Alltag werden häufig Schlüsse gezogen, weil sie aufgrund zunächst vorhandener Informationen plausibel scheinen (z.B. „der Kilometerzähler zeigt 2.000 km, das Auto ist also kaum gefahren"). Sie können durch zusätzliche Information („der Zähler wurde einmal ausgetauscht") hinfällig werden, sind aber für das Überleben unumgänglich, denn über vollständige Information verfügen wir nahezu nie. Weil bei dieser Art zu schließen aus mehr Information weniger geschlossen werden kann, nennt man sie *nicht-monoton* im Gegensatz zum monotonen klassischen logischen Schließen, wo aus mehr Information stets auch mindestens gleich viel geschlossen werden kann. (Dort gilt für alle Aussagemengen $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{P}$, daß wenn $\mathcal{P}$ aus $\mathcal{S}_1$ herleitbar ist, dann auch aus der Vereinigung von $\mathcal{S}_1$ und $\mathcal{S}_2$). Während theoretische Aspekte des nicht-monotonen Schließens u.a. in [Brewka, 1989] und [Genesereth und Nilsson, 1987, Kap. 6] behandelt werden, konzentrieren wir uns hier auf Mechanismen, wie nicht-monotones Schließen in Expertensystemen behandelt werden kann.

Beim nicht-monotonen Schließen müssen Schlüsse, die sich als ungültig herausstellen, mit allen Konsequenzen revidiert werden. Das einfachste Verfahren wäre die vollständige Neuberechnung aller Inferenzen aufgrund der neuen Menge an Ausgangsdaten. Dieser Weg

wird auch in einigen Expertensystemen gewählt. Jedoch wächst der Aufwand mit der Komplexität des Falles, deshalb ist dieses „brute-force" Verfahren nur in kleinen Expertensystemen praktikabel. Die Kernidee zu einer Effizienzverbesserung ist das Abspeichern von Begründungen für jede Schlußfolgerung. Dann kann die Frage, ob eine Schlußfolgerung bei Änderungen gültig bleibt, immer dann bejaht werden, wenn sie nach der Revision noch mindestens eine gültige Begründung besitzt. Die beiden wichtigsten Rücksetztechniken, die auch TMS (für it Truth-Maintenance-System) oder RMS (für *Reason-Maintenance-System*) genannt werden, unterscheiden sich darin, was als Grundlage einer Schlußfolgerung abgespeichert wird:

- direkte Begründungen: JTMS (Justification-based TMS, z.B. das System von Doyle [Doyle, 1979])
- Basisannahmen, die einer Begründung zugrunde liegen: ATMS (Assumption-based TMS, z.B. das System von de Kleer [de Kleer, 1986]).

Der Unterschied läßt sich an einem einfachen Beispiel mit zwei Regeln verdeutlichen: Regel1: $A \rightarrow B$, Regel2: $B \rightarrow C$. Ist A eine Basisannahme, so wird im JTMS für C die Begründung Regel2 abgespeichert, während im ATMS ein Kontext (d.i. eine Menge von Basisannahmen) generiert wird, der A enthält.

Der JTMS-Basis-Algorithmus ist einfach und effizient: eine Änderung eines Faktums bewirkt, daß alle mit diesem Faktum assozierten Begründungen überprüft werden. Falls eine Begründung ungültig ist, wird getestet, ob die zugehörige Schlußfolgerung noch weitere Begründungen hat. Falls nein, wird die Schlußfolgerung zurückgezogen und mit ihr als Input dieser Algorithmus rekursiv aufgerufen.

Das Hauptproblem dabei ist die Behandlung von Ableitungsschleifen; z.B. würde eine Schleife $A \rightarrow B$ und $B \rightarrow A$ im Basis-Algorithmus bewirken, daß A und B nicht mehr rücksetzbar sind. Eine Lösung dieses Problems ist das Abspeichern von nicht-zirkulären (well-founded) Begründungen für eine Schlußfolgerung.

Beim ATMS existiert für jede Schlußfolgerung ein sogenanntes Label, das aus Mengen von Basisannahmen besteht, unter denen die Schlußfolgerung gültig ist. Eine Schlußfolgerung ist gültig, wenn ihr Label mindestens eine Menge enthält, die eine Teilmenge der global gültigen Basisannahmen ist. Ein Vorteil des ATMS ist, daß die Konsequenzen verschiedener Mengen von Basisannahmen leicht miteinander verglichen werden können. Jedoch wächst bei großen Wissensbasen mit sehr vielen Basisannahmen die Anzahl der notwendigen (und für die Effizienz des ATMS kritischen) Mengenvergleiche drastisch an. Auch die Behandlung von Ausnahmen stellt dort ein Problem dar.

7.1.2.6 Temporales Schließen

Die Voraussetzung zum temporalen Schließen ist die Erweiterung der Wissensrepräsentation von Fakten um Zeitangaben, z.B. „Startzeitpunkt der Brustschmerzen: vor drei Wochen". Wichtige Aspekte der Zeitrepräsentation sind:

- punkt- oder intervallbasierte Basisrepräsentation
- exakte, qualitative oder quantitativ ungenaue Zeitangaben
- Bezug auf eine absolute Zeitskala oder auf Referenzereignisse
- Zeitangaben als Zahlen oder als Zahlen mit Zeiteinheiten

Die punkt- und intervallbasierte Repräsentation sind im Prinzip gleich mächtig, da man ein Intervall durch Anfangs- und Endpunkt bzw. einen Zeitpunkt durch ein beliebig kleines Intervall darstellen kann. Trotzdem kann die jeweilige Einfachheit und Eleganz der Handhabung für verschiedene Anwendungsbereiche sehr unterschiedlich sein.

Am einfachsten ist die Repräsentation exakter Zeitangaben. Wenn keine genauen Angaben verfügbar sind, können Ungenauigkeiten qualitativ (A begann vor B) oder quantitativ ungenau (A begann 3–4 Wochen vor B) angegeben werden.

Bei exaktem Wissen kann man gewöhnlich eine absolute Zeitskala (z.B. Datum und Uhrzeit) verwenden. Die Alternative ist die Verwendung von geeigneten Referenzereignissen. Das ist nur dann äquivalent zu einer absoluten Zeitskala, wenn die Beziehungen zwischen allen Referenzereignissen bekannt sind. Bei ungenauen Zeitangaben kann man die Genauigkeit häufig durch Bezug auf mehrere Referenzereignisse erhöhen, z.B. A begann höchstens 3 Stunden vor B, 2–4 Stunden nach C, mindestens 3 Stunden vor D, usw. Für das Ausrechnen der genauesten Zeitrelation von A zu anderen Zeitpunkten oder -intervallen kann man Techniken der Constraint-Propagierung (s. Kap. 2.3) verwenden.

Die Verwendung von Zeiteinheiten kann neben der besseren Lesbarkeit auch dazu dienen, Ungenauigkeiten auszudrücken: so bedeuten „vor einem Jahr“ und „vor 365 Tagen“ nicht unbedingt dasselbe (siehe [Mittal *et al.*, 1984]).

Die Komplexität der Zeitrepräsentation bestimmt, wie aufwendig und effizient zeitbezogene Fragen beantwortet werden können. Typische Fragen an Zeitdatenbanken sind:

- Ist ein Faktum während eines bestimmten Zeit-Intervalls gültig?
- Hat sich ein Wert oder der Anstieg eines Wertes während der letzten Zeit verändert?
- Wie ist die zeitliche Relation zwischen zwei Fakten?

Das erste Expertensystem, das Zeitangaben auswerten konnte, war das Ende der siebziger Jahre fertiggestellte System VM zur Überwachung von Patienten an der Eisernen Lunge [Fagan *et al.*, 1984], das mit einer punktbasierten Basisrepräsentation, exakten Zeitangaben, einer absoluten Zeitskala und ohne Zeiteinheiten operiert. Komplexere Formalismen zum temporalen Schließen sind der Zeitkalkül von Allen [Allen, 1983] und der TMM (Time Map Manager) in [Dean und McDermott, 1987]. Eine ausführlichere Darstellung temporalen Schließens findet sich im Kapitel „Räumliches und zeitliches Wissen“ dieses Buches, sowie in [Materne und Hertzberg, 1991].

7.1.3 Wissensmodellierung

7.1.3.1 Methodologien zur Wissensmodellierung

Expertensysteme werden heute in aller Regel als wissensbasierte Systeme konzipiert. Die Kernidee solcher Systeme läßt sich – ausgehend vom Problemlöseparadigma der Künstlichen Intelligenz – kurz wie folgt erläutern: Grundsätzlich basiert Problemlösung auf Suche, jedoch sind sog. schwache Suchmethoden für komplexe Suchräume nicht adäquat. Menschen benutzen *heuristisches* Wissen, um die Suche einzuschränken, d.i. aus Wissen über den Bereich abgeleitetes Kontrollwissen für die Suche. Ein wissensbasiertes System ist dann als ein Problemlösesystem aufzufassen, in welchem Erfahrung und Expertise eines menschlichen Problemlösers eingebettet ist, um die Komplexität des Suchraums zu reduzieren. Die Aufgabe des Knowledge Engineering besteht darin, derartiges Wissen für einen eingegrenzten Aufgabenbereich von menschlichen Experten zu erheben und in einem geeigneten Repräsentationsformalismus so zu organisieren, daß sich Probleme dieses Bereichs mit einem maschinellen System lösen lassen. Die Grundforderungen, die mit einem wissensbasierten Problemlösesystem erfüllt werden sollen, sind Effizienz, Überschaubarkeit und Nachvollziehbarkeit der Problemlösung.

Methodologien für die Modellierung von Wissen sind daher zu integrieren in Methodologien für die Entwicklung von wissensbasierten Systemen. Diese sollen ein Repertoire an Methoden bereitstellen, mit denen die komplexe Aufgabe der Erstellung eines wissensbasierten Systems strukturiert angegangen werden kann. Die schwierigste Aufgabe ist die Überbrückung der Kluft zwischen der Beobachtung menschlicher Expertise und lauffähigen Computersystemen: Teile des fachlichen und des schwer verbalisierbaren Erfahrungswissens von Experten sollen operationalisiert werden, wobei die Konflikte zwischen den angewandten „weichen" Erhebungsmethoden und den anzustrebenden Formalisierungen ein Hauptproblem darstellen. Techniken der Erhebung von Expertenwissen, die „harten" Ansprüchen der Testtheorie genügen, sind schwerlich denkbar. *Es geht um Approximationen schlecht strukturierter, „diffuser" Problembereiche*, d.h. solcher Problembereiche, die sich mit den Verfahren der algorithmischen Datenverarbeitung nicht gut bewältigen lassen.

Um einen diffusen Bereich zu approximieren, soll das an Menschen beobachtete, oft lokalistische heuristische Problemlösungswissen in geeigneter Form nachgebildet werden. Dabei ist es nicht so, daß man Stück für Stück („Regel für Regel") an das heuristische Wissen gelangen und es sofort in einen Repräsentationsformalismus übernehmen kann, wie es in frühen Phasen der Expertensystemtechnik nahegelegt wurde: Im *Stanford Heuristic Programming Project* wurde mit der Annahme gearbeitet, daß sämtliches Bereichswissen in Form von Bedingungs-Handlungs-Paaren – Regeln – ausgedrückt werden kann, die voneinander unabhängige („modulare") Aussagen über einen Bereich darstellen und mit einem Standard-Interpreter ausgeführt werden können. Diese als *Unabhängigkeitsthese* bekanntgewordene Sichtweise („Die Form der Darstellung von Bereichswissen ist unabhängig von seinem Gebrauch") findet die folgende Kritik: Regelbasierte Systeme sind zwar einfach zu

konstruieren, im Großen aber schwer vorherzusagen und zu warten. In vielen Fällen ergaben sich Vermischungen von Bereichswissen und Kontrollstrategie durch den Zwang, die Ausführung sämtlicher Aktionen des Systems über den Regelinterpreter abzuwickeln, so daß dieser Zugang oft ein falsch verstandenes Knowledge Engineering, nämlich „Hinprogrammieren" erforderte.

Kritiker der Unabhängigkeitsthese waren u.a. *Bylander* und *Chandrasekaran*, die im Gegenzug eine *Interaktionsthese* formulierten [1987]: Die Repräsentation von Wissen zum Zweck, bestimmte Probleme damit zu lösen, wird stark von der Natur des Problems und der auf das Wissen angewandten Inferenzstrategie beeinflußt. Das heißt, Bereichswissen kann nicht unabhängig von Annahmen über seinen Gebrauch repräsentiert werden; das Modell des jeweils betrachteten Problemlöseprozesses diktiert die Form der Repräsentation von Bereichswissen.

Methoden zur Wissensmodellierung müssen sich grob zusammengefaßt auf die folgenden Einzelaspekte richten:

- Techniken, um Daten - wie Videoaufzeichnungen und Protokolle - von Experten zu erheben
- Analyse und Interpretation dieser Daten, um herauszuschälen, worin das zugrundeliegende Wissen und der zugrundeliegende Schlußfolgerungsprozeß der Expertenleistung besteht, was für Wissen dabei eingesetzt wird und in welcher Weise
- Benutzung der interpretierten Daten, um formale Beschreibungen der Expertenstrategien und des eingesetzten Domänenwissens zu gewinnen
- Entwurf und schließlich Implementierung eines operationalen Wissensmodells

Für das weitere ist es hilfreich, Ebenen der Wissensbeschreibung zu unterscheiden, die *Newell* und *Brachman* etwa Anfang der 80er Jahre vorgeschlagen haben; sie lassen sich in den vorgestellten Ansätzen zur Wissensmodellierung wiedererkennen. Newells Vorschlag [1982] unterscheidet die *Wissensebene* (Was für Wissen braucht man, um eine Aufgabe zu bearbeiten? Wie wird dies Wissen dazu eingesetzt?) von der *Symbolebene* (Wie ist Wissen symbolisch repräsentiert? Welche Prozeduren darauf erbringen die erwünschte Leistung?). Nach Brachmans Vorschlag [1979] sollten die folgenden vier Ebenen sequentiell betrachtet werden: die *linguistische Ebene*, die keine Abstraktion über Sprache hinaus vornimmt, die *konzeptuelle Ebene*, auf der primitive Begriffe und Relationen abstrahiert werden, die *epistemologische Ebene*, auf der Typen von Wissenselementen zur Aufgabenbearbeitung abstrahiert werden, und die *Implementierungsebene*, die Daten- und Kontrollstrukturen einer Computersprache beinhaltet.

7.1.3.2 Begriffsabgrenzungen

Wissensmodellierung als Teilaufgabe der Wissensrepräsentation (vgl. Teil 7.1.1.2) betrifft die inhaltliche Frage der Auswahl von Kategorien zur Darstellung des betrachteten Welt-

bereichs, Festlegung ihrer Zwischenbeziehungen und darauf bezogener Schlußmöglichkeiten etc. (vgl. Teil 7.1.1). Das heißt, es handelt sich in erster Linie um eine Design-Tätigkeit, die jedoch nicht ohne vorangehende Analyse der zu modellierenden Wissensdomäne vorgenommen werden kann. Diese Analyse ist neben der Erhebung des maßgeblichen Wissens eine Hauptaufgabe der *Wissensakquisition.*

Bevor wir uns näher mit Ansätzen methodischer Bewältigung des Problems der Wissensmodellierung befassen, sollen zunächst – in teilweiser Anlehnung an [Karbach und Linster, 1990] – die verwendeten Begriffe abgegegrenzt werden. Solche und ähnliche Begriffe finden sich in Variationen in den diversen bekannten Ansätzen der Wissensmodellierung für Expertensysteme. Sie sollen hier nicht als autoritative Definitionen verstanden werden, sondern vor allem unterscheidbare Einzelaspekte in den Blick rücken.

Der Begriff „Knowledge Engineering" ist von Edward Feigenbaum [1977] geprägt worden. Er wird bezogen auf angewandte KI; nicht die Erklärung von Expertenleistung, sondern der Transfer von Expertise auf maschinelle Systeme zwecks Nutzung steht im Vordergrund. Unter *Knowledge Engineering* wird häufig der gesamte Erstellungs- und Wartungsprozeß eines wissensbasierten Systems verstanden (Machbarkeitsstudien, Auswahl von Werkzeugsystemen, Wissensakquisition bis zur Integration eines fertigen Systems in die Einsatzumgebung, Wartung und ggf. Erweiterung; siehe dazu auch Teil 7.1.4). Hier soll aber vor allem der engere Bereich der Erstellung von Wissensmodellen für einen Anwendungsbereich betrachtet werden (vgl. Abb. 7.5).

