KfK 4478
November 1988

Erklärungskomponente für das Expertensystem XUMA unter Berücksichtigung verschiedener Benutzerklassen

K.-P. Huber
Institut für Datenverarbeitung in der Technik
Projekt Schadstoffbeherrschung in der Umwelt

Kernforschungszentrum Karlsruhe

KERNFORSCHUNGSZENTRUM KARLSRUHE

Institut für Datenverarbeitung in der Technik
Projekt Schadstoffbeherrschung in der Umwelt

KfK 4478

Erklärungskomponente für das Expertensystem XUMA unter Berücksichtigung verschiedener Benutzerklassen

Klaus-Peter Huber

Kernforschungszentrum Karlsruhe GmbH, Karlsruhe

Kernforschungszentrum Karlsruhe GmbH
Postfach 3640, 7500 Karlsruhe 1

ISSN 0303-4003

Zusammenfassung

Erklärungskomponente für das Expertensystem XUMA unter Berücksichtigung verschiedener Benutzerklassen

Es werden verschiedene Konzepte zur Realisierung von Erklärungskomponenten in Expertensystemen herausgearbeitet. Dabei stehen drei Aspekte im Mittelpunkt: das Benutzermodell, die Dialogschnittstelle und die Wissensrepräsentation. Das Benutzermodell dient dazu, die spezifischen Anforderungen der Anwender an das Expertensystem zu beschreiben und zu strukturieren. Die Anforderungen aus dem Benutzermodell werden dann in der Dialogschnittstelle auf die vorhandenen Ein- und Ausgabemöglichkeiten abgebildet. Die explizite Darstellung des eingesetzten Wissens bildet die Grundlage der Erklärungsfähigkeit eines Expertensystems. Ausgehend von den genannten Konzepten wurde eine Erklärungskomponente für das Expertensystem XUMA (Expertensystem Umweltgefährlichkeit von Abfallstoffen) entworfen und realisiert. Dabei werden die unterschiedlichen Anforderungen verschiedener Benutzerklassen von XUMA berücksichtigt.

Abstract

Explanation Facility for the XUMA Expert System Taking Account of Different User Classes

Various concepts are elaborated for the development of explanation facilities in expert systems. Three essential aspects are considered: the user model, the user interface, and the representation of knowledge. The user model serves to describe and structure the specific requirements of the users with respect to the expert system. In the user interface the requirements arising from the user model are mapped on existing input and output techniques. The explicit representation of the knowledge constitutes the basis of the explanation capability of an expert system. On the basis of these concepts an explanation facility for the XUMA expert system (Expertensystem Umweltgefährlichkeit von Abfallstoffen = Expert System on Environmental Hazards of Waste) has been designed and implemented. The specific requirements of different user classes of XUMA have been taken into account.

Inhaltsverzeichnis:

1 Einleitung

Die vorliegende Arbeit ist im Rahmen eines gemeinsamen Vorhabens[1)] des Instituts für Datenverarbeitung in der Technik (IDT) im Kernforschungszentrum Karlsruhe und des Instituts für Altlastensanierung der Landesanstalt für Umweltschutz (LfU) in Karlsruhe entstanden. Innerhalb dieses Vorhabens wird das Expertensystem XUMA zur Bewertung der Umweltgefährlichkeit von schadstoffbelasteten Standorten (Altlasten, Mülldeponien) und Abfällen entwickelt. Dieses Expertensystem soll den Anwender bei der Erstellung von problemspezifischen Analysenplänen und bei der Bewertung der Stoffgefährlichkeit auf der Basis von chemisch/-physikalischen Analysen unterstützen.
Eine wesentliche Eigenschaft von Expertensystemen, im Gegensatz zu klassischen Datenverarbeitungs-Systemen, ist die Fähigkeit, das Vorgehen und den Lösungsweg erklären zu können. Da die Ergebnisse von XUMA unter Umständen zu hohen finanziellen Aufwendungen führen, ist die Erklärungskomponente dabei ein wichtiges Element zur Erhöhung der Akzeptanz des Expertensystems. Mit der vorliegenden Arbeit werden Konzepte für eine Erklärungskomponente von XUMA dargelegt und deren Implementierung vorgestellt.

In Kapitel zwei werden die zur Realisierung von XUMA eingesetzten Hardware und Software-Systeme erläutert. Im dritten Kapitel wird dann auf verschiedene Konzepte und deren Realisierung für Erklärungen in Expertensystemen eingegangen. Es folgt eine kurze inhaltliche Beschreibung der Funktionen von XUMA in Kapitel vier. Über die Anwendung einzelner Konzepte für eine Erklärungskomponente und deren Realisierung bei XUMA wird im fünften Kapitel berichtet, bevor das Kapitel sechs die Ergebnisse zusammenfaßt.

[1)] Das Vorhaben wird unter der Nummer PD 87053 durch das Projekt Wasser-Abfall-Boden des Landes Baden-Württemberg gefördert.

2 Rahmenbedingungen

Aufgrund erster Betrachtungen und Analysen der Anforderungen, die an ein System zur Bewertung der Stoffgefährlichkeit von Abfällen gestellt werden, entschied man sich für eine Lösung mit Verfahren und Methoden der künstlichen Intelligenz [Weidemann et al.-87]. Dazu wurden ein Lisp-Rechner und ein hybrides Expertensystem-Werkzeug angeschafft. Über die genaue Ausstattung des Rechnersystems bezüglich Hard- und Software geben die folgenden Unterkapitel einen Überblick.

2.1 Lisp-Rechner mit System-Software

Bei der Durchführung von Expertensystem-Projekten werden häufig sogenannte "KI-Workstations" eingesetzt, um auch auf der Hardware-Ebene eine Unterstützung der KI-Techniken zu erhalten. Der Markt der KI-Workstations teilt sich dabei in die Spezialrechner (Lisp-Maschinen etc.) und die klassischen Workstations auf [Rome, Uthmann-87]. Beim Vergleich der Systeme fällt auf, daß, in Bezug auf den Leistungsumfang, die möglichen Programmierumgebungen und teilweise auch bezüglich der Rechenleistung, die klassischen Workstations inzwischen durchaus als Alternative zu den Spezialrechnern gesehen werden können, zumal sie langfristig die Eingliederung in die konventionelle DV-Umgebung erheblich erleichtern. In der Praxis werden daher heute oft Spezialrechner zur schnellen Entwicklung von Prototypen eingesetzt, während die tatsächlich eingesetzte Programmversion danach auf einem klassischen Rechner reimplementiert wird. Ein erfolgsversprechender Ansatz ist neuerdings die Integration von speziellen Lisp-Chips in Personal-Computern, wie sie derzeit von TI und Apple mit dem MicroExplorer betrieben wird.
Da für die Entwicklung von XUMA ein Lisp-Rechner eingesetzt wird, soll das nächste Unterkapitel kurz auf die Charakteristika von Lisp-Rechnern eingehen.

2.1.1 Charakteristische Eigenschaften von Lisp-Rechnern

Die Bezeichnung Lisp-Rechner [Pleszkum, Thazhuthaveetil-87] resultiert aus dem Sachverhalt, daß diese Klasse von Rechnern von ihrer Architektur und dem Funktionsumfang her den Anforderungen seitens der Sprache Lisp

angepaßt sind. Die meisten der heute kommerziell verfügbaren Lisp-Maschinen basieren auf der Lisp-Maschine, die 1977 vom Massachusetts Institute of Technology (MIT) entwickelt wurde. Die MIT-Lisp-Maschine besaß einen mikroprogrammierbaren Prozessor, der im Microcode die wichtigsten Primitiv-Funktionen von Lisp ausführen konnte. Zusätzlich besaß dieser Lisp-Rechner eine sogenannte 'tagged architecture', bei der in jedem Speicherwort explizit der Datentyp der enthaltenen Information vermerkt wird. Damit ist es möglich, beim Aufruf einer generischen Funktion (Addition, Multiplikation) hardware-gesteuert die richtige Unterroutine aufzurufen. Neben der Entwicklung von Einprozessor-Lisp-Rechnern gibt es auch Entwicklungen mit Mehrprozessor-Architekturen. Hier wird unterschieden zwischen dem Einsatz von einigen spezialisierten Prozessoren (FAIM-1 von Fairchild), die jeweils spezielle Teilaufgaben erfüllen, und dem Einsatz mehrerer identischer Prozessoren zur gleichzeitigen Evaluation von Lisp-Funktionen (EM-3 machine). Der Einsatz solcher Mehrprozessor-Architekturen ist jedoch bisher kaum über die Erprobungsphase hinausgekommen, da die Probleme der Parallelisierung von Lisp-Funktionen noch nicht vollständig gelöst werden konnten.