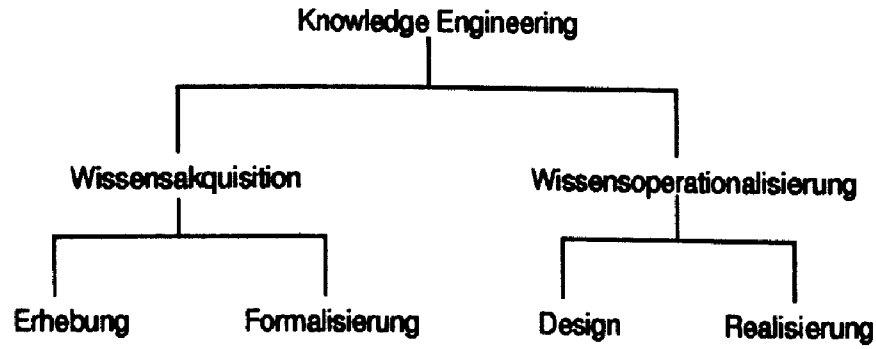

Abbildung 7.5: Die engeren Aufgaben des Knowledge Engineering

Unter *Wissensakquisition (Wissenserwerb)* verstehen einige Autoren den Prozeß von der Erhebung von Wissen aus verschiedenen Wissensquellen bis hin zur Umsetzung in eine operationale Wissensbasis und deren inhaltlicher Wartung (die erneute Wissensakquisition erfordern kann). Entsprechend neuerer Betrachtungsweisen, auf die noch eingegangen wird, kann es hilfreich sein, einen Hauptaspekt der Wissensakquisition bei der Erhebung und (maschinenunabhängigen) Formalisierung von Expertenwissen zu sehen.

Bei der *Wissenserhebung (knowledge elicitation)* geht es darum, Einzelheiten über die besonderen Expertenfähigkeiten „ans Licht zu bringen" und in sog. Wissensprotokollen zu dokumentieren (zunächst als reine Datensammlung). *Wissensanalyse* und *Wissensinter-*

pretation dienen der Umsetzung solcher Wissensprotokolle in formale Repräsentationen: allerdings nicht als neutraler Transfer, sondern vermittelt durch den Knowledge Engineer, der sein Bild von der beobachteten Expertise in ein *mentales konzeptuelles Modell* entwickelt, welches sowohl den Problemlöseprozeß als auch Strukturen des Bereichswissens betrifft. Hier erfolgen die entscheidenden Schritte von der „weichen" zur „harten" Wissensbeschreibung, also der *Formalisierung.*

Analyse und Modellierung von Expertise wirken stark aufeinander ein: die Analyse gibt das zu Modellierende vor; das Modell bestimmt seinerseits die Kategorien der Analyse. Es muß gestatten, Wissen in „modulare" Bestandteile zu zerlegen, die im Problemlösungsprozeß verwendet werden und die nach und nach inkrementell erhoben werden können. Hierfür gibt es stark unterschiedliche Ansätze, auf die später näher eingegangen wird; z.B. verfolgen Rapid-Prototyping-Ansätze die Idee, so früh wie möglich eine Minimalversion des in Angriff genommenen Systems zu bauen, das als Referenz für die weitere Entwicklung dient. Dabei beschränkt man sich zunächst auf das zur Behandlung weniger ausgewählter Fälle von Aufgaben notwendige Wissen. Das Wissensmodell *ergibt* sich hier am Ende der Entwicklung. In *modellbasierten Ansätzen* wird dagegen vor irgendwelchen Schritten zur Implementierung versucht, Modelle des Problemlöseprozesses zu explizieren; zunächst geht es um die Erstellung eines Modells (in vielen Fällen als Papierdokument) in einer dem Experten verständlichen Weise – „auf der Wissensebene".

Beim Übergang zu einem (maschinenhandhabbaren Wissensmodell kommt es darauf an, das formalisierte Expertenwissen zweckbezogen umzusetzen. Zum konzeptuellen Modell, dessen Zweck zunächst eine formale Beschreibung der beobachteten Expertise ist, treten Anforderungen an ein zu erstellendes wissensbasiertes System hinzu: beides ist Ausgangspunkt für das Design des Systems. Die Aufgabe ist hier, ein ausführbares Weltmodell zu schaffen, in dem die identifizierten Wissensbestandteile effektiv von der Problemlösungskomponente des geplanten Expertensystems benutzt werden können. Erst bei der *Wissensoperationalisierung* geht es um die Realisierung der so analysierten und interpretierten Daten in computerlauffähige Repräsentationsformalismen.

7.1.3.3 Erhebung von Expertenwissen

Bei der *Wissenserhebung* soll die Expertise, die Fähigkeiten eines Experten ausmacht, „herauspräpariert" und dokumentiert werden. Eine besondere Rolle spielen dabei die Begriffe, die Experten in der Wahrnehmung ihrer Fachtätigkeit benutzen. Wie bereits erwähnt sind dies keineswegs nur Fachbegriffe, die sich in der jeweils einschlägigen Literatur finden, sondern gerade der von Experten selbst geprägte und verwendete Begriffsjargon stellt möglicherweise einen bedeutenden Faktor ihrer Expertise dar.

Konstruktgitterverfahren. Unter mehreren Verfahren, die sich auf die Abklärung der Begriffsstrukturen von Experten beziehen, soll eine psychologische, standardisierte Erhebungsmethode herausgegriffen werden, die auf Kellys Psychologie der *persönlichen Kon-*

strukte [Kelly, 1955] zurückgeht. Es geht dort ursprünglich darum, Beschreibungsbegriffe für die Welt- und Eigenwahrnehmung von Versuchspersonen oder Patienten zu gewinnen, die nicht von einer Theorie vorgegeben, sondern von Versuchspersonen selbst geliefert werden: sog. persönliche Konstrukte (personal constructs). Nach Kelly gestatten sie, die Welt zu strukturieren und sich daraus ergebende Verhaltenskonsequenzen zu finden, sei es *explizit* formuliert (verbalisierbar) oder *implizit* in Handlung umgesetzt (nichtsprachlich). Die Konstrukte einer Person können sich im Laufe der Zeit durch Benutzung verändern und haben eingeschränkte Brauchbarkeitsbereiche. Sie sind systematisch aufeinander bezogen, können jedoch teilweise voneinander unabhängige Subsysteme bilden, die möglicherweise miteinander inkonsistent sind. Auf diesem Ansatz baut der Role-Construct-Repertory-Test (auch Repertory-Grid-Test) auf, der unter der Bezeichnung *Konstruktgitterverfahren* als eines von mehreren standardisierten Verfahren zur Erhebung von Expertenbegriffen bekannt geworden ist.

Der Ablauf des Konstruktgitterverfahrens sei nur in der Grundform kurz vorgestellt: Zu Beginn nennt die Versuchsperson (hier: Experte oder Expertin) wichtige Elemente der betrachteten Domäne. Sie können konkret oder abstrakt, müssen aber vom gleichen Typ und von gleicher Granularität sein. Drei verwandte Begriffe werden sodann vorgegeben, z.B. *Leitrad, Pumpenrad, Turbinenrad.* Die Versuchsperson soll ein Merkmal angeben, das nur zwei der Begriffe teilen. Mit wechselnden Ausgangsbegriffen wird dieses Verfahren häufig wiederholt. Bei diesem Erhebungsvorgang wird das sog. Konstruktgitter aufgebaut, das die verschiedenen Domänenbegriffe mit den festgestellten charakterisierenden Merkmalen in Beziehung setzt. Die Auswertung erfolgt qualitativ: Es wird festgestellt, welche Begriffe überhaupt benutzt werden und wie sie zueinander in Beziehung stehen. Die Rekonstruktion von Begriffs*strukturen* (z.B. taxonomische Beziehungen) erfolgt z.Tl. automatisiert; es finden auch faktorenanalytische Verfahren zur Extraktion abstrakterer Begriffsstrukturen Einsatz.

Der Grundgedanke des Konstruktgitterverfahrens ist, daß der Experte seine zur Strukturierung der Domäne verwendeten Begriffe selbst liefert: als Menge von Unterscheidungen an relevanten Elementen, die nicht von einer Theorie vorgegeben, sondern eben persönliche Konstrukte sind. Hiermit wird vor allem ein *neutrales* Erhebungsverfahren angestrebt. Als Kritik (die eine grundsätzliche Kritik an phänomenologischen Methoden widerspiegelt) wird jedoch geäußert, daß die Erhebung der Expertenbegriffe verfälscht wird durch interferierende Selbstwahrnehmung, außerdem monoton und ermüdend ist und viel Ausschuß produziert; nur „flache" Relationen können erfaßt werden. Trotz dieser Vorbehalte ist das Konstruktgitterverfahren als Grundlage einiger Spezialwerkzeuge zur Unterstützung des Wissenserwerbs eingesetzt worden, auf die später kurz eingegangen wird.

Interviewtechniken. Um näheren Aufschluß über das Vorgehen von Experten bei der Bearbeitung von Aufgaben und der Lösung von Problemen zu erhalten, wird man nicht umhinkommen, sie eingehend zu beobachten und zu befragen. Neben dem Repertory-Grid-Test und ähnlichen Verfahren finden weitere individuenzentrierte Verfahren zur Erhebung

verbaler Daten Einsatz ([Huber und Mandl, 1982]), die sich u.a. darin unterscheiden, wie strukturiert die Anforderungen an eine Versuchsperson sind, über ihre Überlegungen Auskunft zu geben (z.B. ist der Repertory-Grid-Test oder ein Fragebogen stärker strukturiert als ein freies Interview oder die Methode des „lauten Denkens", bei der die Versuchspersonen möglichst alle Überlegungen während einer Problembearbeitung verbalisieren sollen).

Das vorrangige Erhebungsverfahren des Knowledge Engineering ist das Experteninterview. Beim Interview handelt es sich grundsätzlich um einen diagnostischen Dialog, in dem der Interviewer die Beweggründe für das Handeln, die Hypothesen, Vorhersagen, Allgemeinheit der Begriffe und die Art der Denkoperationen einer Versuchsperson einzugrenzen versucht. Die gewonnenen Daten (mit Tonband oder auch Videogerät aufgezeichnet) sind in erster Linie verbaler Natur; sie werden vom Wissensingenieur qualitativ ausgewertet. Verschiedene Techniken können unterschieden werden: In einem *unstrukturierten Interview* befragt der Wissensingenieur den Experten mehr oder weniger spontan über sein Wissen, auch während der Lösung von Testfällen. In einigen Fällen findet auch die *Introspektion* Anwendung: Der Experte beschreibt, wie er nach seiner Ansicht Fälle seines Zuständigkeitsbereichs löst und welche Strategien er benutzt. Beim *Laut-Denken-Protokoll* verbalisiert der Experte sein Denken, während er ein Problem löst. *Strukturierte Interviews* sind vorgeplante Erhebungen, die z.B. der nachträglichen Ergänzung anderweitig gewonnener Protokolle dienen. Das genaue Vorgehen ist vorab in einem Interview-Skript festgelegt; jedoch ist Flexibilität des Interviewers dort erforderlich, wo unverständliche oder mehrdeutige Antworten aufgeklärt werden müssen.

Die Zielsetzung von Interviews kann sowohl die *Problemcharakterisierung* (Abklärung der Natur des Problemlöseprozesses) als auch der *Erwerb von Bereichswissen* sein. In etlichen Fällen wird es zu Mischformen verschiedener Interviewtechniken kommen. Für die Expertenbefragung eignet sich etwa das aufgabenbasierte (task-based) Interview, eine freie Form des Interviews, das mit einer Problemstellung (einem Testfall) beginnt und in dessen Verlauf der Einsatz von Wissen bei der Problemlösung beobachtet wird. Es besteht die Vereinbarung, daß die Versuchsperson nach Möglichkeit ihr in den Sinn kommende Dinge laut ausspricht und Erklärungen über ihr Handeln abgibt. Der Interviewer kann Rückfragen stellen oder Bemerkungen machen oder sich ganz zurückhalten. Als Testfälle werden behandelt: typische Fälle (Routine), Fälle mit begrenzter Information oder beschränkter Zeit, schwierige Fälle.

Das Frageverhalten des Interviewers wird hinsichtlich der Verwertbarkeit der erhaltenen Information für kritisch gehalten. Erfahrungsgemäß ist es günstig,

- nach einer gestellten Frage zureichend lange ohne weitere Intervention zu warten, daß die Versuchsperson zu antworten beginnt (die Bereitschaft zu antworten erhöht sich);
- der Versuchsperson zu signalisieren, daß ihre Antworten in jedem Fall begehrt und anerkannt sind (schon aufmerksames Zuhören kann dies leisten);

- Begründungen für eine Antwort möglichst umgehend zu erfragen (solange das zur Antwort führende Gedachte noch im Denken der Versuchsperson gegenwärtig ist – später erhöht sich die Wahrscheinlichkeit erfundener Begründungen).

Natürlich wird es trotz aller Sorgfalt nicht gelingen, aus den verbalen Daten ein „isomorphes" Abbild der Denkprozesse von Experten zu erhalten. Kritische Punkte sind etwa, inwieweit die erlangten Daten vollständig sind und ob durch die Interaktionen mit dem Interviewer und durch die geforderten Verbalisierungen nicht Denkprozesse oder gar das Wissen selbst Veränderungen unterworfen sind. Jedoch werden Interviewverfahren aufgrund der Unmöglichkeit, kognitive Prozesse unmittelbar zu beobachten, als derzeit beste Möglichkeit akzeptiert, Aufschlüsse über Denkvorgänge zu erlangen.

Schwierigkeiten bei der Wissenserhebung können auch teils motivationaler Natur sein, teils haben sie mit der bereits erwähnten grundsätzlichen Kluft zwischen beobachtetem Können und den zur Verfügung stehenden Beschreibungsmöglichkeiten zu tun. Experten sind möglicherweise nicht an der Explizierung ihrer Modelle interessiert. Sie artikulieren sich eher in erlebten Situationen und Fallbeispielen als in Abstraktionen, die etwa Regelbeschreibungen darstellen. Häufig werden alte Erfahrungen an eine neue Situation unter Berücksichtigung der Unterschiede angepaßt, um die Situation zu meistern. Als besonders schwierige Punkte erweisen sich die Kontextbezogenheit, Verallgemeinerbarkeit und Adaptierbarkeit des Expertenwissens: Fast zu jeder identifizierten Regel gibt es Ausnahmen. Erklärungen und Begründungen sind auch bei sensibler Interviewführung schwer zu erhalten, sie können konstruiert wirken und es sogar sein. Es kann hilfreich sein, mit mehreren Experten zu arbeiten, was als Vorteil hat, daß sich umfassenderes Wissen erlangen läßt und überdies weniger leicht ein idiosynkratisch gestaltetes System entsteht (bessere Akzeptanz bei Benutzern); als Nachteil muß dabei jedoch größerer Zeitaufwand in Kauf genommen werden, und Konflikte aufgrund schwer vereinbarer Sichten der beteiligten Experten sind möglich.

7.1.3.4 Formalisierung von Expertenwissen am Beispiel KADS

KADS [Wielinga und Breuker, 1986] steht für „Knowledge Analysis and Documentation System" und bezeichnet heute eine umfassende Methodologie zum Entwurf wissensbasierter Systeme mit entwickelten Vorgehensweisen zur Wissenserhebung sowie der Interpretation und Dokumentation des erhobenen Wissens. Sie ist von 1985 bis 1990 im Rahmen eines ESPRIT-Projekts erarbeitet worden und wird in einem weiteren ESPRIT-Projekt *KADS-II* fortentwickelt. Wir beschränken uns hier darauf, die grundsätzlichen Ideen von KADS zu vermitteln. Entscheidend ist die Betonung der *Wissensebene*: Mit der Erstellung eines konzeptuellen Modells unabhängig von Implementierungserwägungen steht KADS im bewußten Gegensatz zum Rapid Prototyping; es handelt sich um eine zunächst vollständig implementationsunabhängige Modellierungsmethodik für Expertenwissen, die deutliche formale Anteile aufweist.

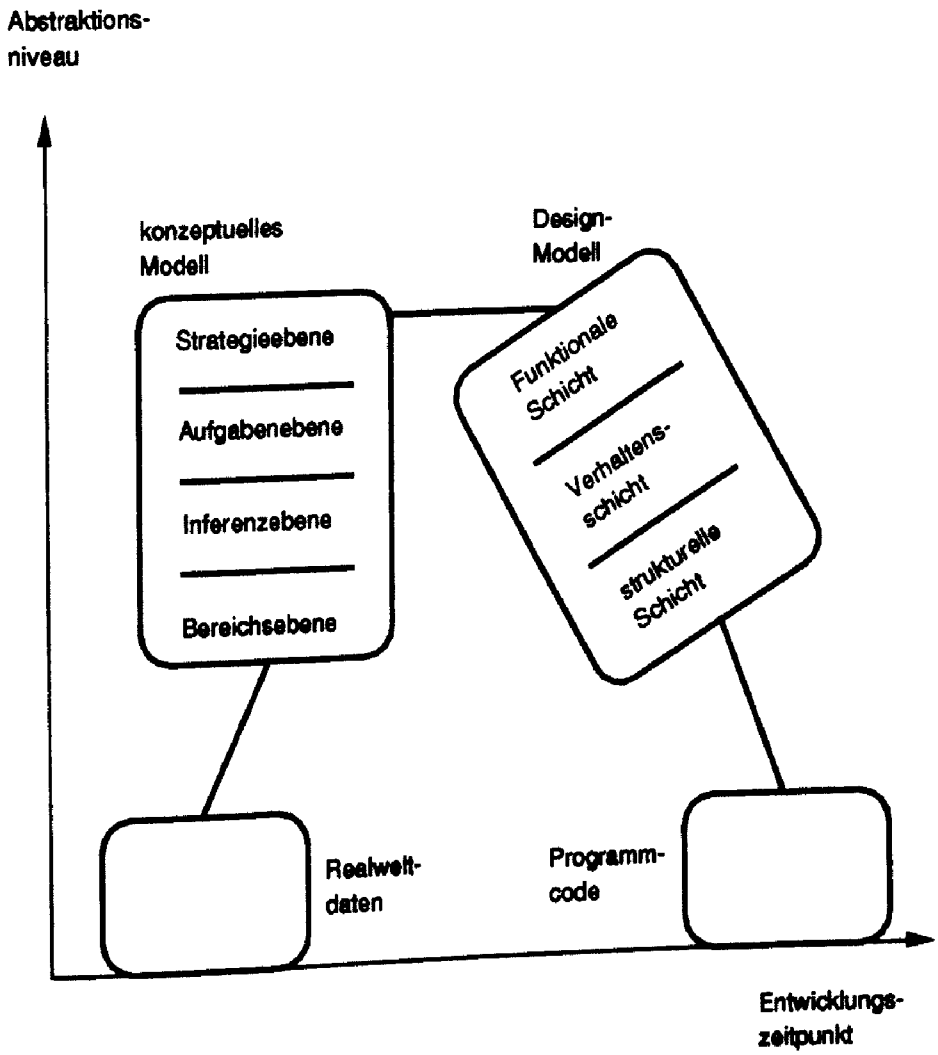

Abbildung 7.6: Strukturierung des KADS-Prozesses: In KADS-I werden die vier Ebenen eines konzeptuellen Modells gleichzeitig bzw. zyklisch erarbeitet, wogegen die Erarbeitung der Schichten des Design-Modells sequentiell vorgesehen ist.