Als besondere Anforderungen an einen Rechner bei der Verarbeitung von Lisp gelten neben einer effizienten Darstellung und Verarbeitung von Listen, vor allem die Verwaltung der vielen Funktionsaufrufe, die Unterstützung von Bindungsumgebungen und die Speicherverwaltung mithilfe von 'garbage-collecting'.
Ein Problem bei den Funktionsaufrufen ist beispielsweise die nötige Evaluation der Aufrufparameter, die in Lisp- Rechnern dadurch effizienter abläuft, daß abhängig von der Anzahl der Aufrufparameter unterschiedliche Subroutinen angesprungen werden. Da beim Eintritt in Funktionen verschiedenen Variablen lokale Werte zugewiesen werden, muß bei jeder Funktionsevaluation eine Bindungsliste existieren, um beim Austritt aus der Funktion die vorherigen Werte zu aktualisieren. Dabei wird in der Praxis zwischen 'shallow-binding' und 'deep-binding' unterschieden. Beim 'deep-binding' wird beim Funktionsaufruf eine Liste aus Parameter/Wert-Paaren übergeben, wobei bei der Suche nach einem Parameterwert das erste gefundene Paar übernommen wird. Dadurch muß unter Umständen mehrmals die gesamte Bindungsliste durchlaufen werden. Beim 'shallow-binding' werden die Parameterwerte in einer Tabelle verzeichnet, wodurch zwar der

Zugriff beschleunigt wird, aber die Ein- und Austragung neuer Werte mehr Aufwand als beim 'deep-binding' erfordert. Bei der MIT- Maschine und den Lisp-Rechnern, die auf deren Basis entwickelt wurden, wird in der Regel das 'shallow-binding' eingesetzt, aber es existieren auch Mischformen.
Eine andere wesentliche Eigenschaft von Lisp ist die Datenstruktur der Liste als Basisdatentyp. Eine Realisierung auf Maschinenebene, die der üblichen grafischen Darstellung von Listen entspräche (Aufteilung der physikalischen Speicherzellen in Inhalt (car) und Pointer auf das nächste Element (cdr)), hat sich als ineffizient beim Abarbeiten von Listen erwiesen. Daher sind hier verschiedene Verfahren entwickelt worden, wie etwa die Vektorkodierung und die Strukturkodierung. Bei der Vektorkodierung, die in etwas modifizierter Form unter anderem von den Lisp-Rechnern der Firma Symbolics verwendet wird, werden die Listenelemente nacheinander in den Speicher geschrieben. Handelt es sich dabei um eingebettete Listen, so kann mithilfe eines Zwei-Bit-Codes der Speicherinhalt als Pointer interpretiert werden. Die strukturorientierten Verfahren basieren auf der Verwendung von Assoziativspeichern und sind daher noch überwiegend Forschungsgebiet.
Die letzte oben angesprochene Anforderung an Lisp-Rechner ist das 'garbage-collecting'. Dieses 'garbage-collecting' sorgt dafür, daß Speicherplätze für nicht mehr verwendete Listenelemente wieder freigegeben werden. Die Freispeicher-Verwaltung läuft dabei in zwei Phasen ab. In der ersten Phase wird der Speicher durchlaufen und es werden alle freien Zellen markiert, bevor dann in der zweiten Phase die freien Speicherzellen der Speicherverwaltung zurückgegeben werden. Grundsätzlich lassen sich dabei die speicherintensiven Verfahren und die zeitintensiven Verfahren unterscheiden. Da sich die beiden Aspekte hohe Rechengeschwindigkeit und geringer Speicherplatzbedarf als komplementär herausgestellt haben, werden in der Praxis Mischformen eingesetzt, um einen Kompromiß zu erreichen.
Da die Probleme bei der Verwaltung von Funktionsaufrufen und Bindungsumgebungen heute als gelöst gelten, konzentriert sich die Forschung auf die Entwicklung neuer Evaluationsparadigmen, auf eine Parallelisierung in Lisp und auf strukturorientierte Listendarstellungen mit Assoziativspeichern.

2.1.2 Rechner-Konfiguration für die Entwicklung von XUMA

Da XUMA mithilfe eines Expertensystem-Werkzeuges entwickelt werden sollte, das auf Lisp basiert, und da zudem große Datenmengen (Branchenwissen, Bewertungswissen etc.) verarbeitet werden sollten, entschied man sich für die leistungsstarke Rechnerfamilie Explorer der Firma Texas Instruments (TI). Die Rechner vom Typ Explorer sind Einplatz-Rechnersysteme, die als spezielle Lisp-Maschinen konzipiert sind. Der Explorer hat ein Lisp-Betriebssystem und unterstützt in erster Linie Common-Lisp, aus historischen Gründen aber auch Zetalisp. Damit bietet sich dieser Rechner speziell für die symbolische Datenverabeitung mit Lisp an, wobei jedoch auch ein hardware-unterstützter Prolog-Interpreter und Compiler (kompatibel zu Clocksin/Mellish) lieferbar sind. Bei den Explorer-Rechnern sind die Typen I und II lieferbar. Der Explorer II unterscheidet sich von dem Explorer I durch seinen 32 Bit VLSI-Lisp-Mikroprozessor, der von TI speziell für eine effiziente Lisp-Verarbeitung entwickelt wurde, und der auch in dem schon angesprochenen MicroExplorer eingesetzt wird.

Zur Entwicklung von XUMA stehen ein Explorer II mit einer Hauptspeicherkapazität von 16 MByte (virtueller Speicher max. 128 MByte) und ein Explorer II mit 8 MByte Hauptspeicher (virtuell max. 128 MByte) zur Verfügung. Die Kommunikation zwischen den Explorer-Rechnern des Instituts erfolgt über ein 'Local Area Network' (LAN) mit Chaosnet, das auf dem Ethernet-Protokoll aufsetzt. Damit sind Funktionen wie 'File Transfer', 'Remote Access' und 'Electronic Mail' möglich. Zusätzlich können über DECnet unter anderem Files zu und von der institutseigenen VAX/VMS übertragen werden, beispielsweise für Sicherheitskopien ('Backups').
Die eingesetzten Explorer-Rechner verfügen über das Lisp-Betriebssystem Release 3.2 mit verschiedenen Software-Utilities, um den Programm-Entwicklungsprozeß zu verkürzen. Dazu gehören ein syntaxorientierter Lisp-Editor (Zmacs), ein inkrementeller Lisp-Compiler, verschiedene Debugger und sehr mächtige Hilfe-Menüs ('Suggestions'), die zu dem jeweiligen Utility (Editor etc.) die möglichen Kommandos aufzeigen. Zur Datenverwaltung ist ein relationales Datenbanksystem (RTMS) von TI mit einer Lisp-Schnittstelle installiert. Weiterhin verfügen die Rechner über Möglichkeiten, Fenster, Grafiken und Auswahl-Menüs zu erzeugen und zu manipulieren.

2.2 Beschreibung des Expertensystem-Werkzeuges ART

Als Software-Werkzeug wird die Expertensystem-Entwicklungsumgebung ART der Firma Inference verwendet. ART steht dabei für 'Automated Reasoning Tool'. Inzwischen existieren einige Übersichten über den Bereich Expertensysteme, in denen verschiedene Expertensystem-Werkzeuge beschrieben [Harmon, King-86] und auch verglichen werden [Richer-86] [Karras et al.-87]. Dabei muß jedoch beachtet werden, daß sich seit Erscheinen dieser Übersichten der Markt verändert hat. Es existieren heute oftmals neue Versionen der dort genannten Werkzeuge, die einen größeren Funktionsumfang besitzen.