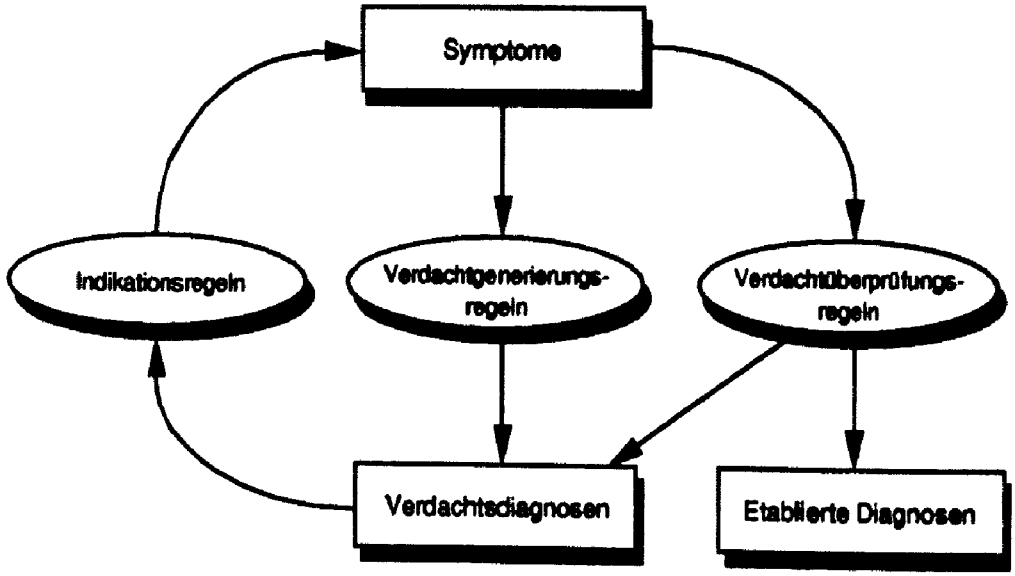

Abbildung 7.7: Inference-level-Skizze der Hypothesize-and-Test-Strategie (nach [Puppe, 1991]); weitere Erläuterung im Text.

Im konzeptuellen Modell wird das Vorgehen des Experten auf vier unterschiedenen Ebenen beschrieben. Auf der *Bereichsebene (domain level)* werden die Fachbegriffe des zu modellierenden Bereichs sowie die Relationen zwischen ihnen und darauf aufbauende Strukturen dargestellt. Auf der *Inferenzebene (inference level)* werden Metaklassen und Wissensquellen eingeführt, die die Rolle von Fachbegriffen und Relationen der Bereichsebene beim Problemlösen beschreiben. Auf der *Aufgaben- oder Problemlösungsebene (task level)* werden die spezifischen Problemlösungsverfahren anhand der festgelegten Metaklassen und Wissensquellen der Inferenzebene formuliert. Die *Strategieebene (strategic level)* schließlich beschreibt Pläne und Metaregeln, die der Erfolgsüberwachung und flexiblen Handhabung verschiedener für den Bereich maßgeblicher Problemlösungsstrategien der Aufgabenebene dienen.

Die Konstruktion von Wissensmodellen wird unterstützt durch Beschreibungssprachen und Kataloge von sog. *Interpretationsmodellen*. Auf dieser Basis wird eine modellgeleitete Analyse des Problemlösungswissens von Experten anhand der erhobenen Daten vorgenommen („model-driven" statt „data-driven"). Das konzeptuelle (Beschreibungs-)modell ist die Grundlage für ein Designmodell des zu erstellenden Systems. Eine grobe Übersicht über die KADS-Vorgehensweise zeigt die Abbildung 7.6.

Die auf die Problemlösung bezogene Einteilung von Bereichskonzepten und -relationen auf der Inferenzebene ist ein wesentliches Moment der KADS-Vorgehensweise, da hier eine *Interpretation* der Begriffe im Hinblick auf ihren Gebrauch bei Problemlösungen vorgenommen wird; dies soll am Beispiel näher beleuchtet werden. Typische *Metaklassen* in der diagnostischen Problemlösung der Medizin sind etwa *Symptome* (z.B. „Brustschmerz"),

Prädispositionsfaktoren (z.B. „Alter"), *Verdachtsdiagnosen, Enddiagnosen, Therapien.* Zu beachten ist, daß ein und derselbe Fachbegriff in verschiedene Metaklassen fallen kann; etwa kann der Begriff „Infektion" für die Metaklassen *Verdachtsdiagnosen, Enddiagnosen,* aber auch *Symptome* relevant sein.

Auch die Relationen zwischen Konzepten der Bereichsebene – „atomares" Inferenzwissen in Form bereichsspezifischer Regeln – werden durch Elemente der Inferenzebene im Hinblick auf die Problemlöseaufgabe abstrakt eingeteilt: *Wissensquellen* setzen Metaklassen zueinander in Beziehung und klassifizieren das verfügbare Inferenzwissen z.B. als verschiedene Typen von Regeln. Als Wissensquellen im oben betrachteten diagnostischen Aufgabenbereich sind etwa *Verdachtsgenerierungsregeln, Verdachtsüberprüfungsregeln, Indikationsregeln* zu benennen. Die benannten Metaklassen und Wissensquellen ergeben das Vokabular für die Charakterisierung der Problemlösungsstrategie auf dem task level des konzeptuellen Modells. Eine Skizze von Metaklassen und Wissensquellen für die Hypothesize-and-Test-Strategie diagnostischer Problemlösung zeigt Abb. 7.7.

Die bereits erwähnte KADS-Modellbibliothek stellt zur Unterstützung der Konstruktion von Wissensmodellen eine Taxonomie von *Interpretationsmodellen* zur Verfügung, die für bestimmte Aufgabenklassen (z.B. Diagnose oder Design) einen Problemlösemechanismus unabhängig von einem konkreten Aufgabengebiet angeben. Sie sind auf der Inferenz- und Aufgabenebene spezifiziert. Sofern sich eine Problemlöseaufgabe mit Hilfe eines dort vorhandenen Interpretationsmodells charakterisieren läßt, kann beim Entwurf des wissensbasierten Systems auf dafür bereits erarbeitete Modellelemente zurückgegriffen werden. Einen Überblick über den Katalog der in KADS bereits betrachteten, teils aber erst nur fragmentarisch beschriebenen Interpretationsmodelle gibt die Abb. 7.8; dies sollte als offener Katalog verstanden werden, der mit zunehmender systematischer Durchdringung des Feldes sicherlich noch fortentwickelt wird. Eine andere Möglichkeit der Einteilung zeigt Abb. 7.9.

Der wesentliche Schritt der Formalisierung von Expertenwissen liegt in der Erstellung des konzeptuellen Modells für einen gegebenen Problemlösungsbereich. Die sich anschließende Design-Aufgabe betrifft die Umsetzung in ein operationales System. Hierauf wird in Teil 7.1.4 noch näher eingegangen.

Die Bedeutung der konzeptuellen Modelle ergibt sich daraus, daß eine Bottom-Up-Analyse des Anwendungsbereichs schwer ohne Vorwissen durchführbar ist – jedoch ist das in KADS berücksichtigte Vorwissen abstrakt und auf die vergleichende Interpretation vieler konkreter Problemlösungsbereiche bezogen. Die Hauptleistung von KADS besteht in der Bereitstellung begrifflichen Repertoires für die Aufgabe der Knowledge Engineers, was man als KADS-spezifische Einschätzung des Tuns sehen sollte. Die Anwendung dieses Repertoires bleibt jedoch der Interpretations- und Urteilsfähigkeit der Knowledge Engineers vorbehalten.

KADS wurde bislang eher als ein Programm mit einer bestimmten Philosophie der Wissensmodellierung für Expertensysteme statt als ein operationalisierbares Verfahren bewertet.

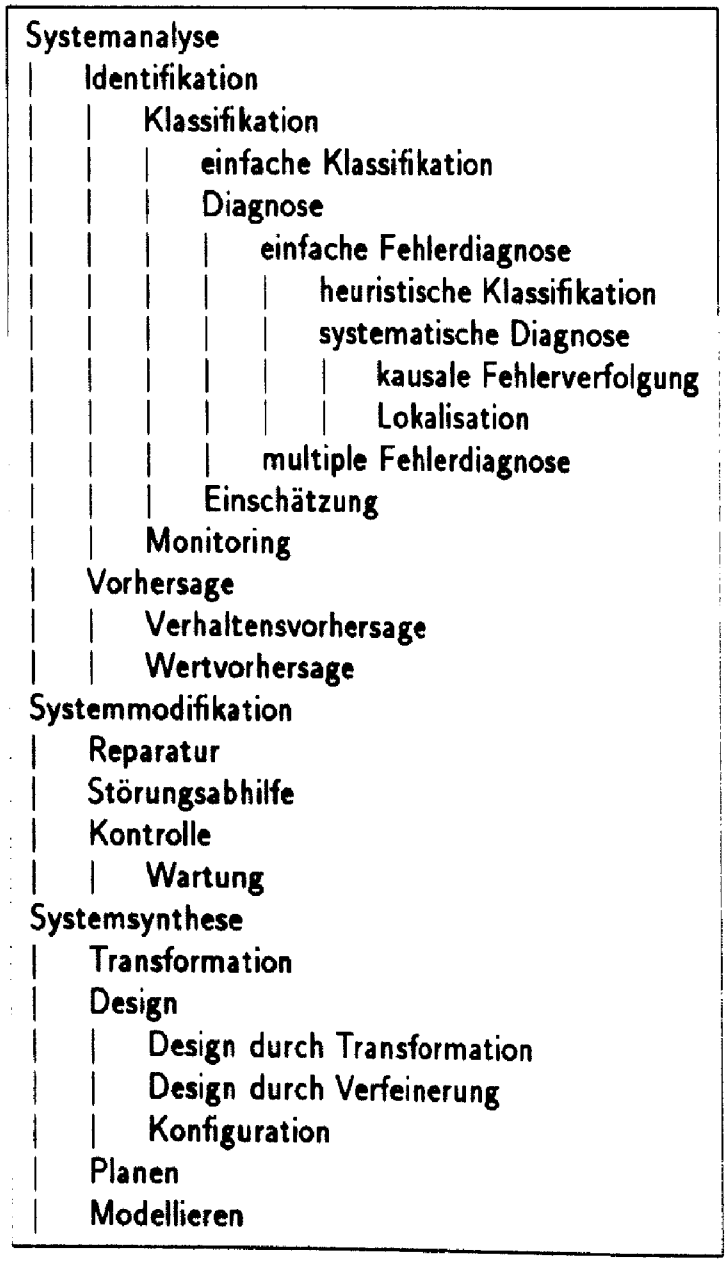

Abbildung 7.8: KADS-Modellbibliothek: Interpretationsmodelle für bestimmte Aufgabenklassen (nach [Breuker *et al.*, 1987])

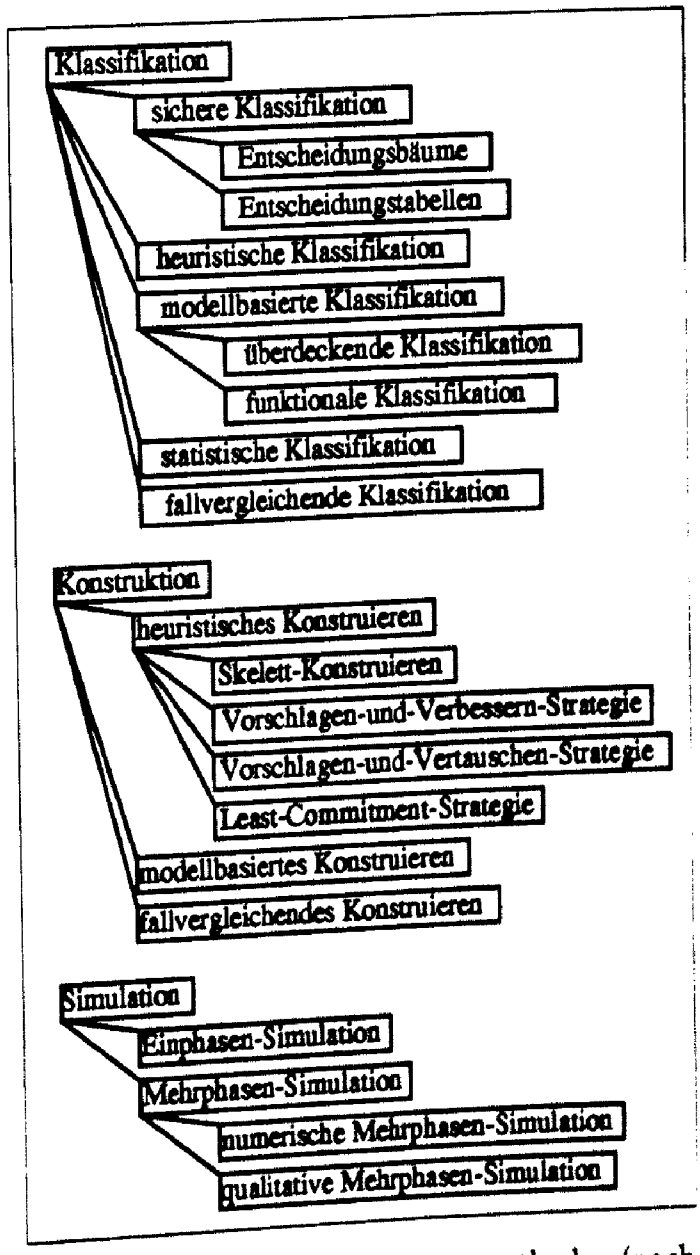

Abbildung 7.9: Übersicht über Problemlösungsmethoden (nach [Puppe, 1990])

An einer Operationalisierung von KADS-Modellierungen wird jedoch gegenwärtig verstärkt gearbeitet, vgl. etwa [van Harmelen und Balder, 1992] und die dort genannten weiteren Ansätze.

KADS ist mittlerweile in etwa 50 Projekten verwendet worden, darunter z.B. für das zur Entdeckung von Kreditkartenmißbrauch gewinnbringend eingesetzte System Fraudwatch [Porter und Taylor, 1990]. Eine gute Übersicht über den Entwicklungsstand am Ende des ersten KADS-Projekts findet sich in [Wielinga *et al.*, 1992]. Das Nachfolgeprojekt KADS-II stellt den Entwicklungsprozeß in einen umfassenden Rahmen von der Projektdefinition bis zur Wartung.

7.1.3.5 Indirekter, direkter und automatischer Wissenserwerb

Die in den bisherigen Ausführungen vorherrschende Betrachtung des Wissenserwerbs (Wissensakquisition) sah diesen als einen Prozeß, der starke Beteiligung eines Knowledge Engineers erfordert. Dies wird immer dann der Fall sein, wenn in einem Projekt die Natur des zu modellierenden Problemlöseprozesses erst noch geklärt werden muß (Problemcharakterisierung) oder wenn sich das Bereichswissen nur im Gespräch zwischen Knowledge Engineer und Experten abklären läßt. In gut verstandenen Bereichen wie der Diagnostik oder in einer Projektphase, die die Problemcharakterisierung bereits erfolgreich abgeschlossen hat, ist es aber auch ein verfolgtes Vorgehen, zunächst eine Shell (vgl. Abschnitt 7.1.5.4) zu entwickeln und das Bereichswissen „in der Breite" direkt vom Experten eingeben zu lassen oder gar eine automatische Erstellung einer Wissensbasis zu erwägen. Entsprechend dieser Vorstellungen sind folgende Grundarten des Wissenserwerbs unterschieden worden [Puppe, 1991]:

- *Indirekter Wissenserwerb:* Ein Wissensingenieur befragt einen Experten und formalisiert die Ergebnisse für das Expertensystem; dieses Verfahren ist aufwendig und naturgemäß fehleranfällig, da das Wissen auf dem Umweg über den Knowledge Engineer akquiriert wird.
- *Direkter Wissenserwerb:* Der Experte formalisiert sein Wissen selbst. Dieses Vorgehen erfordert ein komfortables Wissenserwerbssystem, ist aber nur in gut verstandenen Anwendungsbereichen durchführbar.
- *Automatischer Wissenserwerb:* Das Expertensystem extrahiert sein Wissen selbständig aus Falldaten oder verfügbarer Literatur: Hierzu gibt es erste Ansätze, die bisher aber nicht praxisreif sind.