Das hybride Werkzeug ART ist implementiert in der Programmiersprache COMMON-LISP und bietet verschiedene Formen der Wissensdarstellung an, insbesondere Frames (in ART Schemas genannt), Regeln und Lisp-Funktionen, die beliebig kombiniert werden können.
ART war ursprünglich als rein vorwärtsverkettendes Regelsystem konzipiert, und vermutlich aufgrund dieser Tatsache ist das Kontrollwissen in ART an Regeln gebunden oder durch Regeln erzeugt. Dadurch kann zwar die Ablaufsteuerung regelbasiert erfolgen, jedoch fehlt eine saubere Trennung zwischen Regel- und Kontrollwissen. Zur Abarbeitung der Regeln stehen Vorwärts- und Rückwärtsverkettung sowie ein Agendamechanismus zur Verfügung.

Bei der Erstellung eines Expertensystems durch einen Entwickler wird die Wissensbasis in einem File definiert und dann in ART eingelesen. Beim Einlesen übersetzt ART die Wissensbasis in eine effiziente interne Darstellung.
ART gilt als eines der wenigen Expertensystem-Werkzeuge, die auch unter Echtzeitbedingungen eingesetzt werden können [Karras et al.-87]. Dies wird unter anderem dadurch erreicht, daß durch die Vorübersetzung eine sehr effiziente Darstellung der Wissensbasis erzeugt wird. Dies ermöglicht ein schnelles Abarbeiten der Regeln, die Übersetzung hat aber zur Folge, daß ein Teil der Wissensbasis seine ursprüngliche Form verliert. Diese Transformation des extern definierten Wissens in die interne Form kann bei der Erklärung von Abläufen zu Schwierigkeiten führen. Es werden zum Beispiel die Schemainformationen in Fakten dargestellt (Abb. 2.1) und die Regeln

teilweise zerlegt (Abb. 2.2). Damit besteht eine Wissensbasis in ART nach der Übersetzung aus einer Menge von Fakten und von Regeln, die auf der Faktenmenge operieren. Lisp-Funktionen können in Aktionsteilen der Regeln oder auch in Schemas eingesetzt werden.
Im folgenden werden die angesprochenen Konzepte von ART näher erläutert.

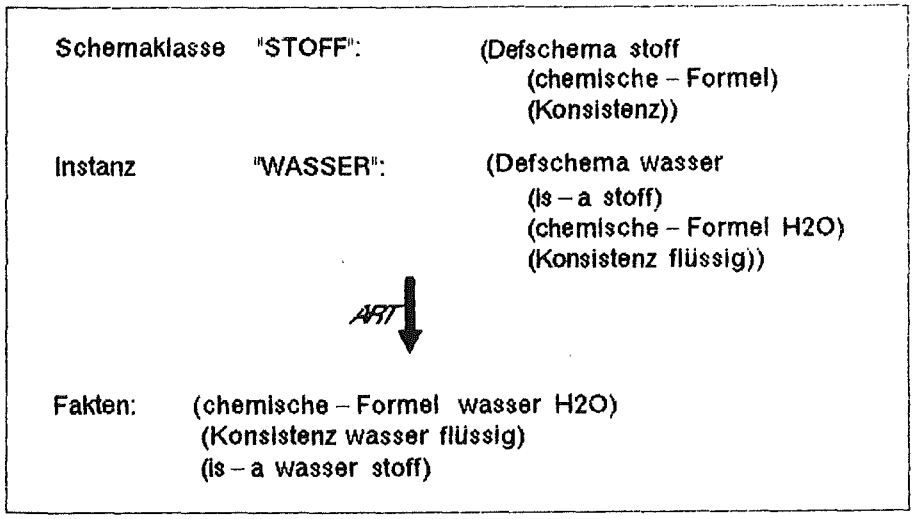


Abb. 2.1: Zerlegung von Schemainformation in Fakten

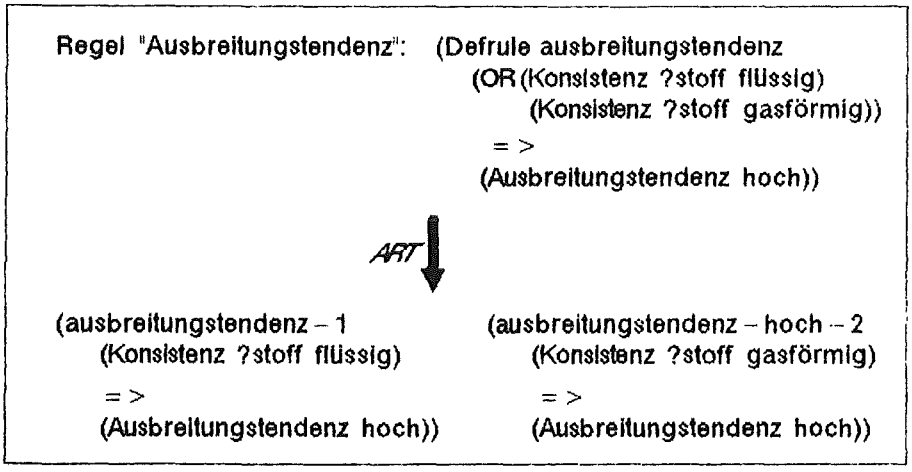


Abb. 2.2: Regelzerlegung bei einer Regel mit Oder-Prämisse

Schemas in ART:

Als Schema wird in ART das bezeichnet, was im Bereich der wissensbasierten Systeme sonst meistens Frame genannt wird. Jedes Schema besteht aus einem identifizierenden Namen und beliebig vielen Slots. Das gesamte Schemasystem in ART basiert auf wenigen Wurzel-Schemas ('kernel schema'), die sehr allgemein definiert sind. Schemas und Slots, die vom Benutzer definiert werden, sind jeweils wieder Schemas, die Instanzen der Wurzelschemas sind. Damit ergibt sich ein sehr flexibles und in sich konsistentes Hilfsmittel zur Definition von Objekten.

Der Slot eines Schemas hat verschiedene Eigenschaften ('Facets'), die vom Programmierer definiert werden können. Diese Eigenschaften beinhalten beispielsweise den Vererbungsmechanismus oder die Anzahl von Slotwerten ('single-value' oder 'multiple-value'). Zur Bildung von semantischen Netzen oder hierarchischen Strukturen können Schemas durch definierbare Relationen beliebig miteinander verbunden werden. Diese Relationen werden als Slots in Schemas sichtbar (etwa 'is-a' oder 'has-instances') und können vom Entwickler individuell definiert werden. ART stellt fünf verschiedene Beziehungsarten zur Verfügung, wie zum Beispiel die Klasse-Instanz- oder die Menge-Untermenge-Beziehung. Zusätzlich kann eine Abhängigkeit zweier Relationen so definiert werden, daß beim Einfügen einer Beziehung automatisch andere Relationen eingetragen werden, wenn für bestimmte Objekte bestimmte Relations-Ketten gelten. Existiert zum Beispiel eine Beziehung "Vater-von" und "verheiratet-mit", so kann definiert werden, daß beim Eintragen einer "Vater-von"-Beziehung eine "Schwiegervater-von"-Beziehung erzeugt wird, wenn das entsprechende Kind in der Relation "verheiratet-mit" steht (Abb. 2.3).

Neben den Relationen, die zwischen Schemas definiert werden, können auch allgemeine Relationen (wie etwa "ist-äquivalent-mit") dargestellt werden. Zu diesen allgemeinen Relationen können logische Abhängigkeiten definiert werden. Damit kann eine Relation beispielsweise als transitiv, reflexiv oder symmetrisch definiert werden, was dazu führt, daß von ART automatisch Rückwärts-Regeln generiert werden, die für die Einhaltung dieser Eigenschaften sorgen.

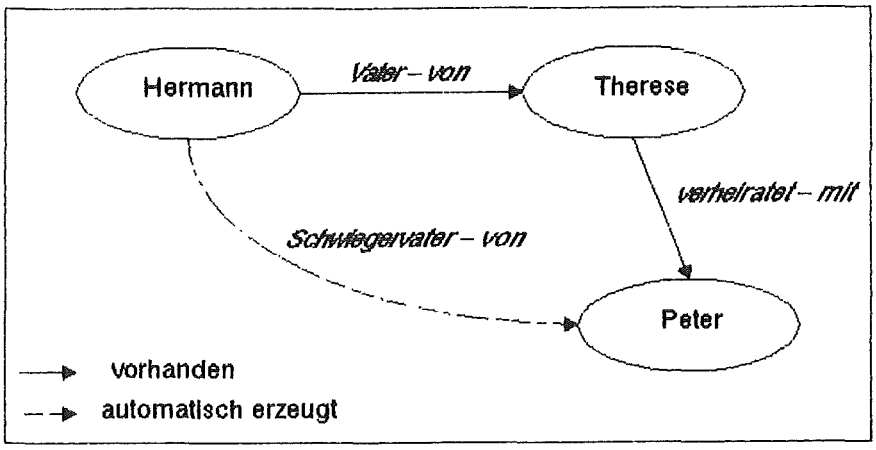


Abb. 2.3: Grafische Darstellung der logischen Abhängigkeit von Relationen

Weiterhin steht in ART ein Vererbungsmechanismus zur Verfügung, mit dem Slots und Slotwerte an andere Schemas vererbt werden können. Dabei kann zum Beispiel definiert werden, welcher Slotwert übernommen werden soll, wenn dieser Slot von zwei oder mehreren Schemas gleichzeitig vererbt wird ('multiple inheritance').
Zur Unterstützung der objektorientierten Programmierung gibt es in ART zwei Konzepte. Das erste entspricht der klassischen objektorientierten Vorstellung, wobei jedem Schema Methoden (Lisp-Funktionen) zugeordnet werden, die durch das Senden von Nachrichten zwischen den Objekten ('message passing') aktiviert werden. Damit ist es zum Beispiel möglich, generische Prozeduren zu implementieren. Ein anderes Konzept in ART stellen die sogenannten "aktiven Werte" ('active values' oder 'demons') dar, die Slots zugeordnet werden können. Diese aktiven Werte reagieren auf jedes Auslesen, Einlesen oder Verändern der entsprechenden Slotwerte so, daß bei jedem Zugriff automatisch eine Lisp-Funktion aufgerufen wird. Damit gelingt es beispielsweise, Trigger-Funktionen zu realisieren, die bei der Eintragung von Slotwerten diese erst noch umrechnen. Oder es können "aktive Grafiken" erzeugt werden (z.B. Tachometer oder Schalter), die immer den aktuellen Stand der Wissensbasis aufzeigen. Wird dann ein Wert in der Wissensbasis verändert, so kann mithilfe eines aktiven Wertes eine ständige Aktualisierung der Grafik erreicht werden.

Regeln in ART:

In ART bestehen Regeln aus einem Namen, einem Kommentar, einem Prämissen- und einem Aktionsteil. Wie oben schon kurz angesprochen wurde, besteht die Wissensbasis in ART neben den Regeln aus Fakten. Grob gesagt dienen Regeln dazu, aufgrund existierender Fakten neue Fakten zu erzeugen, alte Fakten zu löschen oder beliebige andere Aktionen (Lisp-Aufrufe) auszulösen. Dazu werden im Prämissenteil sogenannte Muster ('Pattern') angegeben, die der Regelinterpreter mit der vorhandenen Faktenmenge vergleicht. Die Muster in den Prämisssen einer Regel können beliebig komplex über "Und", "Oder", "Nicht" und andere spezielle Quantoren miteinander verschachtelt werden. Beim Einlesen der Regeln in ART werden die Regeln dann teilweise in einfachere Regeln zerlegt, um eine effiziente Abarbeitung zu erreichen. Beim Ablauf versucht der Regelinterpreter, die Fakten der Faktenbasis mit den Mustern der Regeln zu 'matchen'. 'Matchen' bedeutet dabei, daß die Muster und die Fakten in der Datenbasis auf Übereinstimmung geprüft werden, und daß dann innerhalb des Musters die angegebenen Variablen instanziiert werden. Können alle Muster im Prämissenteil einer Regel erfolgreich 'gematcht' werden, so wird die Regel auf die Agenda gelegt. Dabei kann eine Regel zu einem Zeitpunkt auch mehrfach auf der Agenda liegen, wenn die Muster mit unterschiedlichen Fakten 'gematcht' werden konnten. Agenda ist hier der Begriff für eine geordnete Liste, in der die Regeln eingetragen sind, die "feuern" können (siehe auch Inferenzmechanismen).

Durch die Mächtigkeit des Pattern-Mechanismus ist es möglich, die Regeln auch zur Ablaufsteuerung einzusetzen. Zur Abbildung verschiedener Bearbeitungsphasen kann beispielsweise ein Fakt 'current-phase' definiert werden, das die aktuelle Ablaufphase enthält. Wird dann in einigen Regeln das Faktum (current-phase init) als Bedingung eingesetzt, so feuern diese Regeln nur in der Phase 'init'. Damit kann erreicht werden, daß immer nur eine spezielle Regelmenge ('rule-set') aktiv sein kann. Nach Abarbeitung der Regeln dieser Phase wird dann das Faktum durch eine andere Regel umgesetzt, zum Beispiel auf (current-phase ablauf). Die Abbildung 2.4 soll das oben Geschilderte noch verdeutlichen. Die gezeigte Regel wird so oft aktiviert, wie in der Faktenbasis die Fakten (zustand ?stoff flüssig) und (current-phase init) 'gematcht' werden können. Dabei wird die Variable ?stoff jeweils mit den entsprechenden Werten instanziiert.

```
(Defrule Testregel
    (declare (salience 1000))
    (current – phase init)
    (zustand ?stoff flüssig)
    = >
    (Aktionen))
```

Abb. 2.4: Definition einer ART-Regel, die auf die Relationen 'current-phase' und 'zustand' zugreift.

Im Aktionsteil von ART-Regeln können sowohl Fakten und Schemas gelöscht und erzeugt, als auch Lisp-Funktionen aufgerufen werden. Da innerhalb von Lisp-Funktionen wieder Regeln definiert oder Fakten erzeugt werden können, sind sehr komplexe Aktionen realisierbar.
Regeln können Aktionen ausführen und/oder logische Abhängigkeiten definieren. Hat beispielsweise eine Regel eine Abhängigkeit zwischen Fakt A und Fakt B erzeugt, und wird dann A später zurückgezogen, so wird auch das logisch abhängige Fakt B wieder gelöscht. Als Einschränkung muß jedoch gesagt werden, daß sich diese Abhängigkeit nur beim Hinzufügen (wenn A, dann B) und Löschen (Gilt A nicht mehr, dann lösche B) von Fakten benutzen läßt. Versucht man aber, eine logische Abhängigkeit durch eine Regel "Wenn A, dann lösche B" zu definieren, dann würde nach dem späteren Zurückziehen von A das Fakt B nicht wieder erzeugt.

Inferenzmechanismen:
Zur Verfügung stehen hier die Vorwärts- und die Rückwärtsverkettung. ART war ursprünglich als vorwärtsverkettendes Regelsystem konzipiert, und vermutlich liegt daher die Stärke von ART in der Vorwärts-Verarbeitung von Regeln. Die Abarbeitung von Rückwärts-Regeln, die sich syntaktisch an die Vorwärts-Regeln anlehnen, ist in ART etwas umständlich zu handhaben. Hinzu kommt noch, daß eine Regel jeweils nur in einer Richtung ausgewertet werden kann, was bedeutet, daß Regeln gegebenenfalls zweimal

definiert werden müssen. Es ist jedoch kein Problem, während des Ablaufes zwischen Vorwärts- und Rückwärts-Verkettung zu wechseln.
Zusätzlich bietet ART eine Möglichkeit an, den Ablauf der Regeln, die sich auf der Agenda befinden, zu beeinflussen. Dieser Agendamechanismus dient dazu, zu steuern, welche der aktivierten Regeln als nächste ausgeführt werden soll. Die Ordnung der Agenda beruht prinzipiell auf dem LIFO-Prinzip ('Last-In-First-Out'). Zusätzlich kann jeder Regel bei der Definition eine Priorität ('salience') zugeordnet werden, die eine Sortierung innerhalb der Agenda erlaubt. Dabei kommen Regeln mit höheren Prioritäts-Werten vor die Regeln mit niedrigeren Prioritäts-Werten. Nach der Ausführung einer Regel wird die Agenda aktualisiert, da sich beim Ausführen der Regel die Faktenmenge geändert haben kann. Es können neue Regeln aktiviert sein, weil alle Prämissen dieser Regeln jetzt gelten, oder es können auf der Agenda vorhandene Regeln inaktiv werden, weil nicht mehr alle Prämissen erfüllt sind. Befinden sich beispielsweise zwei Regeln R0 mit Priorität 20 und R1 mit Priorität 5 auf der Agenda, und es kommt die Regel R3 mit Priorität 10 dazu, so wird als erstes R0 ausgeführt. Danach wird die Agenda aktualisiert, und wenn sich nichts geändert hat, kommt dann R3 zur Ausführung, bevor schließlich R2 "feuert".