Das Thema „Wissensmodellierung" befaßt sich in erster Linie mit dem *indirekten* Wissenserwerb. Einige Bemerkungen zum direkten und zum automatischen Wissenserwerb sollen diesen Abschnitt beschließen.

Beim *direkten* Wissenserwerb ist der Grundgedanke, daß eine „Programmierumgebung für Experten" geschaffen wird, die Experten von den formalen Anforderungen der Wissensba-

siserstellung weitgehend befreit und es ihnen ermöglicht, ihr spezifisches Wissen selbst zu formalisieren und ggf. zu warten. Solche *Wissenserwerbssysteme* sind Komponenten einer Expertensystem-Shell zum Aufbau und zur Wartung einer Wissensbasis durch Experten. Sie sollen eine dem Experten bekannte Wissensrepräsentation verwenden und sollen komfortabel sein hinsichtlich der Eingabeunterstützung, der sofortigen Überprüfbarkeit von Änderungen (mit Hilfe einer Testfälle-Datenbank) und der Durchführung von Konsistenztests. Die meisten leistungsfähigen Wissenserwerbssysteme betreffen bisher die Diagnostik; Beispiele findet man in [Puppe, 1991].

Im Hinblick auf einen *automatischen Wissenserwerb* wird einerseits die Extraktion von Sachverhaltswissen aus Falldaten und andererseits aus Texten (Protokollen oder verfügbarer Fachliteratur) diskutiert. Es geht im Grunde also um lernfähige Systeme, die bislang nur in der KI-Grundlagenforschung behandelt werden, allerdings mit stark zunehmendem Interesse. Auch hier beziehen sich für Expertensysteme relevante Entwicklungen fast ausschließlich auf die Diagnostik, vgl. dazu [Puppe, 1991]. Am erfolgversprechendsten erscheinen zur Zeit Ansätze, die aus großen Falldatenbanken auf Regularitäten schließen. Das System KRITON [Diederich, 1989] verwendet Konstruktgitter- und Protokollanalyse-Techniken, um – allerdings sehr einfache – Regeln direkt aus Interviewdaten zu generieren. Dieser Ansatz ist jedoch wegen erkennbarer Grenzen nicht weiter verfolgt worden. Die Extraktion von Sachverhaltswissen aus Texten ist Gegenstand jüngerer Projekte zum Verstehen natürlicher Sprache (wie LILOG); die Anwendung für die automatische Wissensakquisition ist zwar ein forschungsleitendes Ziel, gegenwärtig für den Bereich Expertensysteme aber noch nicht von praktischer Bedeutung.

7.1.4 Problemlösungsmethoden in Expertensystemen

7.1.4.1 Problemklassen und Problemlösungsmethoden

In diesem Abschnitt werden auf den Grundtechniken der Wissensrepräsentation aufbauende Problemlösungsmethoden für die *Problemklassen* Klassifikation, Konstruktion und Simulation skizziert. Vorausgeschickt sei, daß sich aus der Kenntnis der Problemklasse bereits grob auf die nötigen Grundtechniken der Wissensrepräsentation und -verarbeitung schließen läßt. Eine Übersicht über diese Zuordnung zeigt die Abb. 7.10. In den angeführten Problemklassen läßt sich häufig sogar festlegen, wie bereichsspezifisches Wissen zur Problemlösung verwendet wird. Einen Algorithmus, der dies leistet, bezeichnen wir als *Problemlösungsmethode.* Es ist sinnvoll, zwischen schwachen und starken Methoden zu unterscheiden. Schwache (oder *Basis*problemlösungsmethoden) sind solche, die offen für eine große Vielfalt von Wissensrepräsentationen sind, wie z.B. „Regeln mit Vorwärtsverkettung". Sie sind zwar breit anwendbar, stellen aber keine große Hilfe beim Wissenserwerb dar, da sie Repräsentation und Funktion bereichsspezifischen Wissens nicht festlegen. Eine weitgehende Festlegung in dieser Hinsicht charakterisiert gerade die starken Problemlösungsmethoden, die dadurch zwar weniger flexibel sind, aber – sofern anwend-

bar – umfassende Unterstützung beim Wissenserwerb und beim Systementwurf leisten. Im günstigsten Fall kann so viel festgelegt sein, daß das Wissen einfach durch Instantiierung vorgegebener Objekttypen und Angabe der Beziehung zwischen Instanzen eingegeben werden kann. Im folgenden werden die wichtigsten starken Problemlösungsmethoden vorgestellt. Eine ausführliche Behandlung findet sich in [Puppe, 1990].

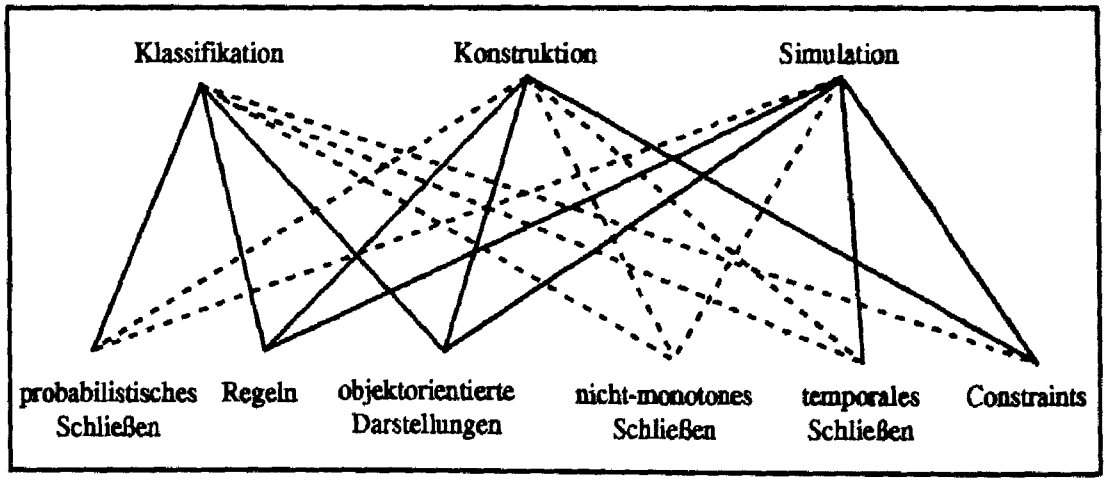

Abbildung 7.10: Grobe Zuordnung von Problemlösungstypen zu Grundtechniken der Wissensrepräsentation und -verarbeitung – nach [Puppe, 1991]

7.1.4.2 Klassifikation

Die Problemklasse *Klassifikation* (häufig auch Diagnose genannt) umfaßt alle Problembereiche, bei denen die Lösung aus einer vorgegebenen Menge von Alternativen ausgewählt wird, wie z.B. in der medizinischen Diagnostik, bei der Fehlersuche in technischen Geräten, bei der Qualitätskontrolle, bei der Prozeßüberwachung in der Fertigung, bei der Auswahl von Produkten oder der Überprüfung, welche juristischen Bestimmungen auf eine Situation anwendbar sind[2].

Historisch haben sich die Diagnostik-Expertensysteme aus dem medizinischen Bereich heraus entwickelt. Die praktischen Erfolge von Diagnostiksystemen liegen jedoch fast ausschließlich im technischen Bereich, während die medizinischen Systeme mit wenigen Ausnahmen Demonstrationsprototypen geblieben sind.

Das für die Diagnostik typische Zurückschließen von Beobachtungen auf Systemzustände bzw. Objekte, die die Beobachtungen hervorrufen, ist eine Form der *Abduktion*: wenn eine Ursache U ein Symptom S verursacht, und das Symptom S wird beobachtet, dann ist U eine mögliche Erklärung für das Symptom S.

[2]Wenn nicht anders ausgewiesen, sind die Abbildungen in diesem Abschnitt [Puppe, 1990] entnommen.

Die Abduktion ist natürlich keine logisch zwingende Schlußweise wie die Deduktion, da eine Beobachtung viele Ursachen haben kann. Die dadurch bedingte Unsicherheit in der Diagnosebewertung läßt sich durch Auswertung zusätzlicher Daten reduzieren. Weil die Symptomerhebung aufwendig und risikoreich sein kann, müssen Kosten und Nutzen diagnostischer Untersuchungen sorgfältig gegeneinander abgewogen werden, z.B. reichen in Routinefällen wenige Daten aus, während bei komplexeren Problemen eine umfassendere Symptomerhebung erforderlich ist. Besonders in der Prozeßdiagnostik ist auch die zeitliche Entwicklung von Symptomen und ihre Änderung unter dem Einfluß von Therapiemaßnahmen aufschlußreich. Da der Diagnostiker nicht immer mit Diagnosen und Therapien warten kann, bis alle relevanten Symptome erhoben sind, muß er plausible Hypothesen aufstellen und diese bei gegenteiliger Evidenz zurückziehen können. Die Situation verkompliziert sich dadurch, daß „Widersprüche" (z.B. je ein Symptom spricht für und gegen eine Diagnose) viele Ursachen haben können: außer einer falschen Interpretation gehören dazu Fehler bei der Symptomerhebung, eine noch nicht erkannte Ursache für eines der Symptome oder unzureichendes Wissen über das Anwendungsgebiet. Aus diesen Überlegungen ergeben sich folgende Anforderungen an ein Diagnosesystem, die insgesamt zu berücksichtigen allerdings eine noch sehr schwierige Aufgabe darstellt:

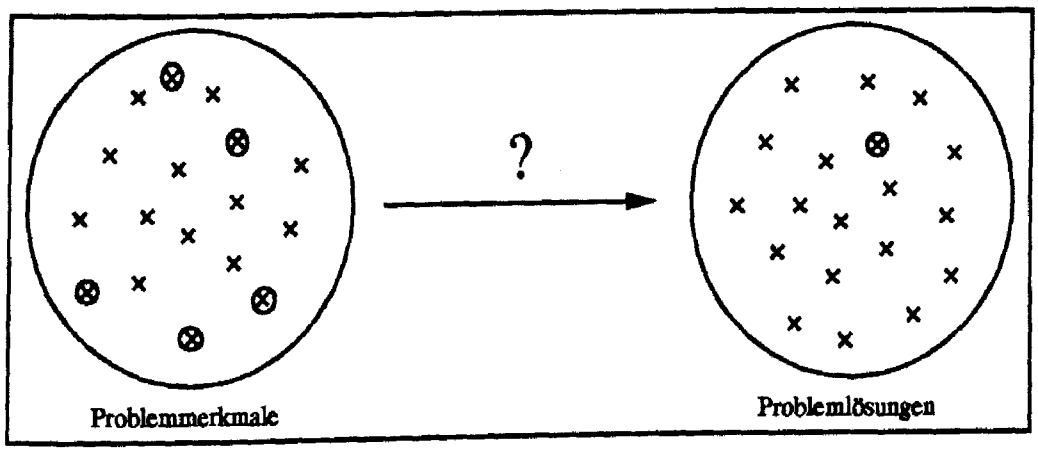

Abbildung 7.11: Basisstruktur der Diagnostik mit Problemmerkmalen (Symptomen) und Problemlösungen (Diagnosen)

- Diagnosebewertung mit unsicherem Wissen,
- Diagnosebewertung mit unvollständigem Wissen,
- Plausibilitätskontrolle der Eingabedaten,
- Erkennen von Mehrfachdiagnosen,
- adäquate Behandlung von Widersprüchen,
- kosteneffektive Symptomerhebung,
- Auswertung von Folgesitzungen.

Standardtechniken zur Bewältigung von Diagnoseproblemen, die auch in psychologischen Untersuchungen bestätigt wurden [Elstein *et al.*, 1978], sind die Verwendung eines diagnostischen Mittelbaus, die hypothetisch-deduktive Vorgehensweise und die Differentialdiagnostik. Der diagnostische Mittelbau beschreibt die Verdichtung der Rohdaten zu den Enddiagnosen über einfache Symptominterpretationen und Grobdiagnosen [Clancey, 1984; 1985], z.B.:

Rohdaten:	Puls = 100, Blutdruck = 80	→
Einfache Symptominterpretation:	hoher Schockindex	→
Grobdiagnose:	Kreislaufschock	→
Feindiagnose:	Ursache des Kreislaufschocks	

Während die Datenvorverarbeitung, d.h. der Schluß von den Rohdaten zu einfachen Symptominterpretationen, meist mit einfachen und sicheren Regeln möglich ist, basiert die diagnostische Auswertung beim Übergang zu Grob- und Feindiagnosen gewöhnlich auf unsicherem Wissen. In manchen Anwendungsbereichen gibt es eine strenge Diagnosehierarchie. Dort läßt sich die „Establish-Refine-Technik" anwenden, bei der zunächst eine Diagnoseklasse bestätigt wird, die anschließend zu einem Hierarchienachfolger verfeinert wird. In vielen Bereichen gibt es jedoch multiple Diagnosehierarchien, die eine flexiblere Technik erfordern. Dazu eignet sich die *hypothetisch-deduktive Strategie*, bei der ausgehend von den Anfangssymptomen mittels Vorwärtsverkettung Diagnosen verdächtigt werden, die dann gezielt mittels Rückwärtsverkettung überprüft werden. Zur Überprüfung werden gegebenenfalls zusätzliche Symptome angefordert. Schließlich wird die Entscheidung für oder gegen eine Diagnose nicht isoliert getroffen, sondern „differentialdiagnostisch" durch expliziten Vergleich der wahrscheinlichsten Diagnosen mit ihren Konkurrenten.

Das Wissen zur Lösung diagnostischer Probleme kann man auf verschiedene Weise repräsentieren. Wenn das verfügbare Wissen sichere Schlußfolgerungen erlaubt, eignen sich einfache Entscheidungsbäume und Entscheidungstabellen. Beim Umgang mit unsicherem Wissen sind die vier wichtigsten Arten die statistische, die heuristische, die fallbasierte und die kausale Diagnostik. Bei der statistischen Vorgehensweise geht man von Symptom-Diagnose-Tabellen aus, deren Wahrscheinlichkeiten durch statistische Auswertung großer Falldatenbanken errechnet wurden. Die Wahrscheinlichkeiten kann man mit dem in Abschnitt 7.1.2.4 beschriebenen Theorem von Bayes verknüpfen. Ein erfolgreiches statistisches Diagnoseprogramm ist das System von de Dombal [de Dombal *et al.*, 1972] zur Differentialdiagnose akuter Bauchschmerzen.

Die heuristische Vorgehensweise unterscheidet sich von der statistischen dadurch, daß die Symptom-Diagnose-Wahrscheinlichkeiten von Experten geschätzt werden. Der Erfolg heuristischer Diagnosesysteme resultiert aus ihrer Flexibilität, Zusatzwissen berücksichtigen zu können, z.B. Symptomkombinationen und Ausnahmen in Regeln, verschiedene Arten des diagnostischen Mittelbaus und Kosten-Nutzen-Analysen zur Indikation aufwendiger diagnostischer Untersuchungen. Ein Beispiel für ein flexibles Werkzeug zur Erstellung heuristischer Diagnosesysteme ist D3 [Bamberger *et al.*, 1991].

Die fallvergleichende Diagnostik basiert ebenso wie die statistische Diagnostik auf der Auswertung großer Falldatenbanken. Jedoch werden dabei keine statistischen Mittelwerte errechnet, sondern zu einem neuen Fall möglichst ähnliche Fälle in der Falldatenbank gesucht. Zur Bewertung der Unterschiede zwischen zwei Fällen ist Zusatzwissen über das Ähnlichkeitsmaß erforderlich, insbesondere zur relativen Gewichtung der Symptome und der Symptomausprägungen. Ein Beispiel für ein fallvergleichendes Diagnosesystem ist CCC+ [Puppe, 1991].

Bei der kausalen Diagnostik werden Diagnosehypothesen danach bewertet, wie gut sie die beobachteten Symptome in einem vorgegeben Modell erklären können. Dabei gibt es zwei Arten von Modellen: „funktionale Modelle", bei denen von dem Normalverhalten des Systems ausgegangen wird und Fehler als Änderungen in dem Modell dargestellt werden, und „Fehlermodelle", bei denen die verschiedenen möglichen Fehlerzustände bereits explizit in dem Modell repräsentiert und durch Regeln verbunden sind. Beispielsysteme für funktionale Modelle sind HTE [Davis, 1984] oder GDE+ [Struß, 1989] und für Fehlermodelle ABEL [Patil *et al.*, 1982], MOLE [Eshelman, 1988] oder FEMO [Goos, 1990].