Viewpoints:
Ein sehr komplexes Hilfsmittel von ART sind die 'Viewpoints'; das sind hypothetische Welten, die dynamisch erzeugt, verglichen und gelöscht werden können. Viewpoints können zum Beispiel zur Darstellung von Zeit- oder Ortsbedingungen verwendet und auch mehrdimensional eingesetzt werden.
Als Beispiel soll das "n-Damen Problem" dienen. Es geht hier darum, auf einem n*n großen Schachbrett n Damen (vom Schachspiel) aufzustellen, ohne daß sie sich schlagen können. Dieses Beispiel soll mithilfe des Viewpoint-Mechanismus in ART gelöst werden. Dazu werden ausgehend von einer Dame sukzessive neue Damen auf das Feld hinzugenommen. Jedesmal wird dafür ein neuer Viewpoint erzeugt, der somit einer noch unvollständigen Stellung entspricht. Wird festgestellt, daß sich die Damen in einem Viewpoint schlagen können, so wird diese Welt gelöscht. Stellt ART fest, daß zwei Welten symmetrische, also identische Stellungen haben, so werden die beiden Welten in einer Welt zusammengefaßt. Die Welten werden solange erzeugt, bis ein Abbruchkriterium für die Ausgabe der

Lösungen sorgt. Das Erzeugen ('hypothesize'), Zusammenfassen ('merge') und Löschen ('poison') von Welten wird in ART mit Regeln realisiert.
Ein anderes Beispiel wäre ein Diagnosesystem, bei dem die Hypothesen als Viewpoint dargestellt werden. Hypothesen-Viewpoints, die sich bei genauerer Betrachtung nicht bestätigen lassen, werden dann gelöscht.

Weiterhin bietet ART ein Benutzer-Interface an, mit dem aktive Grafiken, Fenster, Pop-Up-Menus und ähnliches definiert und manipuliert werden können. Genaue Details über die Möglichkeiten und über deren Anwendung können in den entsprechenden Handbüchern der verwendeten ART-Version nachgeschlagen werden.

3 Konzepte zur Realisierung von Erklärungskomponenten

Die Eigenschaften, durch die sich Expertensysteme von klassischen DV-Systemen unterscheiden, sollen hier nicht einzeln aufgezählt werden. Ein ganz wesentlicher Unterschied ist jedoch, daß Expertensysteme (ES) fähig sind, die hergeleiteten Aussagen zu rechtfertigen [Appelrath-85] [Puppe-88]. Auch Hawkins, der das 'Denken von Experten' untersucht hat [Hawkins-83], verweist darauf, daß die Erklärungsfähigkeit zu den wesentlichen Eigenschaften von Expertensystemen gehört. Diese Erklärungen, oft auch mit Rechtfertigungen bezeichnet, beziehen sich dabei meist auf den Herleitungsweg von Aussagen und auf den Systemzustand.
Erstaunlicherweise ist jedoch in den meisten bisher entwickelten Expertensystemen die Erklärungskomponente eher als rudimentär zu bezeichnen, obwohl gerade sie einen wesentlichen Einfluß auf die Akzeptanz durch den Benutzer hat. Da Expertensysteme in spezialisierten, meist nicht formal definierbaren Problembereichen eingesetzt werden, benötigt der Anwender (der meist kein Experte ist) Informationen über die Herleitung der Ergebnisse. Er erwartet vom System eine transparente Darstellung der internen Abläufe. Da es bisher nicht möglich ist, bei wissensbasierten Systemen eine formale Verifikation durchzuführen, ist der Anwender bei der Überprüfung der Ergebnisse darauf angewiesen, daß er die Aktionen und Reaktionen des Expertensystems nachvollziehen kann. Aus diesem Grund ist die Existenz einer benutzergerechten Erklärungskomponente ein wichtiges Kriterium für Expertensysteme. Die Bezeichnung Komponente wurde hier gewählt, da die Erklärungsfähigkeit durch ein eigenständiges Programm-Modul innerhalb des Gesamtsystems realisiert werden sollte. Sie sollte so unabhängig realisiert werden, daß bei einer Erweiterung oder Änderung der Wissensbasis keine oder kaum Anpassungen der Erklärungskomponente notwendig werden.

3.1 Historische Entwicklung der Erklärungsfähigkeit bei Expertensystemen

Das Expertensystem MYCIN zur Diagnose und Therapie bestimmter Infektionskrankheiten wird heute als "Großvater" der Expertensysteme bezeichnet und oft als Vergleichsmaßstab für andere Expertensysteme

genommen [Puppe-88]. Es existierten zwar vorher schon Systeme wie MACSYMA zur Manipulation algebraischer Gleichungen oder DENDRAL zur Molekülidentifikation mithilfe von Massenspektogrammen, die mit bereichsspezifischem Wissen arbeiteten. Aber erst MYCIN enthielt explizit eine Erklärungskomponente, die es dem Benutzer erlaubte, Fragen an das System zu stellen.
MYCIN war ein Expertensystem, dessen Wissen ausschließlich in rückwärtsverketteten Regeln repräsentiert wurde [Buchanan, Shortliffe-84]. Beim Ablauf einer Diagnose durch MYCIN mußte der Benutzer verschiedene Fragen beantworten, aus deren Antworten MYCIN mithilfe seiner Regelbasis eine Diagnose ableitete. Immer wenn das System eine Frage stellte, konnte der Benutzer an das System bestimmte Fragen stellen. Dabei konnte der Benutzer aus zwei Fragetypen auswählen, um sich das Vorgehen von MYCIN erklären zu lassen. Er konnte entweder die Frage stellen "Warum wurde die aktuelle Frage gestellt?" oder "Wie ist eine bestimmte Schlußfolgerung hergeleitet worden?". Auf die Wie-Frage stellte MYCIN die Regeln dar, die bereits gefeuert hatten und deren Aktionsteile die Schlußfolgerung enthielten. Als Antwort auf die Warum-Frage erhielt man die Regel angezeigt, deren Bearbeitung zu der aktuellen Frage geführt hatte. Diese beiden Möglichkeiten bildeten die Erklärungskomponente von MYCIN und gelten heute noch als Mindestanforderungen an Erklärungskomponenten bei Expertensystemen.

Beim praktischen Einsatz von MYCIN zeigte es sich, daß die Erklärungen des Systems den Anforderungen der Benutzer nicht genügten. Da die Regeln neben dem Diagnosewissen auch Steuerinformationen, Kontrollwissen (beispielsweise die Reihenfolge der Prämissen, die Einfluß auf den Ablauf hatte) etc. enthielten, konnte der Benutzer das gesamte Diagnosewissen, das hinter dieser Regel steckte, nicht erkennen. Daher versuchte Weiner ein Expertensystem zu entwickeln, das insbesondere eine benutzernahe Erklärung ermöglichen sollte [Weiner-80]. Er nahm dazu verschiedene linguistische und psychologische Studien zu Hilfe, um mit dem Expertensystem 'Blah' die Erklärungen in der Art und Weise zu formulieren, die der von Experten entspricht. Dazu wurden beispielsweise bei der Betrachtung von Alternativen die Unterschiede vor den Gemeinsamkeiten aufgezeigt. Auch wurden in 'Blah' ganze Herleitungsketten dargestellt und nicht nur einzelne Regeln, wie etwa in MYCIN.