7.1.4.3 Konstruktion

Der Problemlösungstyp *Konstruktion* unterscheidet sich von der Diagnose dadurch, daß die Lösung aus kleinen Bausteinen zusammengesetzt werden muß, anstatt ausgewählt werden zu können. Zur Konstruktion gehören *Planung*, bei der eine Folge von Handlungen (Operatoren) zum Erreichen eines Zielzustandes gesucht wird (Abb. 7.12), *Konfigurierung* zum Entwurf eines Objektes aus Komponenten, das bestimmten Anforderungen genügen muß (Abb. 7.13) und *Zuordnung*, bei der zwei vorgegebene Objektmengen unter Beachtung von Randbedingungen einander zugeordnet werden (Abb. 7.14). Beispiele sind etwa die Festlegung von Bearbeitungsschritten für Werkstücke, die einen Rohling in das gewünschte Produkt überführen, die Erzeugung einer vollständigen Komponentenliste und eines Layouts für Computer anhand geforderter Leistungen oder die Stundenplanerstellung ausgehend von Angaben über Räume, Lehrer, Fächer usw. Häufig entspricht die zeitliche Anordnung von Operatoren bei der Planung einer räumlichen Anordnung von Komponenten bei der Konfigurierung: so kann man „eine Reise planen" oder „einen Reiseplan konfigurieren". Während wir hier nur einen kurzen Abriß geben können, bietet Kapitel 7.2.3 eine ausführliche Darstellung von Planung und Konfigurierung.

Konstruktionsprobleme zeichnen sich gewöhnlich durch einen sehr großen Suchraum aus, z.B. bei einem Zuordnungsproblem mit zwei Objektgruppen von je n Elementen in der Größenordnung von $n!$ oder bei einem Planungsproblem mit einer durchschnittlichen Operatorsequenz der Länge n und durchschnittlich m Alternativen bei der Operatorauswahl in der Größenordnung von m^n. Die Grundlage für Konstruktionsmethoden sind daher häufig Suchtechniken. Jedoch scheitert eine einfache Breiten- oder Tiefensuche meistens an dem zu hohen Aufwand. Durch Einbezug von heuristischem Wissen läßt sich der Suchaufwand

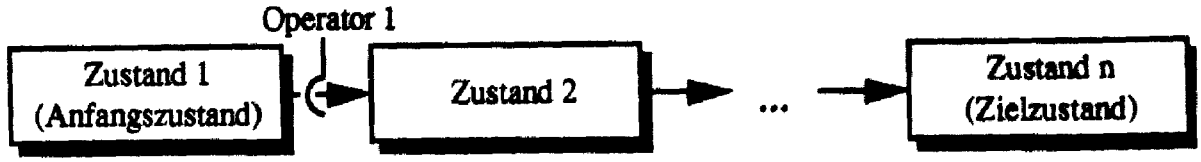

Abbildung 7.12: Grundstruktur von Planungsproblemen: Gegeben sind Ausgangs- und Zielzustand, gesucht ist eine Sequenz von Operatoren, die den Ausgangs- in den Zielzustand überführen.

oft erheblich reduzieren. Dafür eignet sich insbesondere die Hill-Climbing-Strategie, bei der an jeder Verzweigung Wissen verfügbar sein muß, um die lokal beste Alternative auszuwählen. Da Sackgassen nicht ausgeschlossen werden können, sind zum Backtracking außerdem Rücksetztechniken erforderlich, die effizient mit einem TMS (Abschnitt 7.1.2.5) und gegebenenfalls mit Zusatzwissen realisiert werden können.

Weitere Grundtechniken sind Abstraktion und Modularisierung, mit denen das Ausgangsproblem in eine Menge von kleineren Problemen zerlegt wird, die möglichst unabhängig voneinander lösbar sein sollen. Die Hauptschwierigkeit ist der Umgang mit den oft unvermeidlichen Verletzungen der Unabhängigkeitsannahme („*Interaktionen*"). Schließlich kann man auch versuchen, das Problem umzuformulieren, um es in einer anderen Darstellung einfacher lösen zu können. Ein Beispiel ist die Generierung von (einfachen) Constraints aus der ursprünglichen Problemstellung, die dann mit einem Constraint-Propagierungssystem ausgewertet werden können.

Aus diesen Grundideen lassen sich einige effiziente Problemlösungsmethoden ableiten, die bei ausreichendem Wissen den Suchraum so handhabbar machen, daß nur noch sehr wenige Alternativen ausprobiert werden müssen:

1. *Skelett-Konstruieren* (Abstraktion + Modularisierung + lokale Auswahlheuristiken): Das Konstruktionswissen ist hierarchisch in einem nicht-rekursiven Und-Oder-Graphen strukturiert, dessen Expansion mit heuristischen Regeln gesteuert wird. Und-Knoten repräsentieren Teil-von-Beziehungen oder Skelett-Pläne, von denen jeder Nachfolgeknoten bearbeitet werden muß. Oder-Knoten repräsentieren Spezialisierungsbeziehungen oder Konstruktionsalternativen, von denen nur ein Nachfolgeknoten ausgewählt wird. Spezialfälle des Skelett-Konstruierens umfassen die Auswahl und Verfeinerung von Skelett-Plänen aus einer Bibliothek und die Phaseneinteilung, bei der ein einziger Standard-Skelett-Plan benutzt wird.
2. *Vorschlagen-und-Verbessern* (Hill-Climbing + wissensbasiertes Rücksetzen): Es wird mit heuristischen Regeln immer die lokal beste Alternative vorgeschlagen. Bei Sackgassen wird mit heuristischen Regeln eine geeignete Korrekturmöglichkeit ermittelt und mit einem TMS umgesetzt. Eine Erweiterung von Vorschlagen-und-Verbessern für Zuordnungsprobleme ist Vorschlagen-und-Vertauschen.
3. *Least-Commitment* (Problemumformulierung + Constraint-Propagierung): Zunächst

wird das Problem strukturell umformuliert, und das einfachere Problem kann dann häufig mit Constraint-Propagierung oder Basissuchstrategien gelöst werden.

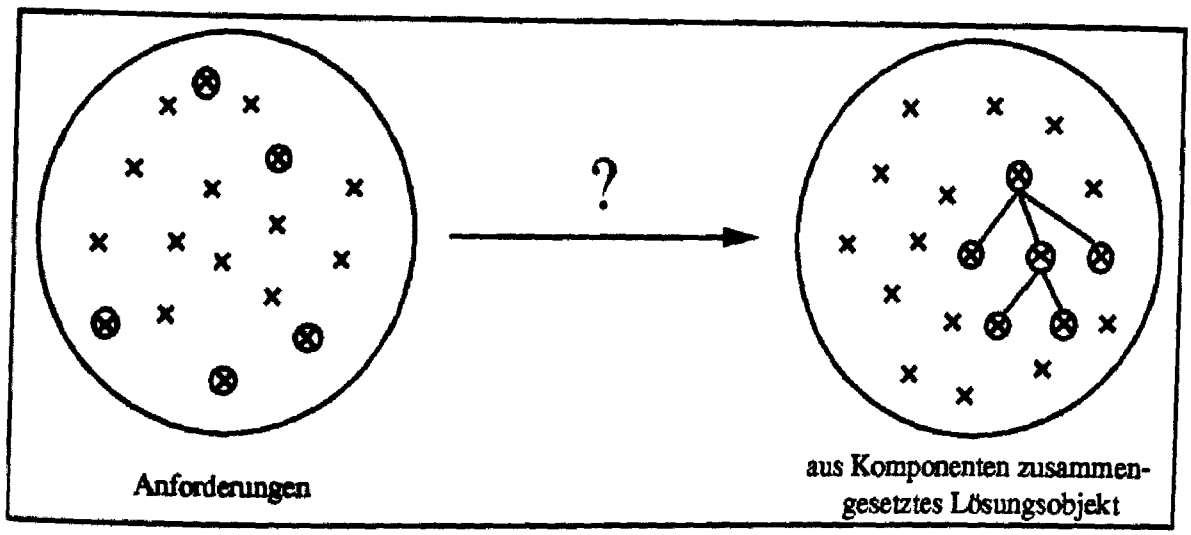

Abbildung 7.13: Grundstruktur von Konfigurierungsproblemen, bei denen aufgrund von Anforderungen das Lösungsobjekt aus Komponenten zusammengesetzt wird.

Beispielsysteme sind zum Konstruieren aus Skeletten Friedlands MOLGEN [Friedland, 1979; Friedland und Iwasaki, 1985], das molekulargenetische Experimente plant, zur Phaseneinteilung das berühmte XCON (R1) [Barker und O'Connor, 1989], das Computer konfiguriert (s. auch S. 785), zum Vorschlagen-und-Verbessern SALT [Marcus, 1988], das Fahrstühle konfiguriert (s. auch S. 785), für Vorschlagen-und-Vertauschen REST [Poeck, 1990], das Schulstundenpläne generiert, und zum Least-Commitment Stefik's MOLGEN [Stefik, 1981] und GARI [Descotte und Latombe, 1985], das Arbeitspläne für Werkstücke erstellt.

Diese heuristischen Techniken funktionieren jedoch nur, wenn man viel Erfahrungswissen über den Anwendungsbereich hat. Für Planungsprobleme mit wenig Erfahrungswissen eignet sich die Differenzenmethode (means-ends-analyis), bei der zunächst die Differenz zwischen Ausgangs- und Zielzustand festgestellt und dann ein Operator gesucht wird, der diese Differenz maximal verringert. Falls der ausgewählte Operator nicht direkt anwendbar ist oder seine Aktion den Zielzustand nicht vollständig herleitet, wird für die neue Differenz rekursiv ein Unterziel generiert, bis ein vollständiger Plan vorliegt. Im allgemeinen wird auch dabei zunächst mit Groboperatoren geplant, die später verfeinert werden.

Bei der Differenzenmethode kann die obengenannte Schwierigkeit von Interaktionen zwischen unabhängig voneinander verfeinerten Groboperatoren besonders leicht auftreten, da in die Grobplanformulierung (im Gegensatz zum Skelett-Konstruieren) kein Erfahrungswissen eingeht. Für die Behandlung von Interaktionen gibt es verschiedene Techniken, die wir am Beispiel einer „negativen" Interaktion im Problem „Streiche die Leiter und die

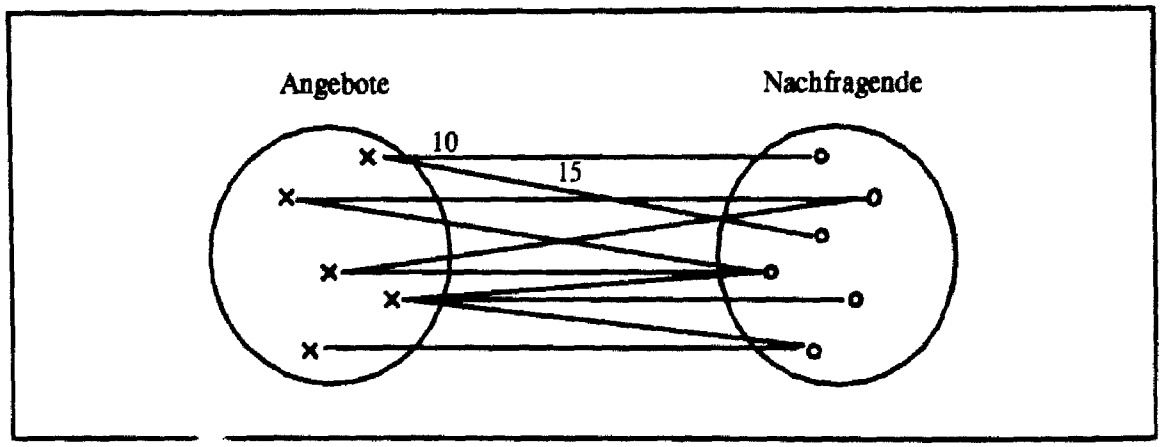

Abbildung 7.14: Grundstruktur von Zuordnungsproblemen: Gegeben sind mindestens zwei Objektmengen und verschiedenartige Randbedingungen (z.B. die durch Zahlen angedeuteten Präferenzen), gesucht ist eine möglichst optimale Zuordnung.

Decke" skizzieren (beginnt ein ungeschickter Plan mit dem Streichen der Leiter, so ist das Streichen der Decke wegen der nun unbrauchbaren Leiter für eine Weile blockiert):

- Der Plan wird umgestellt, so daß der blockierte Operator vor dem ihn blockierenden Operator ausgeführt wird (lineares Planen mit Verschieben, wie etwa in [Waldinger, 1977]).
- Man faßt einen Plan nicht als Folge auf, sondern als eine nur teilweise geordnete Menge von Operatoren (nicht-lineares Planen) und legt die Reihenfolge nur soweit fest, wie zwingende Gründe dazu bestehen. Stellt man – etwa bei der Verfeinerung eines Plans – fest, daß eine mögliche Operator-Anordnung zu einer Blockierung führt, schließt man diese Anordnung durch Einführen eines Constraints aus. Die endgültige Reihenfolge der Operatoren wird erst festgelegt, wenn eine eindeutige Entscheidung (z.B. durch Constraint-Propagierung) möglich ist (Least-Commitment, s.o). Im Beispiel wird die Reihenfolge beider Operationen erst festgelegt, nachdem die Interaktion entdeckt wurde.
- Wenn es nicht anders möglich ist, wird ein zusätzlicher Planschritt generiert[3], der die verletzte Voraussetzung des Operators wiederherstellt. In diesem Fall würde man sich zum Streichen der Decke eine neue Leiter besorgen oder warten, bis die alte getrocknet ist.

Eine vereinheitlichende Darstellung der diesen Verfahren unterliegenden abstrakten Struktur findet man in [Hertzberg und Horz, 1989]. Wenn die Planerstellung selber kompliziert wird, weil z.B. verschiedene Methoden miteinander kombiniert werden müssen, dann kann auch die Anwendung von Planungstechniken zur Planerstellung sinnvoll sein (Meta-Planen,

[3]Das kann sowohl beim linearen wie beim nichtlinearen Planen erforderlich sein

Beispiel: Stefiks MOLGEN [Stefik, 1981]). Eine Integration vieler Konstruktionstechniken ist auch in der Planungs- und Konfigurierungsshell PLAKON [Cunis *et al.*, 1991], s. S. 786, realisiert.

7.1.4.4 Simulation

Mit der *Simulation* wird das Verhalten eines Systems vorhergesagt. Sie dient oft zur Plausibilitätskontrolle von Problemlösungen, z.B. zur Überprüfung einer Verdachtsdiagnose in einem kausalen Modell oder zum Testen eines Planes. Die Simulation kann man als Komplement zum Planen auffassen: während beim Planen Anfangs- und Zielzustand gegeben sind und Aktionen zum Erreichen des Zieles gesucht werden, sind bei der Simulation der Anfangszustand und die Aktionen (Prozesse) bekannt und Folgezustände werden gesucht. Ein Zustand wird durch eine Menge von Objekten mit Parametern charakterisiert. Während man bei der Einphasensimulation aus einer partiellen Zustandsbeschreibung nur die vollständige Zustandsbeschreibung ermittelt, wird bei der Mehrphasensimulation eine Folge von Zuständen generiert, die das Verhalten bzw. die möglichen Verhaltensweisen des Systems beschreiben.

Für die Mehrphasensimulation gibt es verschiedene *Simulationstypen*: während bei der numerischen Simulation das Verhalten durch eine Folge von Zahlenwerten der Parameter zu verschiedenen Zeitpunkten angegeben wird, wird bei der analytischen Simulation das Verhalten durch algebraische Umformungen der zugrundeliegenden mathematischen Gleichungen und Auflösung nach den interessierenden Parametern beschrieben. Die qualitative Simulation entspricht der numerischen Simulation mit dem Unterschied, daß der Wertebereich der Parameter auf qualitative interessante Werte beschränkt wird. So kann ein Parameter nur größer, kleiner oder gleich seinen Landmarkenwerten sein. Landmarkenwerte für Wassertemperatur wären z.B. Gefrier- und Siedepunkt. Die qualitative Simulation hat gegenüber der numerischen Simulation den Vorteil, daß sie auch mit partiellem Wissen über das System durchgeführt werden kann, und gegenüber der analytischen Simulation, daß weniger mächtige Berechnungstechniken erforderlich sind. Deswegen eignet sie sich für Expertensysteme ganz gut.

Ausgangspunkt der qualitativen Simulation ist eine Wertzuweisung für einige Parameter des Ausgangszustandes. Aufgrund der Beziehungen zwischen den Parametern, die als Constraints oder Regeln formuliert sein können, wird dann eine möglichst vollständige Zustandsbeschreibung generiert, indem die qualitativen Werte und die Änderungstendenzen aller Parameter bestimmt werden. Diese Phase nennt man Intrastate-Analyse. Bei der folgenden Interstate-Analyse werden die Änderungstendenzen der Parameter soweit propagiert, daß zumindest ein Parameter seinen qualitativen Wert ändert. In diesem Folgezustand wird dann mittels einer Intrastate-Analyse die vollständige Parameterbelegung bestimmt und anschließend wieder zu einem Folgezustand übergegangen. Das Hauptproblem ist die Behandlung von Unsicherheiten über den Folgezustand, z.B. wenn ein Para-

meter sowohl einem positiven und einem negativen Einfluß ausgesetzt ist. Die Bestimmung des Summeneffektes würde zusätzliches Wissen über die Größenordnung der Einflüsse oder sogar quantitative Berechnungen erfordern. Wenn das nicht möglich ist, muß das Simulationssystem in alle möglichen Folgezustände verzweigen.