Das Anliegen von Weiner war es, durch eine verbesserte Erklärungsfähigkeit von Expertensystemen deren Benutzerakzeptanz zu erhöhen. Ein anderer Ansatz wurde von Clancey verfolgt, der versuchte, das Expertenwissen von MYCIN mithilfe der Erkärungen zur Schulung von Studenten einzusetzen [Clancey-83]. Dabei stellte er fest, daß die vorhandene Wissensbasis dazu nicht geeignet war. Das Problem lag darin, daß aus der Formulierung einer Regel nicht hervorging, warum gerade diese Prämissen in dieser Reihenfolge formuliert worden waren. Clancey stellte fest, daß die Regeln sowohl Kontrollstrukturen als auch Diagnosen-Vorauswahlen und Klassifikationen enthielten.
Bei der genaueren Analyse dieses Sachverhalts zeigte sich als wichtigstes Problem das implizit in der Wissensbasis vorhandene Wissen. Da dieses implizite Wissen von einer Erklärungskomponente nicht erzeugt oder ausgegeben werden kann, bleibt es für den Anwender unsichtbar. Da aber der Ablauf von Regeln meist bestimmten Strategien folgt (wie 'Hypothesize and Test', 'Divide and Conquerer'), forderte Clancey, daß diese Strategien explizit dargestellt und in Form von Regeln formuliert sein müssen, um eine akzeptable Erklärungsfähigkeit zu erreichen. Dazu sollte jedoch vorher eine Einteilung des verwendeten Wissens in die Bereiche strategisches, strukturelles und unterstützendes Wissen erfolgen.
Zusätzlich zur expliziten Darstellung des Wissens erwies es sich als hilfreich, neben dem heuristischen Wissen kausales Wissen (Ursache-Wirkung) in den Problemlösungsprozeß einfließen zu lassen. Das kausale Wissen hat dabei besonders im Bereich der diagnostischen Expertensysteme an Bedeutung gewonnen. Einerseits wird es eingesetzt, um die Ergebnisse durch zugrundeliegendes "Tiefenwissen" zu rechtfertigen, andererseits kann es auch zur Diagnose selbst verwendet werden [Davis-84]. Der Einsatz von kausalem Wissen ist jedoch nur in Bereichen möglich, in denen die Abhängigkeiten zwischen Ursachen und Wirkungen hinreichend bekannt sind. Da aber Expertensysteme oftmals gerade in diffusen Bereichen eingesetzt werden, weil sie es erlauben, ohne Tiefenwissen zu arbeiten, scheint die Verwendung von Tiefenwissen nur in Spezialfällen sinnvoll.

Noch konsequenter als in den oben beschriebenen Arbeiten versuchte Swartout die Forderung nach der Erklärungsfähigkeit eines Expertensystems zu erfüllen [Swartout-88]. Swartout forderte nicht nur eine

verbesserte Darstellung des im ablauffähigen Expertensystem eingesetzten Wissens, sondern er ging soweit, daß er das Expertensystem aus dem explizit vorhandenen Wissen erst generierte. Exemplarisch implementierte er dazu in seinem EES-Projekt (Explainable Expert System) eine Wissensbasis, die in terminologisches, beschreibendes und problemlösendes Wissen aufgeteilt war sowie einen Programmgenerator, der aus der Wissensbasis das eigentliche Expertensystem erzeugte. Dadurch wurde erreicht, daß jede Aktion und Reaktion des Expertensystems durch das zugrundeliegende Wissen erklärt werden konnte.

3.2 Bedeutung der Erklärungskomponente

Wie in der Einleitung dieses Kapitels schon kurz geschildert wurde, zeichnen sich Expertensysteme unter anderem durch ihre Erklärungsfähigkeit aus. Der Begriff der Erklärung oder auch Rechtfertigung wird in dem Sinne verwendet, daß dem Anwender die Zustände des Systems und der Regelablauf transparent gemacht werden sollen. In der Literatur werden verschiedene Gründe für die Notwendigkeit einer Erklärungskomponente in einem Expertensystem angeführt. Nachfolgend werden zwei davon exemplarisch wiedergegeben:

Appelrath nennt dazu

> die möglicherweise folgenschweren Entscheidungen aufgrund von Ergebnissen,
> die Erkennung der Argumentationskette und
> die Fehlererkennung [Appelrath-85].

Puppe führt in diesem Zusammenhang

> die Plausibilitätskontrolle der Lösung und Transparenz für den Laien,
> die Nachvollziehbarkeit und Darstellung wichtiger Prinzipien,
> den Nachweis der Korrektheit und
> die Rückführung des Ergebnisses auf zugrundeliegendes Wissen an [Puppe-88].

Auf den ersten Blick liegt die Vermutung nahe, daß die Erklärungsfähigkeit eines Expertensystems mit der Dokumentation der Regeln zu vergleichen

ist. Bei einigen Expertensystemen beschränkt sich die Erklärungskomponente tatsächlich darauf, vorbereitete Texte auszugeben. Vergleicht man jedoch eine einzelne Funktion eines klassischen Programms (klassisch im Sinne eines sequentiellen Programms) mit einer Regel in einem wissensbasierten Expertensystem, so ergeben sich im Hinblick auf ihre Bedeutung zwei wesentliche Unterschiede. Diese liegen dabei einmal in der Detaillierung und zum anderen in der Bedeutung der Reihenfolge begründet.

In einem klassischen Programm machen einzelne Funktionsaufrufe nur im Zusammenhang mit den vorhergehenden und den folgenden Programmzeilen einen Sinn. Da eine isolierte Funktion im allgemeinen keine anwendungsspezifische Bedeutung hat, und dieselbe Funktion in mehreren Programmen mit jeweils unterschiedlicher Bedeutung vorkommen kann, reicht es im allgemeinen aus, das funktionale Verhalten klassischer Programme auf Modulebene zu beschreiben. (Das Gesagte bezieht sich nicht auf die trotzdem notwendige Dokumentation einzelner Programmzeilen zur Erhöhung der Lesbarkeit des Programmcodes.) Bei regelbasierten Systemen ist das anders. Jede Regel stellt für sich einen bestimmten Sachverhalt dar, der unter Umständen beliebig komplex sein kann (siehe dazu auch Ausführungen über Clancey in Kapitel 3.1). Dieser Sachverhalt ist (idealerweise) nur fachspezifisch und hat auch losgelöst von der Regelmenge eine eigenständige Bedeutung. Vor allem die Abhängigkeit zwischen Prämissen und Aktionen in einer Regel bedürfen einer Erklärung, die dem Anwender des Expertensystems zur Verfügung gestellt werden muß.

Der zweite, noch wichtigere Unterschied liegt in der Bedeutung der Reihenfolge bei Programmzeilen (somit auch von Funktionsaufrufen), die durch das Programm fest definiert werden, und bei Regeln, die im allgemeinen ungeordnet sind. Dieser Unterschied beruht auf der Trennung zwischen Wissensbasis und Ablaufsteuerung in wissensbasierten Systemen. In Expertenssystemen stellt der Systementwickler das Expertenwissen beispielsweise in Regeln dar, während der Regelinterpreter den Ablauf steuert und eine Lösung erarbeitet. Im Prinzip muß also der Systementwickler keine Problemlösungsverfahren (Algorithmen) erarbeiten, die aus seinen Daten eine Lösung ermitteln. In der Praxis ist jedoch auch in den Regeln Kontroll- und Steuerwissen vorhanden, um einen effizienten Ablauf zu ermöglichen. (Das klassische Beispiel hierzu ist PROLOG, in dem die Reihenfolge der Klauseln

nicht nur Einfluß auf die Effizienz, sondern auch auf die Terminierung des Programmes hat.) Um die ausgegebenen Lösungen kontrollieren zu können, muß der Benutzer die Möglichkeit haben, während oder nach dem Ablauf des Systems die Reihenfolge und die Namen der Regeln zu erfahren, die zu einer Lösung geführt haben.

Die Erklärungsfähigkeit eines Expertensystems existiert nicht schon durch die Existenz einer Wissensbasis, sondern muß durch ein eigenständiges Programm-Modul realisiert werden. Abhängig von der Realisierung des Expertensystems wird auch die Erklärungskomponente mit Regeln, Frames oder Methoden arbeiten. Sie muß sowohl auf die statische und dynamische Wissensbasis als auch auf die Inferenzprozeße zugreifen können (Abb. 3.1). Dabei sollte sie als ein unabhängiges Modul innerhalb des wissensbasierten Systems entwickelt werden.

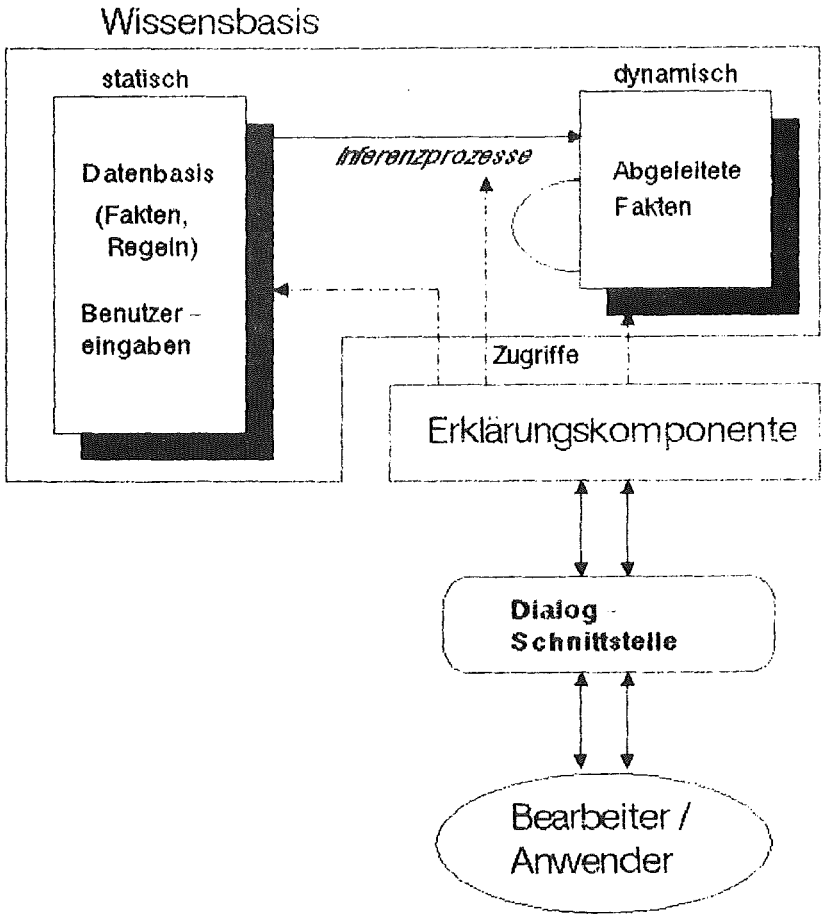


Abb. 3.1: Integration der Erklärungskomponente

Faßt man die hier angesprochenen Gesichtspunkte zusammen, so treten drei Aspekte zur näheren Betrachtung in den Vordergrund:

1) Akzeptanz durch den Anwender
Gerade diesem Punkt kommt in der Praxis die größte Bedeutung zu. Wie sich bei vielen Softwareprodukten gezeigt hat, hängt die Akzeptanz eines Programmsystems nicht nur von der tatsächlichen Leistungsfähigkeit des Systems, sondern ganz wesentlich auch von der Schnittstelle zum Anwender ab.
Speziell bei Expertensystemen kommt noch ein psychologischer Aspekt zum Tragen. Expertensysteme nehmen für sich in Anspruch, die Arbeit eines Experten unterstützen, simulieren oder gar ersetzen zu können. Aus einer "Rationalisierungsfurcht" heraus besteht daher beim Anwender und beim Experten ein Mißtrauen gegenüber solchen Systemen. Zum Beispiel erklären Schnupp und Huu die größere Beliebtheit von DENDRAL (ein Expertensystem zur Bestimmung der Struktur komplexer Moleküle) gegenüber MYCIN damit, daß sich DENDRAL dem Benutzer als "Assistent des Experten", MYCIN sich aber als "Experte" darstellt [Schnupp, Huu-87].

2) Test und Validation der Wissensbasis
Für die Erstellung einer Wissensbasis und deren Wartung spielen die Fehlersuche und die Systemoptimierung (Tuning) eine wichtige Rolle. Die Entwickler eines Expertensystems kennen sich mit dem zugrundeliegendenWerkzeug oder der verwendeteten Programmiersprache soweit aus, daß sie bei der Fehlersuche die System-Hilfsmittel (Debugger, Tracer, etc.) benutzen können. Der Fachexperte oder der Anwender aber, der eine Wissensbasis auf semantische Korrektheit prüfen oder die Wissensbasis optimieren will, kann nicht auf diese "Werkzeug-Ebene" zurückgreifen. Daher muß ihm ein Hilfsmittel bereitgestellt werden, mit dem er seine Aufgaben erfüllen kann. Bei der Wartung und der Erweiterung einer Wissensbasis benötigt der Fachexperte die Erklärungskomponente, um die Modifikationen der Wissensbasis auf ihre Korrektheit hin untersuchen zu können. Eine Wissensakquisitionskomponente, die es dem Fachexperten ermöglicht, ohne Wissensingenieur die Wissensbasis zu definieren und zu verändern, prüft die Eingaben im allgemeinen nur auf syntaktische Korrektheit. Die Integration

des eingegebenen Wissens in die Wissensbasis kann dann mit der Erklärungskomponente getestet werden.

3) Nachvollziehbarkeit des zugrundeliegenden Wissens
Ein anderer Gesichtspunkt von Bedeutung, der für Clancey Ausgang seiner Untersuchungen über MYCIN war [Clancey-83], ist die Möglichkeit für den unerfahrenen Anwender, die Herleitung der Ergebnisse nachvollziehen und erlernen zu können. Im wesentlichen bedeutet das, daß der Anwender vom Expertensystem lernt, welche Lösungsstrategien und welches Basiswissen das System zur Herleitung von Lösungen verwendet hat. Gelingt es dem Anwender, sich neben dem nötigen Grundlagenwissen auch die Lösungsverfahren vom Expertensystem anzueignen, so kann er den Wissensstand eines dedizierten Fachexperten erreichen.
Als Beispiel soll hier die klassiche Diagnose dienen. In der Diagnose geht es darum, aus einer Menge meist quantitativer Symptome (z.B. Benzinverbrauch in Litern/100km oder Cyanidgehalt in mg/l) über Vorinterpretationen (Benzinverbrauch ist zu hoch etc.) erste Hypothesen zu generieren. Die Hypothesen werden auf Differentialdiagnosen (Diagnosen, die die gleiche Symptommenge erklären) hin untersucht und dann entweder bestätigt oder verworfen. Als letztes wird aus den restlichen Hypothesen die wahrscheinlichste Hypothese als Diagnose bestimmt. Existiert dieser Ablauf in einem Diagnose-Expertensystem explizit (beispielsweise als Strategieregeln, wie es Clancey in [Clancey-83] fordert), so kann ein unerfahrener Benutzer diese Vorgehensweise erkennen und übernehmen.
In der Praxis spielt diese allgemeine Anforderung an Expertensysteme, außer bei speziellen Lehrsystemen (CAI, Computer Aided Instructions), jedoch nur eine untergeordnete Rolle.

Die drei hier beschriebenen Aspekte sind nicht für jeden Benutzer gleich wichtig. Welcher Aspekt der Erklärungskomponente für welchen Benutzer am wichtigsten ist, hängt im allgemeinen von seinen Erwartungen an das System ab. Bei einer Analyse der einzelnen Benutzer eines Expertensystems ergeben sich im wesentlichen drei Benutzerklassen. Diese drei Benutzertypen sind der Wissensingenieur, der Experte und der Anwender. Jeder dieser Benutzer hat dabei seine eigenen Anforderungen bezüglich der Notwendigkeit und der Ausgestaltung einer Erklärungskomponente. Während beispielsweise der Anwender an einer Schnittstelle interessiert ist,

die mit ihm in seiner Fachsprache kommuniziert und nur das semantisch sinnvolle Wissen darstellt, ist der Systementwickler gerade darauf angewiesen, auch sämtliche Kontroll- und Steuerinformationen in systemnaher Darstellung zu erhalten.