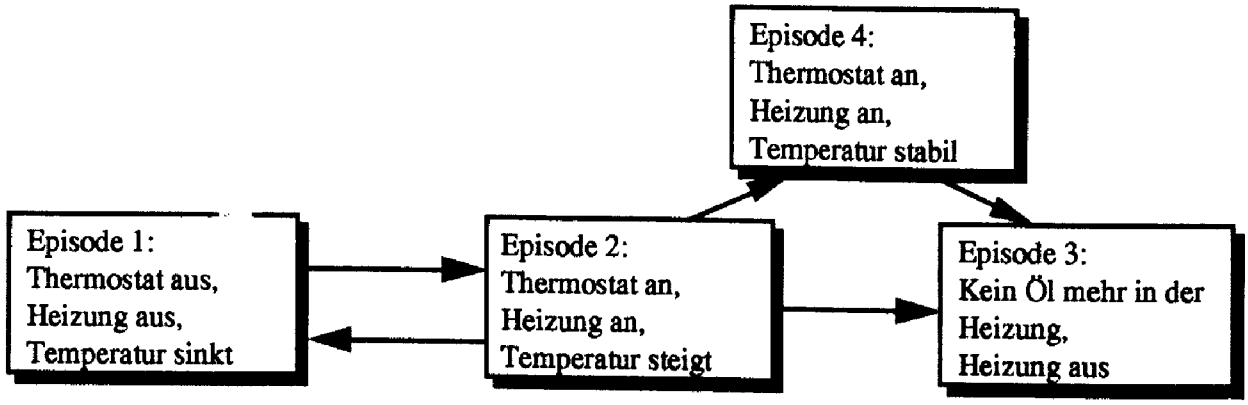

Abbildung 7.15: Ergebnis einer qualitativen Simulation eines Thermostat-Heizungssystems; nach [Charniak und McDermott, 1985]

Der Übergang zwischen Intra- und Interstate-Analyse wird durch die globale Analyse gesteuert, die für das Erkennen von Schleifen, Gleichgewichtszuständen und Vereinigungen von Verzweigungen zuständig ist. Ein Beispiel für das Ergebnis einer qualitativen Simulation zeigt Abb. 7.15.

Bekannte Ansätze zur qualitativen Mehrphasensimulation sind QSIM [Kuipers, 1984], ENVISION [de Kleer, 1984] und QPT [Forbus, 1984]. QPT (Qualitative Process Theory) basiert auf einer anderen Wissensrepräsentation, bei der nicht Komponenten und ihre Parameter wie die Temperatur der Wärmequelle, sondern Prozesse wie der Wärmefluß die Hauptobjekte in der Strukturbeschreibung sind. Eine vergleichende Darstellung der Ansätze findet sich in [Bredeweg und Wielinga, 1988] und [Struß, 1989]. Allerdings führt die Vernachlässigung der Größenordnungen der relevanten Parameter leicht zu einer kombinatorischen Explosion der Fallunterscheidungen. Ein System, das die Größenordnung qualitativer Zusammenhänge relativ detailliert modelliert, ist das qualitative Einphasensimulations-System in [Long *et al.*, 1986; 1988].

Eine ausführliche Darstellung der qualitativen Simulation findet sich in [Iwasaki, 1989]. Es darf jedoch nicht übersehen werden, daß die derzeitigen Systeme zumeist noch sehr einfache Fragestellungen behandeln und kaum praxisreif sind.

7.1.5 Entwicklung von Expertensystemen

7.1.5.1 Methodiken der Expertensystementwicklung

In diesem Abschnitt sollen Expertensysteme vom Standpunkt der Software-Entwicklung betrachtet werden. Ein Expertensystem ist ein so kompliziertes und umfangreiches Gebilde, daß es halsbrecherisch wäre, sich blindlings in seine Entwicklung zu stürzen. Bei der Suche nach einer Entwicklungsmethodik liegt es vielleicht zunächst nahe, diejenige des klassischen Software-Engineering („Wasserfallmodell") zu verwenden. Dies schlägt jedoch fehl, denn das klassische Vorgehen beruht entscheidend auf der Existenz einer präzisen Spezifikation[4], die es bei den „diffusen", „schlecht definierten" Anwendungsbereichen (vgl. Abschnitt 7.1.1.3) von Expertensystemen gerade *nicht* geben kann. Daher ist man weitgehend auf prototypisches, explorativ es Vorgehen verwiesen. Wir wollen in diesem Abschnitt aufzeigen, welche Aspekte bei der komplexen Aufgabe, ein Expertensystem zu erstellen, berücksichtigt werden müssen und welche Strukturierungsansätze diskutiert werden.

Die meisten vorgeschlagenen Entwicklungsmethodiken bleiben dem klassischen Modell insoweit verhaftet, als sie eine Einteilung in zeitlich aufeinanderfolgende, a priori bekannte, allerdings gröbere Phasen vorsehen. Während die Methodiken sich im Umfang der insgesamt berücksichtigten Tätigkeiten und in Feinheit und Abgrenzung der einzelnen Phasen unterscheiden, sind sie sich in der Abfolge der zu behandelnden Aufgaben recht ähnlich (vgl. [Karbach und Linster, 1990, Kap. 4, S. 38]). Ein Unterschied zu klassischen Phasenmodellen liegt im häufigen Fehlen präzise definierter Dokumente als Produkte der jeweiligen Phasen. Die folgende Darstellung konzentriert sich zunächst auf gemeinsame Aspekte zweier häufig genannter Vorgehensweisen, des Rapid-Prototyping- und des modellbasierten Ansatzes. Auf Besonderheiten dieser unterschiedlichen Ansätze gehen wir anschließend genauer ein.

Problemidentifikation. Bevor man ein System in Angriff nimmt, muß man die Ziele des Systems definieren, d.h. die zu lösenden Probleme abgrenzen und beschreiben. Die Unterprobleme, die zur Verfügung stehenden Daten und die vorgesehenen Benutzergruppen sind zu bestimmen; es ist festzulegen, was eine Lösung ausmacht (reicht etwa eine Diagnose oder werden auch Therapievorschläge gebraucht?). Die Beschreibung der Anforderungen muß auf jeden Fall Kompetenzgrenzen und Verantwortlichkeiten für eventuelle Fehler festlegen. Von vornherein sollten auch die zukünftigen Benutzer in die Planung des Systems einbezogen werden (partizipative Entwicklung), um nicht am Bedarf vorbeizuentwickeln.

Prüfung der Angemessenheit und Realisierbarkeit. Hinsichtlich der *Angemessenheit* eines Expertensystems ist zunächst der praktische Nutzen zu klären, den ein System zur Lösung der Probleme überhaupt erwarten läßt: Wie häufig treten Probleme auf? Wie groß ist die Fehlerwahrscheinlichkeit bei der Problemlösung durch Laien? Wie stark wirken sich Fehlentscheidungen aus? Ist das Einsatzszenario günstig? Für die weitere Einschätzung

[4] auch, wenn diese Spezifikation ggf. mit Hilfe eines Prototypen gewonnen wird

braucht man eine erste Vertrautheit mit der Problemdomäne sowie Kenntnis der Einsatzbedingungen aus unmittelbarer Anschauung. Ein Expertensystem ist nur dann angemessen, wenn es für das Problem keine konventionelle algorithmische Lösung gibt. Die Komplexität des Problembereichs muß groß genug sein, um den Aufwand für ein solches System zu rechtfertigen, andererseits sollte ein Experte zum Lösen eines durchschnittlichen Problems nicht länger als wenige Stunden brauchen.

Expertensysteme haben bisher recht enge Grenzen hinsichtlich dessen, was machbar ist; man muß deshalb die *Realisierbarkeit* ernsthaft prüfen. Dabei ist bereits zu bedenken, daß die Dateneingabe nicht zu aufwendig sein darf: die Übermittlung der Problemdaten sollte zwischen Menschen in vertretbarer Zeit per Telefon, insbesondere verbal, möglich sein. Es ist günstig, wenn viele Daten automatisch erfaßt werden. Zur Zeit realistische Systeme können nur eingeschränkt mit natürlicher Sprache umgehen. Auch Alltagswissen darf nur in geringem Umfang eingehen; mit seinen speziellen Aspekten Raum und Kausalität kann man bisher kaum, mit Zeit nur unter Schwierigkeiten umgehen. Es muß gesichert sein, daß das System nur mit Problemen seines Kompetenzbereichs konfrontiert wird. Unabdingbar ist schließlich, daß ein motivierter Experte zeitlich ausreichend für die Wissensakquisition zur Verfügung steht. Ausführlichere Checklisten findet man in [Puppe, 1991, Kap. 17] und [Mertens *et al.*, 1990, Kap. 5]

Wissenserhebung und Wissensanalyse. Hat die Problemidentifikation ein Expertensystem als angemessen und realisierbar erwiesen, muß Wissen in größerem Umfang erhoben und (zumindest für Beispiele) analysiert werden. Diese Entwicklungsphase wird von Buchanan et al. in [Hayes-Roth *et al.*, 1983] *Konzeptualisierung* genannt; sie erstreckt sich auf benötigte Konzepte und Relationen, Problemlösungsprozesse, Kontrollmechanismen, Strategien und Teilaufgaben. Methoden für die Wissenserhebung sind in Teil 7.1.3 bereits angesprochen worden. In dem Ausmaß von Erhebung und Analyse vor Beginn der Implementation liegt ein entscheidender Unterschied zwischen dem modellbasierten und dem Rapid-Prototyping-Ansatz, auf die wir in Abschnitt 7.1.5.2 zu sprechen kommen.

Implementierung. Vor der Implementierung eines Expertensystems ist festzustellen, ob es geeignete Entwicklungswerkzeuge gibt und welches davon man verwenden will. Soweit möglich sollte eine problemspezifische Shell gewählt werden. Beim Rapid Prototyping geschieht diese Auswahl gleich nach der Konzeptualisierung, beim modellbasierten Ansatz erst, wenn ein ausgearbeitetes Wissensmodell vorliegt. Zu klären ist, ob sich das Wissen überhaupt in den angebotenen Formalismen darstellen läßt, ob die Inferenzmöglichkeiten relevante Aspekte der Domäne wie nicht-monotones Schließen und spezielle Relationen (z.B. *ist-Teil-von*) unterstützen und schließlich, welche Hilfe für die Wissensakquisition geboten wird. Gängige Werkzeuge besprechen wir im Abschnitt 7.1.5.4.

Im Unterschied zu klassischen Systemen kann bei Expertensystemen der zentrale Teil, die Wissensbasis, in kleinen Schritten inkrementell implementiert werden. Dabei sollten immer wieder die Anmerkungen des beteiligten Experten und auch der zukünftigen Benutzer eingeholt und durch Korrigieren und Modifizieren berücksichtigt werden. Beim Ra-

pid Prototyping wird in der Implementierungsphase der Hauptteil des Wissens akquiriert; Wissenselemente, die ausreichend präzise bekannt sind, werden sofort in den Repräsentationsformalismus überführt.

Wenn sich auch die Wissensbasisentwicklung klassischen Vorgehensweisen des Software-Entwurfs entzieht, sollte man doch stets versuchen, Teile des Gesamtsystems zu identifizieren, die sich klassisch entwickeln lassen (wie häufig die Benutzeroberfläche), und hier auch entsprechende Verfahren verwenden [Partridge, 1989].

Systemtest. Festzustellen, ob das System korrekt und anwendungsreif ist, ist bei Expertensystemen sehr schwierig. Manchmal lassen sich Aspekte der Konsistenz von Wissensbasen (z.B. Widerspruchsfreiheit) formal testen. Informell kann man für ein Diagnosesystem beispielsweise prüfen, ob sämtliche möglichen Diagnosen auch als Ergebnisse auftauchen können. Es ist ratsam, Testfälle so bereitzuhalten, daß man bei Änderungen des Systems leicht prüfen kann, ob sie noch genauso (gut) wie früher gelöst werden („Regressionstest"). Eine Verifikation im klassischen Sinne kann es mangels präziser Spezifikation nicht geben.

Einsatz und Wartung. Die notwendige Integration in die Einsatzumgebung erfordert häufig Schnittstellen des Systems zu anderer und andersartiger Soft- und Hardware. Wie immer müssen vor dem Einsatz die Benutzer auf die Einführung des Systems vorbereitet und im Umgang mit ihm geschult werden.

Die Wartung, vor allem die in der Regel unvermeidbare laufende Aktualisierung des in Expertensystemen verwendeten Wissens, ist besonders wichtig und aufwendig; etliche Projekte sind schließlich hieran gescheitert [Mertens *et al.*, 1990]. Beim Rapid Prototyping wird die Wartung dadurch erschwert, daß die Dokumentation notgedrungen unvollständig ist, besonders was die abstrakte Ebene (wie Entwurf, Problemlösungsstrategie) angeht. Um hier den Überblick zu behalten, ist es von zentraler Wichtigkeit, übersichtliche Darstellungen der Wissensbasis bekommen zu können. Das Wartungsproblem wird durch die deklarative Darstellung des Wissens erleichtert, weil so auch entsprechend geschulte Nichtprogrammierer (erfahrene Benutzer) Wartungstätigkeiten übernehmen können. Der Personalaufwand, der pro Jahr für die Wartung eines Experstensystems erforderlich ist, wird in [Schild, 1987] auf ein Drittel des Potentials geschätzt, das für Entwicklungsarbeiten benötigt wird.

Evaluation. Hierunter versteht man die Bewertung des eingesetzten Systems hinsichtlich seiner Leistung und Nützlichkeit. Zur Bewertung der *Systemleistung* wird untersucht, ob das Wissen korrekt, konsistent und vollständig ist, ob die Kontrollstrategie einen natürlichen Ablauf erbringt und ob die Erklärungen des Systems adäquat sind. Der wichtigste Punkt bei der Leistungsbewertung, ob nämlich die Problemlösungen angemessen sind, ist ähnlich schwierig wie die Beurteilung eines Experten selbst, denn es gibt *keine objektiv richtigen Lösungen*: verschiedene Experten können unterschiedlicher Ansicht sein. So muß man sich damit begnügen, daß das System nach dem Urteil eines Expertengremiums keine groben Fehler macht und hinsichtlich der Lösungen dem Vergleich mit natürlichen Experten standhält. Es bleibt ein *Risiko*: Bei Menschen, die eine Reihe komplizierter Beispielfälle

richtig lösen, kann man darauf vertrauen, daß sie *generell* keine groben Fehler machen werden – bei einem technischen System läßt sich das nicht ausschließen.

Die *Systemnützlichkeit* wird u.a. danach beurteilt, ob die Lösungen Benutzern signifikant helfen, ob die Ergebnisse angemessen organisiert und detailliert sind und ob eine bedienungsfreundliche Benutzerschnittstelle einen akzeptablen Umgang mit dem System erlaubt. Wegen der häufig erforderlichen Wartung ist schließlich die *Änderungsfreundlichkeit* des System von entscheidender Bedeutung.

7.1.5.2 Rapid Prototyping vs. modellbasierter Ansatz

Die ersten Expertensysteme entstanden nach dem *Rapid-Prototyping*-Verfahren. Auch heute werden viele Systeme nach diesem Verfahren erstellt, allerdings spielen Werkzeuge und Shells dabei eine größere Rolle als früher. Wie bereits in Teil 7.1.3 angedeutet, versucht man beim Rapid Prototyping, möglichst frühzeitig eine Minimalversion des gewünschten Systems zu erstellen. Dies bedeutet konkret, daß man ein kleines System mit etwa 25 bis 50 Einträgen in der Wissensbasis erzeugt, sobald das erste Beispiel gut verstanden ist – auch wenn die Regeln noch nicht perfekt sind [Hayes-Roth *et al.*, 1983]. Mit dem Prototypen soll vor allem überprüft werden, ob die gewählten Formalismen problemadäquat sind; Effizienzgesichtspunkte werden zunächst vernachlässigt. Aus den Erfahrungen mit der Vorversion wird der Entwurf für ein umfassenderes System abgeleitet. Teilweise wird empfohlen, danach den Prototypen in jedem Fall aufzugeben und das geplante System von Grund auf neu zu entwickeln.

Als Vorteil von Rapid-Prototyping-Ansätzen gilt, daß eine Fokussierung bei der Wissenserhebung und -interpretation sowie unmittelbare Überprüfung der erfolgten Wissensmodellierung (Fakten, Inferenzstrategie, Systemverhalten) möglich wird; es ergeben sich kürzere Feedback-Zyklen mit dem Experten. Anforderungen an Werkzeug und Benutzerschnittstelle können besser präzisiert, die Verwendbarkeit bekannter Ansätze und Algorithmen frühzeitig geprüft werden. [Partridge, 1989] vertritt sogar die Ansicht, Entwicklung nach dem Run-Debug-Edit-Verfahren sei für KI-Systeme unumgänglich.

Als Nachteil des Rapid Prototyping wird gesehen, daß die Repräsentation der Expertise tendenziell durch Implementierungsformalismen bestimmt wird. Daß die Systemarchitektur vor der abschließenden Analyse der Wissensprotokolle festgelegt werden muß, beengt die Sicht auf die Expertise; spätere Änderungen sind teuer. Die Planung ist unübersichtlich: selbst während der Implementierung ist nie recht klar, wieviel noch zu tun ist (z.B. akquiriert werden muß). Es ist ferner ungewiß, ob die Struktur des kleinen Systems tatsächlich im Großen standhält (*„scaling up"*). Vor allen Dingen wird aber keine explizite Entscheidung über das Wissensmodell getroffen; es *ergibt* sich im Verlauf der Entwicklung.

Im *modellbasierten Ansatz* wird dagegen versucht, Modelle des Problemlöseprozesses explizit zu machen: das bedeutet die Offenlegung des mentalen konzeptuellen Modells, das sich der Knowledge Engineer von der beobachteten Expertise macht, *unabhängig* von ir-

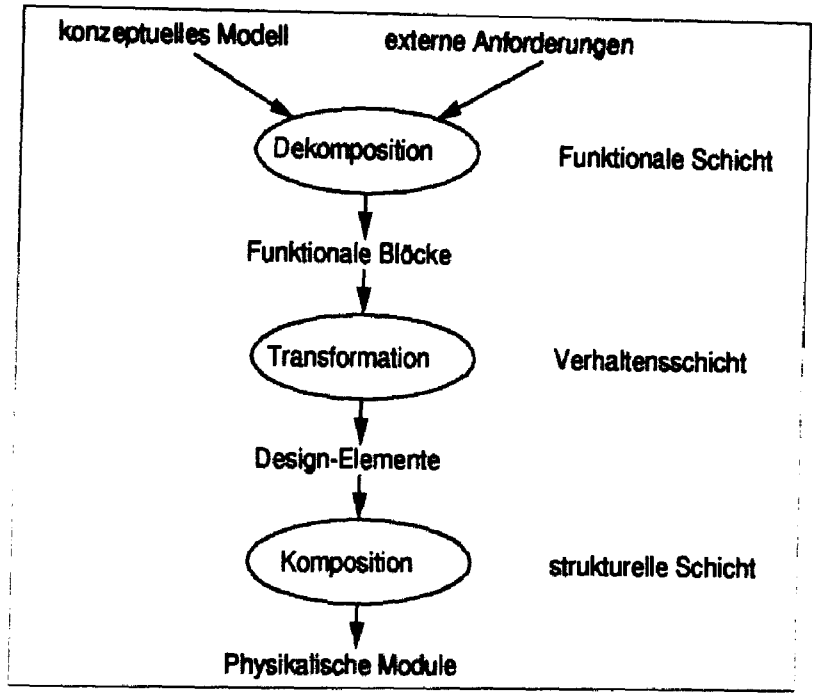

Abbildung 7.16: Erarbeiten der Schichten des Design-Modells, vgl. auch Abb. 7.5 und 7.6

gendwelchen Schritten zur Implementierung. Die Vorteile des modellbasierten Entwurfs bestehen vor allem in der klaren Trennung von Analyse und Implementierung, die evtl. sogar von verschiedenen Personen durchführbar sind. Nach der Analysephase liegt eine relativ vollständige Beschreibung der Expertise auf einer implementierungsunabhängigen Darstellungsebene vor. Der Aufbau des mentalen Modells wird transparent und dokumentierbar, und sog. generische Modelle von Teilaufgabenstrukturen können die Interpretation und die Wissensakquisition leiten. Fehler der Interpretation resultieren nicht in unnötiger Implementierungsarbeit; der Änderungsaufwand wird geringer. Dafür entfallen leider die Vorteile des Rapid Prototyping; insbesondere kann die Dynamik des Modells nicht direkt überprüft werden.

Der wohl bekannteste modellbasierte Entwicklungsansatz ist KADS. Das dort festgelegte Vorgehen im Rahmen der Wissensakquisition und bei der Erstellung des konzeptuellen Modells ist in Abschnitt 7.1.3.5 behandelt worden. Wichtig für die Entwicklungsmethodik ist die Tatsache, daß hier mehrere Zwischenschritte zwischen dem Erwerb von Expertenwissen und der Implementation eingelegt werden, was die zu leistende Aufgabe in Abschnitte geringerer Komplexität aufteilt. Die Bedeutung des ersten Abschnitts, der Erstellung des *konzeptuellen Modells*, liegt darin, daß das auf der Inferenzebene, Aufgabenebene und Strategieebene formulierte *Interpretationsmodell* den Knowledge Engineer bei der Wissensakquisition und auch bei der Auswahl von Werkzeugen für die Systementwicklung leiten kann.

Auf dem konzeptuellen Modell baut das sogenannte *Design-Modell* auf. Geht es bei dem

ersten um eine kognitiv angemessene Beschreibung, zielt das zweite auf einen Realisierungsentwurf. Nach der KADS-Methodologie werden hier – unter Berücksichtigung der formulierten Anforderungen an das zu erstellende System – nacheinander drei Schichten erarbeitet (vgl. Abb. 7.16). Für die *funktionale Schicht* wird eine Zerlegung in funktionale Blöcke wie Suchen, Ein-/Ausgabe vorgenommen. In der *Verhaltensschicht* werden abstrakte Programmbausteine wie etwa Tiefensuche zur Verwirklichung dieser Blöcke ausgewählt. Diese *Design-Elemente* werden in der *strukturellen Schicht* zu physikalischen Software-Moduln kombiniert. Diese Schicht bildet die Basis für detaillierten Entwurf und Implementierung. Die schriftliche Dokumentation des konzeptuellen und des Design-Modells erleichtert die spätere Wartung. Im Idealfall erhält das Design-Modell die Struktur des konzeptuellen Modells; dies spielt dann die Rolle einer abstrakten Spezifikation. Bislang bietet KADS zwar die Leitgedanken für einen solchen Entwicklungsprozeß, aber noch keine ausformulierten Methoden; an der Operationalisierung wird jedoch gegenwärtig – wie bereits angesprochen – in großem Umfang gearbeitet. Orientierte sich das ursprüngliche Lebenszyklusmodell von KADS [Barthélemy *et al.*, 1987] am Wasserfallmodell, lehnt sich ein mittlerweile entwickeltes [Taylor *et al.*, 1989] an das Spiralmodell [Boehm, 1988] an.

Die Design-Elemente haben somit eine ähnliche Funktion wie die *generic tasks* von [Chandrasekaran, 1986]. Diesem Ansatz liegt die Idee eines Baukastensystems kleiner Problemlösungswerkzeuge zugrunde, welche wiederkehrenden Teilaufgaben, die in vielen größeren Systemen verwendbar sind, Rechnung tragen. Dieser Baukasten enthält mittlerweile u.a. Teilsysteme zur hierarchischen und probabilistischen Klassifikation, zum Skelett-Planen und zur Vorhersage der Auswirkungen von Zustandsänderungen. Gegenwärtig hat eine Diskussion darüber begonnen, solchen grundsätzlichen Vorgehensweisen vor allem aus Gründen der Wiederverwendbarkeit von Software für Expertensysteme zentrale Aufmerksamkeit zu widmen [AI-magazine, 1991].

7.1.5.3 Komponenten von Expertensystemen

Eine idealtypische *Architektur* von Expertensystemen zeigt Abb. 7.17. Die zentralen Teile, die *Wissensbasis* und die *Problemlösungskomponente*, sind bereits in Abschnitt 7.1.1.3 diskutiert worden. Die Interaktion mit Endbenutzern bei anwendungsreifen Systemen sowie auch in der Entwicklung und Wartung des Systems mit Knowledge Engineer bzw. Experten übernimmt die Benutzerschnittstelle, bestehend aus Interviewerkomponente, Erklärungskomponente und Wissenserwerbskomponente. Die *Interviewerkomponente* wird vom Problemlöser aktiviert, um fallspezifisches Wissen vom Endbenutzer zu erfragen, das in die dynamische Wissensbasis eingetragen wird. Im folgenden gehen wir kurz auf die Rolle der anderen genannten Komponenten ein.

Die *Erklärungskomponente* stellt für den Endbenutzer das Lösungsergebnis und den Lösungsweg dar und begründet evtl. die gefundene Lösung. Hier ist besonders darauf hinzuweisen, daß sinnvolle Erklärungen nur dann möglich sind, wenn man bei der Syste-

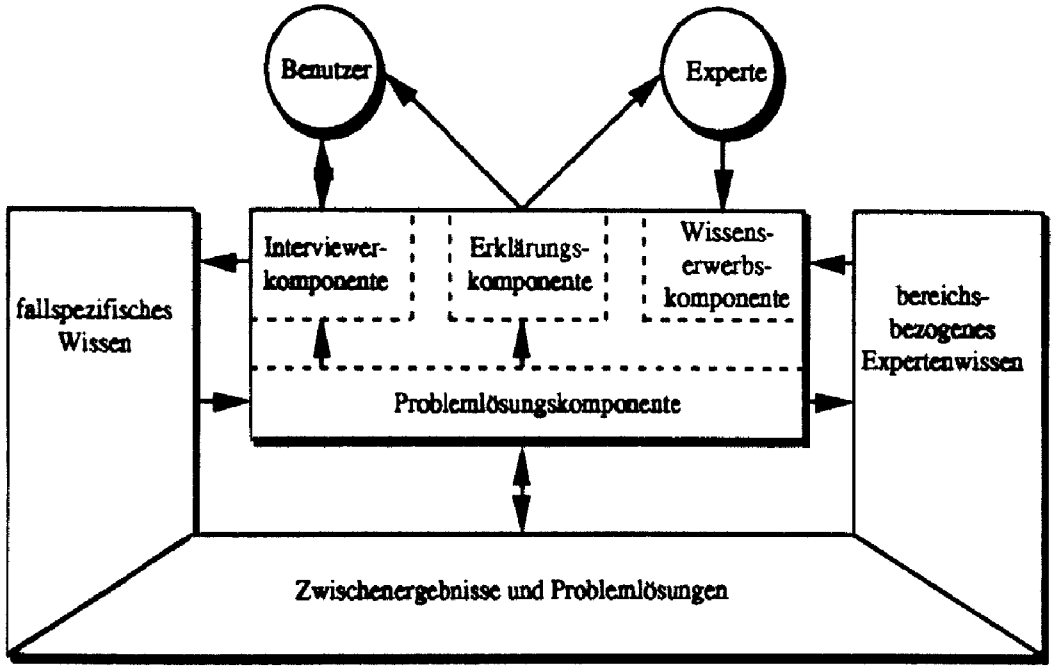

Abbildung 7.17: Architektur von Expertensystemen. Quelle: [Puppe, 1991]

merstellung von vornherein an ihre Erzeugung gedacht, insbesondere entsprechendes Wissen akquiriert hat. Käufliche Expertensystemwerkzeuge bieten hier meistens wenig Unterstützung. Oft werden die Funktionen der Erklärungskomponente auch beim Aufbau des Expertenwissens zur Begutachtung von Testfällen gebraucht. Der häufig angebotene Trace des Inferenzprozesses ist hier als ein Debugging-Werkzeug für den Systementwickler nützlich; den Bedürfnissen des Endbenutzers wird er nicht gerecht.

Die *Wissenserwerbskomponente* dient dem Experten bzw. dem Wissensingenieur dazu, das Expertenwissen in die (statische) Wissensbasis einzubringen. Für den Wissensingenieur kann sie weit mächtiger ausgelegt werden als für den Experten, der den Repräsentationsformalismus und die Abhängigkeiten zwischen Teilen der Wissensbasis weniger durchschauen wird. Soweit möglich sollte die Wissenserwerbskomponente die Prüfung der Wissensbasis auf Vollständigkeit und Konsistenz unterstützen.

Die *Benutzerschnittstelle* insgesamt muß wegen der großen Komplexität der zu lösenden Aufgaben sehr komfortabel sein; sie muß den Benutzer einerseits ausreichend führen, damit er sich nicht verirrt und nichts vergißt, ihm aber andererseits genügend Freiheit lassen, alles relevante Wissen in der für ihn natürlichen Reihenfolge einzugeben und irrtümliche Eingaben nachträglich zu korrigieren. Eine moderne Dialogtechnik (Masken, Menüs, Grafik) ist nötig, die es erlaubt, mehrere Fragen im Zusammenhang zu beantworten. Wichtig ist dabei eine ausreichende Reaktionsgeschwindigkeit des Systems und ein großer Bildschirm, auf dem mehrere Fenster gleichzeitig sichtbar gehalten werden können.

7.1.5.4 Expertensystemwerkzeuge

Da bei gegenwärtigen Expertensystemprojekten in der Regel nicht mehr „from the scratch", sondern mit Unterstützung von Expertensystemwerkzeugen gearbeitet wird, wollen wir im folgenden die Rolle solcher Werkzeuge diskutieren. Im Anschluß stellen wir dann einige hybride Expertensystemwerkzeuge vergleichend gegenüber.

Arten von Werkzeugen. Für die Entwicklung eines Expertensystems kann man auf drei verschiedenen Ebenen aufsetzen: (KI-)Programmiersprache, Expertensystemwerkzeug, Expertensystemshell.

Die ersten Expertensysteme entstanden in der Regel auf der Grundlage einer KI-Sprache wie Lisp. In Systemen, die jeweils auf die spezifische Anwendung zugeschnitten waren, wurden zunächst oft vermischte Wissens- und Kontrollstrukturen geschaffen. Später trennte man Wissensbasis und Inferenzstrategie; so entstanden Werkzeuge, die man für eine ganze Klasse von Anwendungen benutzen konnte (z.B. EMYCIN). Diese Art von Werkzeugen wird üblicherweise als *Shell* bezeichnet: Die Art des dargestellten Wissens (z.B. Symptome, Diagnosen, Beziehungen zwischen beiden) und seine Verarbeitung (z.B. hypothesize-and-test) liegen fest; das konkrete Wissen über eine Domäne (z.B. Lungenkrankheiten) muß eingebracht werden. Einige Shells (z.B. SYNTEL und Financial Advisor) enthalten zusätzlich Basiswissen des Anwendungsbereichs („Shells mit Basiswissen"). Eine andere Entwicklung führte zu den *Expertensystemwerkzeugen* im engeren Sinne: hier werden Repräsentationsformalismen (z.B. Regeln, Frames), elementare Inferenzverfahren (z.B. Vererbung, Vorwärtsverkettung) und Oberflächen für Zugriff auf Objekte und Kontrolle der Inferenz zur Verfügung gestellt.

Hybride Werkzeuge bieten mehrere Wissensrepräsentationsformalismen und darauf abgestimmte elementare Inferenzmechanismen an. Fast alle stellen außerdem eine fensterorientierte, grafikfähige Oberfläche zur Verfügung, die vom Endbenutzer und vom Wissensingenieur genutzt werden kann. Wichtigste Merkmale sind die Mächtigkeit der Repräsentationsformalismen und die Integration der verschiedenen Komponenten untereinander. Wenn nicht deutliche Unterschiede in der Mächtigkeit der Repräsentationsformalismen vorliegen, dann scheint ein aussagekräftiger Vergleich von Werkzeugen ohne die Möglichkeit, sie mit Bezug auf eine vorgesehene Aufgabe auszuprobieren, sehr schwierig. Die Qualität der Oberfläche (Klarheit, Robustheit, Geschwindigkeit) hat entscheidenden Einfluß auf die Effizienz des Werkzeuggebrauchs. Während früher Expertensystemwerkzeuge fast immer auf Lisp oder Prolog basierten, geht mittlerweile in industriellen Bereichen ein deutlicher Trend zu einer C-Basierung. Einer der Gründe dafür ist die leichtere Integration in klassische Umgebungen.

Bekannte Expertensystemwerkzeuge. Die nachfolgende kurze Charakterisierung einiger bekannter Werkzeuge basiert zum großen Teil auf den Erfahrungen der Autoren. Zur Zeit der Buchentstehung (1992) hatten die Autoren direkten Zugriff auf die aktuellsten Versionen von KEE, Knowledge Craft, Babylon, ROCK, ProKappa, Kappa-PC und

Nexpert Object. Hinsichtlich Hardware-Umgebung und soweit nichts speziell anderes bemerkt, kann man sagen, daß die grafikorientierten Entwicklungsumgebungen vorwiegend für Unix-basierte Workstations, die eine hohe Rechenleistung und genügend Hauptspeicher und Swapkapazität pro Arbeitsplatz zur Verfügung stellen können, konzipiert sind und eine solche Umgebung auch erfordern. Bei allen Werkzeugen muß man die zugrundeliegende Programmiersprache beherrschen, wenn man objektorientiert programmieren möchte.

Zu den ersten hybriden, leistungsfähigen Werkzeugen gehörten ART („Automated Reasoning Tool"), KEE („Knowledge Engineering Environment") und *Knowledge Craft*. Alle drei Werkzeuge bauen auf Lisp auf und bieten u.a. Regeln mit beiden Verkettungsrichtungen, Frames, objektorientierte Programmierung, Formen der Abhängigkeitsverwaltung (Truth maintenance) und fensterbasierte Grafikoberflächen.

Die Wissensdarstellung in ART basiert in erster Linie auf den vier Hauptkomponenten Fakten, Schemata, Regeln und Kontexte (viewpoints). Deklaratives Wissen kann als Faktum, Schema oder Kontext, prozedurales Wissen in Regelform repräsentiert werden. Dabei unterscheidet ART zwei Typen von Regeln, die 'if-then' Regeln und die 'while-then' Regeln. Von beiden Teilen der Regel können selbstdefinierte Lisp-Funktionen aufgerufen werden. Die Kontexte in ART bieten die Möglichkeit zum hypothetischen Schlußfolgern durch den Vergleich verschiedener Szenarien, wobei jeder Kontext ein separates Szenario darstellt.

Im Gegensatz zu ART bietet KEE eine objektorientierte Programmierumgebung. Das grundlegende Konzept von KEE ist das Framekonzept („Units"). Alle Bestandteile eines wissensbasierten Systems werden in KEE als Units realisiert. Dies gilt sowohl für Objekte und Objektstrukturen als auch für Regeln, Windows, Graphiken, usw. In KEE können Lisp-Funktionen, TellAndAsk und Dämonen integriert werden. Bei der Entwickleroberfläche wurde in KEE eine Erweiterung der Standard-Fenster-Systeme um objektorientierte Elemente vorgenommen. Zum Teil sind dies KEE-spezifische Erweiterungen von Fenstersystemen um Standardbegriffe und Standardfenster (Common-Windows-Paket). Auf den Funktionen von Common Windows ist ein weiteres Paket aufgesetzt, das neben speziellen Fenstern auch entsprechende Methoden zur Verwaltung dieser Strukturen enthält. Die Flexibilität und die Mächtigkeit der Entwickleroberfläche erfordert Disziplin, um die Übersicht zu wahren; in der aktuellen Version für SUN-4 ist die Einbindung des Editors unzureichend. In KEE ist ein Vergleich multipler hypothetischer Situationen schwieriger zu repräsentieren als in ART.

Knowledge Craft zeichnet sich durch ein mächtiges Relationen-Konzept aus. Neben dynamischen Vererbungs-Mechanismen bietet Knowledge Craft u.a. Meta-Informationen, Dämonen, entwicklerdefinierte Relationen und Abhängigkeitsbeziehungen. Die Integration der verschiedenen Komponenten ist in Knowledge Craft relativ mühsam, z.B. wäre bei den angebotenen Realisierungen von OPS und Prolog eine komfortablere Integration mit den Frames wünschenswert. Jedoch sehen es einige Entwickler als Vorteil, nach Erstellung eines Prototyps direkt auf Prolog oder OPS weiterentwickeln zu können. Im Gegensatz zu KEE bietet Knowledge Craft keine so komplexe und komfortable Entwicklerschnittstelle.

Man vermißt bei diesen drei Werkzeugen die Unterstützung von Regressionstests; Schnittstellen zu Datenbanken sind nicht vorhanden oder müssen zusätzlich gekauft werden.

Mittlerweile gibt es zu jedem der drei Werkzeuge C-basierte Pendants mit mehr oder minder eingeschränkter Funktionalität: ART-IM, *Pro-Kappa* und ROCK („Representation of Corporate Knowledge"). ROCK zeichnet sich durch mächtige Framestrukturen mit Vererbungsstrategien, Metawissen (z.B. kann an einen Slot ein Metaobjekt angehängt werden), objektorientierte Programmierung, offene Schnittstellen zu X-Windows und OSF-Motif aus. Der Funktionalitätsumfang ist z.Zt. bei ROCK stark eingeschränkt, da Regelinterpreter und Grafikoberfläche noch selbst geschaffen werden müssen. Alle drei Werkzeuge bieten Datenbankschnittstellen.

Ähnlich mächtig ist das ebenfalls Lisp-basierte *Babylon*. In diesem Tool kann Wissen außer in objektorientierter und regelbasierter Form auch durch Abhängigkeitsnetze (Constraints) repräsentiert werden. Die Regelabarbeitung kann durch die Strukturierung der Regeln in Regelmengen und durch die Vergabe von Prioritäten gesteuert werden. Multiple Vererbung ist möglich; jedoch können die Vererbungsstrategien vom Entwickler wie auch bei KEE nicht selbst definiert werden. Die Möglichkeit der Abhängigkeitsverwaltung ist z.Zt. in Babylon nicht realisiert.

Twaice ist in Prolog und C implementiert und läuft außer auf Unix-Rechnern auch auf PCs unter DOS. Auch hier gibt es jetzt über die (automatisch auf Konsistenz prüfbaren) Regeln hinaus Frames mit Dämonen, Formen von Truth-Maintenance, Konfidenzverwaltung, Schnittstellen zu Datenbanken und Integration von Grafik. Twaice läuft jedoch auch auf nicht grafikfähigen Arbeitsplätzen. Objektorientierte Programmierung ist dagegen kaum ausgeprägt. Außergewöhnlich sind die Erzeugung von Erklärungen in natürlicher Sprache, eine Induktionskomponente für das automatische Lernen von Regeln aus Daten und die Verbindung mit einem Hypertextsystem.

Zu den etwas einfacheren Werkzeuge gehören GoldWorks und Nexpert Object. Beide stehen auch auf PC und auf Macintosh zur Verfügung. *GoldWorks* ist Lisp-basiert und umfaßt über die bei den „großen" Werkzeugen aufgezählten Möglichkeiten hinaus eine Konfidenzverwaltung. *Nexpert Object* stellt eine Record-and-Replay-Funktion für Regressionstests und eine vordefinierte *ist-Teil-von*-Relation zur Verfügung; in die Regelverkettung läßt sich weitgehend eingreifen. Ein einfacherer KEE-Abkömmling für den PC ist *Kappa-PC*; es ist ebenso wie *Nexpert Object* C-basiert. Beide Werkzeuge zeichnen sich durch Geschwindigkeit und klare Oberfläche aus.

Häufig wird mittlerweile die Erzeugung einer Laufzeitversion für den Endbenutzer angeboten, die geringere Anforderungen an die Umgebung stellt. Ausführliche Vergleiche von Expertensystemwerkzeugen findet man in [von Bechtolsheim *et al.*, 1991]. Mit einzelnen Systemen beschäftigen sich [Wolf und Setzer, 1991], [Christaller *et al.*, 1989] und [Mellis, 1990].

Die hauptsächliche Unterstützung, die solche hybriden Expertensystemwerkzeuge leisten, liegt darin, dem Knowledge Engineer komfortable Ausdrucksmöglichkeiten für Wissen

und gleichzeitig elementare Inferenzmechanismen darüber zu bieten. Beides beschleunigt die Entwicklung erheblich gegenüber der Verwendung von Programmiersprachen. Allerdings lassen die aktuellen Werkzeuge noch etliche Wünsche offen, auf der technischen Seite z.B. bzgl. Schnelligkeit und Robustheit, Integrierbarkeit in klassische Umgebungen, Testunterstützung; auf der Seite der Ausdrucksfähigkeit Darstellung zeitlicher und räumlicher Bezüge, Umgang mit Ungewißheit, fallbasiertes Schließen, Generierung sinnvoller Erklärungen. Dabei ist allerdings zu bedenken, daß die damit verbundenen Probleme auch wissenschaftlich noch nicht ausreichend gelöst sind. Hinsichtlich Schnelligkeit und Integrierbarkeit gibt es deutliche Fortschritte; es zeichnet sich eine Entwicklung ab zu verteilten, Mehrbenutzer-fähigen Systemen und zur Einbindung grafischen Materials wie Bildern, Fotos, Animationen.

Expertensystemshells. Expertensystemshells sind problemtypspezifische Werkzeuge, die eine bereits implementierte Problemlösungsstrategie für einen speziellen Problemlösungstyp zur Verfügung stellen, dem Verwender aber den eigenständigen Aufbau einer dazu passenden Wissensbasis mit bereichsspezifischem Problemlösewissen überlassen. Bekannte Diagnoseshells sind D3 (Puppe), TEST bzw. TESTBENCH (Carnegie Group), ESS (Coherent Thought); ein Konfigurierungssystem ist PLAKON [Cunis *et al.*, 1991], s. auch S. 786. Wir erläutern beispielhaft D3 [Bamberger *et al.*, 1991], das nach dem Prinzip der heuristischen Klassifikation arbeitet. Neben der Symptom- und Diagnosehierarchie ist Wissen über Symptome einer Diagnose, über die Prädispositionen von Diagnosen und über probabilistische Korrelationen zwischen Symptomen und Hypothesen erforderlich. Es wird durch Auswahl aus angebotenen Menüs und durch Eintrag in Tabellen direkt vom Experten eingegeben. Wie bei allen Shells muß der Experte dazu vorher in die Grundstrukturen des Werkzeugs, die Arten benötigten Wissens und Kriterien für sinnvolle Hierarchisierung eingeführt werden, was jedoch durch die grafische Entwicklungsoberfläche vereinfacht wird. Durch Anhängen von geeigneten Fragegruppen kann der Experte außerdem deklarativ die Benutzerschnittstelle schaffen. Das System ist stets unmittelbar ablauffähig, kann also jederzeit durch die Expertin oder den Experten getestet werden.

Man sollte sich darüber im klaren sein, daß eine Diagnoseshell nicht notwendigerweise für *alle* Diagnoseaufgaben geeignet ist, z.B. wenn die Lösung einer Aufgabe über die hier dargestellten Wissensarten Symptome, Diagnosen und deren statistische Bezüge hinaus weitere Wissensarten benötigt. Entsprechendes gilt für Shells für andere Problemlösungstypen.

7.1.6 Resümee

Erfolg und Grenzen der Expertensysteme Expertensysteme haben begonnen, aus den Forschungslabors in die Anwendung vorzudringen. Es hat sich herausgestellt, daß wirklich *im Wissen Macht liegt* (und Problemlösen nicht einfach durch Means-Ends-Analysis oder schlaue Beweiser möglich ist). Gleichzeitig ist jedoch offenbar geworden, wie umfassend und divers menschliches Wissen ist. Deshalb sind die Anwendungen bislang stärker

spezialisiert als es erhofft wurde. Es gibt große Bereiche von Expertenwissen und insbesondere auch -handeln, die sich einer effizienten Formalisierung mit den bisherigen Mitteln widersetzen. Die erfolgreich modellierbaren Expertiseanteile sind stets in schwer faßbare Bereiche eingebettet. Dazu gehört vor allem „unspezifisch relationales" Wissen, über das Menschen im Hintergrund verfügen und das sie *ad hoc* zur Lösung einer Aufgabe heranziehen können. Diese Erkenntnisse haben sicher dazu beigetragen, daß sich in jüngerer Zeit die Auffassung von Expertensystemen als autonom problemlösenden Systemen zu eher unterstützenden Systemen in kooperativen Arbeitszusammenhängen zu verschieben begonnen hat.

Für Expertensysteme wird von vornherein nur „ausgerichtetes" Wissen erhoben und repräsentiert, nämlich solches Wissen, das auf die Bearbeitung bestimmter (wiederkehrender) Aufgaben zugeschnitten ist. Dieses Wissen wird bezogen – und dies hat sich auch als sinnvoll erwiesen – auf *Problemlösungstypen.* Solche Typen sind ggf. feiner zu typisieren (Untertypen); sie werden entdeckt bei der vergleichenden Beobachtung von Experten bei ihrer Fachtätigkeit. Bei der Wissenserhebung dienen sie als strukturierendes Moment und sind schließlich in Beziehung zu setzen mit den technischen Mitteln für die Wissensverarbeitung. *Hierauf beziehen sich einige der wesentlichen Aufgaben von Knowledge Engineers.* Aber: Es könnte qualifizierte Tätigkeiten geben, die in dem bisher bekannten Rahmen noch nicht verstanden werden können; es gibt Leute, die solchen Typisierungen insgesamt skeptisch gegenüberstehen, da sich die Komplexität bestimmter qualifizierter Tätigkeiten einer Typisierung verschließt. Und das „unspezifisch-relationale" Wissen ist bisher nicht Gegenstand der Betrachtung (auch bei KADS nicht).

Beim gegenwärtigen Stand der Technik ist deshalb das Anwendungsfeld eines Expertensystems stets scharf abgegrenzt, und an seinen Rändern kommt es zum *Kliff-und-Plateau-Effekt.* Große Schwierigkeiten macht nach wie vor auch der Umgang mit unsicherem Wissen. Wo Menschen ohne Nachdenken intuitiv vom Richtigen oder jedenfalls Angemessenen ausgehen und sich mit dem eigentlich stets Ungewissen auseinandersetzen, wo sie mühelos als nicht zutreffend Erkanntes verwerfen und die aktuell wesentlichen abhängigen Folgerungen revidieren, müssen Expertensysteme mühsam Defaults spezifizieren und Annahmen einführen, mit ausgefeilten Mechanismen einen Ballast von Abhängigkeiten verwalten, in dem Wesentliches und Unwesentliches nur nach starren Regeln zu unterscheiden ist.

Knowledge Engineering ist wesentlich eine Design-Aufgabe. Modelle, die zunächst Abbild von Expertise und dann Vorbild für Expertensysteme sind, scheinen ein gutes Leitbild für das Knowledge Engineering zu liefern. Während man früher dazu tendierte, das Wissen gleich bei der Erfassung in die Repräsentationsform zu „gießen" (mittlerweile als „Kübelmethode" verrufen), hat die Erfahrung gezeigt, daß man meistens besser beraten ist, zuerst ein konzeptuelles Modell zu erstellen, das die Expertise interpretiert und das direkt oder über den Zwischenschritt eines weiteren Modells als Vorlage für die Implementierung des Expertensystems dient. Die Schwierigkeit dabei: Das konzeptuelle Modell ist maßgeblich für die Analyse des Expertenwissens und Ausgangspunkt für das Design eines wissens-

basierten Systems, jedoch erfordert das Design mehr als eine mechanische Umsetzung der Analyse. Genau hier liegt einer der Punkte, warum Knowledge Engineering derzeit (und möglicherweise immer) unersetzbar ist. Gutes Design ist allenfalls werkzeugunterstützbar, aber nicht automatisierbar.

Ein Problem ist die Bewertung der Gültigkeit der erstellten Modelle und ihre Abhängigkeit von zeitlichen Veränderungen des Aufgabenbereichs oder auch des Wissens darüber. Ferner muß der Aspekt der Strukturierung von bereichsspezifischem Wissen noch weiter geklärt werden, zum einen auf der technischen Seite (wie läßt sich das effizient und komfortabel realisieren?), zum andern empirisch auf der kognitiven Seite (bildet Expertenwissen ein in sich kohärentes System, oder gibt es eher modulare Wissenseinheiten?). Besonders wichtig wäre dies für die Realisierung von Systemen mit größeren Kompetenzbereichen, denn für die notwendigen großen Wissensmengen scheint eine *auf der Wissensebene* vorgenommene Modularisierung unabdingbar.

Expertise der Knowledge Engineers. Trotz fortgeschrittener Techniken scheint die Zusammenarbeit von Experten und Knowledge Engineers bei der Expertensystemerstellung bislang nur teilweise entbehrlich. Die Wissensakquisition gilt nach wie vor als der Flaschenhals der Expertensystementwicklung. Daß ein Experte sein Wissen direkt mit Hilfe einer Wissenserwerbskomponente in ein System eingibt, ist zur Zeit nur in gut verstandenen, leicht strukturierbaren Anwendungsbereichen möglich. Wissensanalyse und Wissensinterpretation erfordert hochentwickeltes spezifisches *know-how*: die Expertise des Knowledge Engineering, die durch technische Mittel bisher nicht ersetzt werden kann. Die Auswertung von Wissensprotokollen erfordert „weiche" (interpretierende, vom Verstehen abhängige) Vorgehensweisen und ist ein kreativer Akt, der notwendig mit den Schwächen hermeneutischer Vorgehensweisen behaftet ist. Es ist fraglich, ob sie sich je durch „harte" (mechanisch, ohne menschliche Perspektive und Interpretation durchführbare) Methoden, etwa durch maschinelles Lernen ersetzen lassen.

Dies alles gilt zumindest dann, wenn symbolische Verarbeitungsmethoden benutzt werden. Ob sich das Erfahrungswissen von Experten insgesamt auf einer symbolisch-abstrakten Ebene rekonstruieren läßt, ist eine unbeantwortete Frage. Die von Experten behandelten Problemfälle können immer in Variationen auftreten, die in keiner Regelabstraktion berücksichtigt sind, aber trotzdem *Ähnlichkeitsurteile* des Experten ermöglichen. Zur Zeit wird vermehrt diskutiert, ob *Fälle* eine geeignete Grundlage für eine Modellierung von Erfahrungswissen sind. Fallbasierte Expertensysteme sind dort geeignet, wo Wissen über ähnliche Fälle die Arbeit erleichtert bzw. neue Fälle explizit unter Vergleich mit alten bearbeitet werden: etwa in der Medizin, Jurisprudenz, bei Konstruktions- und Planungsaufgaben. Bei erkennbaren Vorteilen, die fallbezogene Modelle haben (z.B. kontextbezogene Wissensmodellierung), ist das schwere Problem der Ähnlichkeitsbewertung nicht gelöst. Ob die flexibler lernenden neuronalen Netze hier Abhilfe schaffen können, ist eine offene Frage. Auch bei positiver Antwort ist nicht unumstritten, ob man sie in Expertensystemen verwenden sollte, da sie ihre Schlüsse oder Aktionen wahrscheinlich kaum erklären können

werden. Kann man solche Systeme, die sich aufgrund ihrer Erfahrungen vielleicht von Tag zu Tag ändern, für wichtige Entscheidungen einsetzen?