Wird der Begriff der Erklärung etwas weiter gesehen, so bietet es sich an, dem Benutzer innerhalb einer Erklärungskomponente noch zusätzliche Informationen zur Verfügung zu stellen. Diese Informationen können beispielsweise Werte von Objekten, Kommentare oder auch Tabellenwerte sein. Auch eine Form von Thesaurus ist denkbar. Durch diese Erweiterungen kann erreicht werden, daß der Anwender das System als nützliche Informationsquelle betrachtet, die ihn auch beim Aufsuchen von allgemeiner Fachinformation unterstützt.

3.3 Anforderungen und Realisierungsansätze bei Erklärungen in Expertensystemen

Wie schon kurz angedeutet wurde, existieren verschiedene Ansätze zur Realisierung von Erklärungskomponenten in Expertensystemen. Neben den in Kapitel 3.1 geschilderten Arbeiten, die sich mit Grundlagen der Erklärungsfähigkeit beschäftigten, gibt es zahlreiche Arbeiten, die spezielle Teilaspekte bei Erklärungskomponenten behandeln. Dabei wird unter anderem betrachtet, inwieweit Textframes mit zusätzlichen Informationen zur Erklärung beitragen können [Molokova-86], oder wie unsicheres Wissen bei Erklärungen berücksichtigt werden kann [Strat-87] [Ganascia-85].
Faßt man die einzelnen Konzepte und Betrachtungsrichtungen zusammen, so ergeben sich drei Aspekte, die bei der Realisierung einer Erklärungskomponente berücksichtigt werden müssen:
Es handelt sich dabei um das Benutzermodell, mit dem Anforderungen des Benutzers beschrieben werden können, um die Dialogschnittstelle, die für die Ausgestaltung der Kommunikation zwischen Benutzer und Expertensystem zuständig ist und um die Wissensrepräsentation innerhalb des Expertensystems, die letztlich bestimmt, was erklärt werden kann.

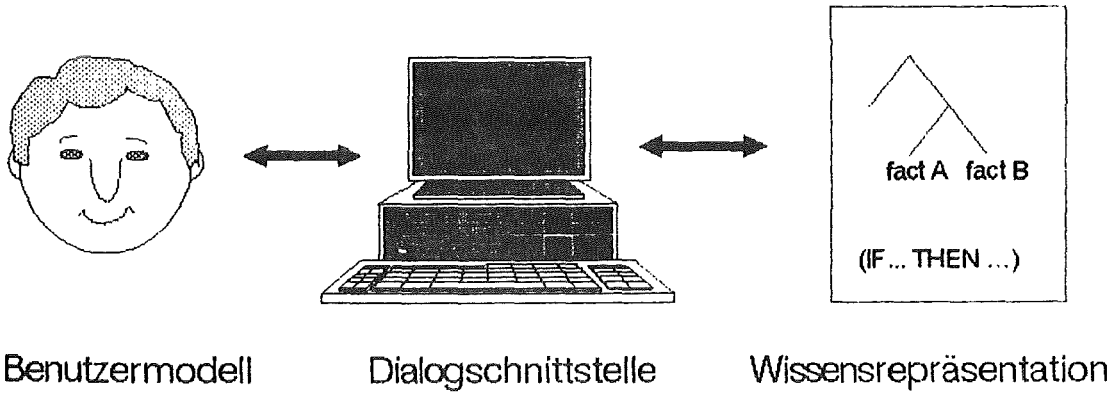


Abb. 3.2: Aspekte bei einem Expertensystem

Diese drei Punkte sollen in den folgenden Unterkapiteln detailliert daraufhin untersucht werden, welche Aspekte bei der Realisierung einer Erklärungskomponente beachten werden müssen.

3.3.1 Benutzermodell

In Kapitel 3.2 wurde mehrfach auf die Bedeutung der Akzeptanz eines DV-Systems durch den Anwender hingewiesen. Um eine hohe Akzeptanz zu erreichen, muß zu Beginn des Projektes ermittelt werden, welche Anforderungen die künftigen Benutzer an das System stellen. Hierfür sollte ein Benutzerprofil definiert werden, das unter anderem beinhaltet, welche Benutzertypen am System arbeiten, welche Vorkenntnisse sie haben und welche Informationen sie zu einer Erklärung benötigen. Die einzelnen Aspekte eines solchen Benutzerprofils sollten die folgenden Punkte berücksichtigen:

- *Definition von Benutzerklassen*

Bevor über Details des Benutzerprofils entschieden werden kann, muß herausgefunden werden, ob und welche Klassen von Benutzern mit dem zu installierenden System arbeiten. Der Begriff der Benutzerklasse soll im folgenden nicht im Sinne einer Zuordnung zu Personen entsprechen, sondern soll im Sinne von Rollen gesehen werden, die der Benutzer dem System gegenüber einnimmt. Es sollte dem Benutzer möglich sein, seine Rolle

gegenüber dem System, also die Zuordnung zu einer Benutzerklasse, zu ändern.
Im einfachsten Fall gehören alle Benutzer zu einer Klasse und es ist keine Differenzierung notwendig. Betrachtet man jedoch speziell die Erklärungskomponente eines Expertensystems, so sollten von Anfang an zumindest die Klassen "Anwender" und "Wissensingenieur" berücksichtigt werden. Diese beiden Klassen stellen bezüglich der Quantität und Qualität der erwarteteten Informationen sehr unterschiedliche Anforderungen an eine Erklärungskomponente.
Hat man eine erste Klassifizierung gefunden, so sollten die nun folgenden Punkte des Benutzerprofils für jede Klasse extra betrachtet werden, um eine anwenderfreundliche Implementierung zu erreichen.

- *Vorkenntnisse des Benutzers*

Grundsätzlich wird durch die Einordnung von Benutzern in Benutzerklassen eine erste Berücksichtigung von Vorwissen erreicht, jedoch ist diese Einteilung noch sehr grob. Um eine echte Anpassung an den individuellen Benutzer zu ermöglichen, sollte der Kennntnisstand der Benutzer möglichst genau dokumentiert werden. Dabei sind zwei Aspekte zu betrachten.
Der erste ist der fachspezifische Aspekt, der beinhaltet, inwieweit der Anwender auch Experte im Anwendungsbereich ist. Dabei sollten Grund- und Spezialwissen, spezielle Methodiken, aber auch Fachbegriffe und typische Abkürzungen erfaßt werden. Der zweite Aspekt ist der DV-spezifische, der berücksichtigen sollte, inwieweit DV-Kenntnisse, wie Umgang mit Terminals, Tastatur und Software-Werkzeugen oder auch Programmiererfahrungen vorhanden sind. Ausgehend von den vorhandenen Vorkenntnissen der Systembenutzer kann dann versucht werden, mithilfe der Erklärungskomponente dem jeweiligen Benutzer die für ihn relevanten Informationen, aufbereitet mit der passenden Terminologie, zur Verfügung zu stellen.
Eine etwas allgemeinere Anforderung, die sich nicht explizit auf das Benutzervorwissen bezieht, ist hier die Forderung nach einer gemeinsamen Fachterminologie bei Anwender, Experte und Systementwickler. Nur so können langfristig größere Mißverständnisse über die späteren Funktionen des Programmsystems vermieden oder zumindest gemildert werden.

- *Klassifizierung der möglichen Fragen von Benutzern an das System*

Natürlich hängt der Inhalt der Fragen, die ein Benutzer an ein System stellt, und der Antworten, die das System liefern soll, primär vom Anwendungsgebiet ab. Daher sollte für jedes Anwendungsgebiet ein Fragenkatalog (evtl. aus der Systemspezifikation ableitbar) erstellt werden, der es dem Systementwickler ermöglicht, die Benutzerfragen im Bereich der Erklärungskomponente zu berücksichtigen. Die vielfältigen Frageformen sollten dazu in einzelne Gruppen unterteilt und strukturiert werden. Die Strukturierung sollte auf die Bildung verschiedener Frageklassen hinauslaufen, wobei als erste Orientierung die allgemeine Klassifikation von [Hughes-86] dienen kann. Hughes hat versucht eine solche Klassifikation im Hinblick auf wissensbasierte Systeme zu erarbeiten (Abb. 3.3). Sie hat dabei festgestellt, daß sich jede Frage in ein Fragekonzept (entspricht den Umständen der Frage, z.B. auf wen sich die Frage bezieht) und in eine Fragekategorie (entspricht der erfragten Information, wie Ort, Subjekt etc.) zerlegen läßt.

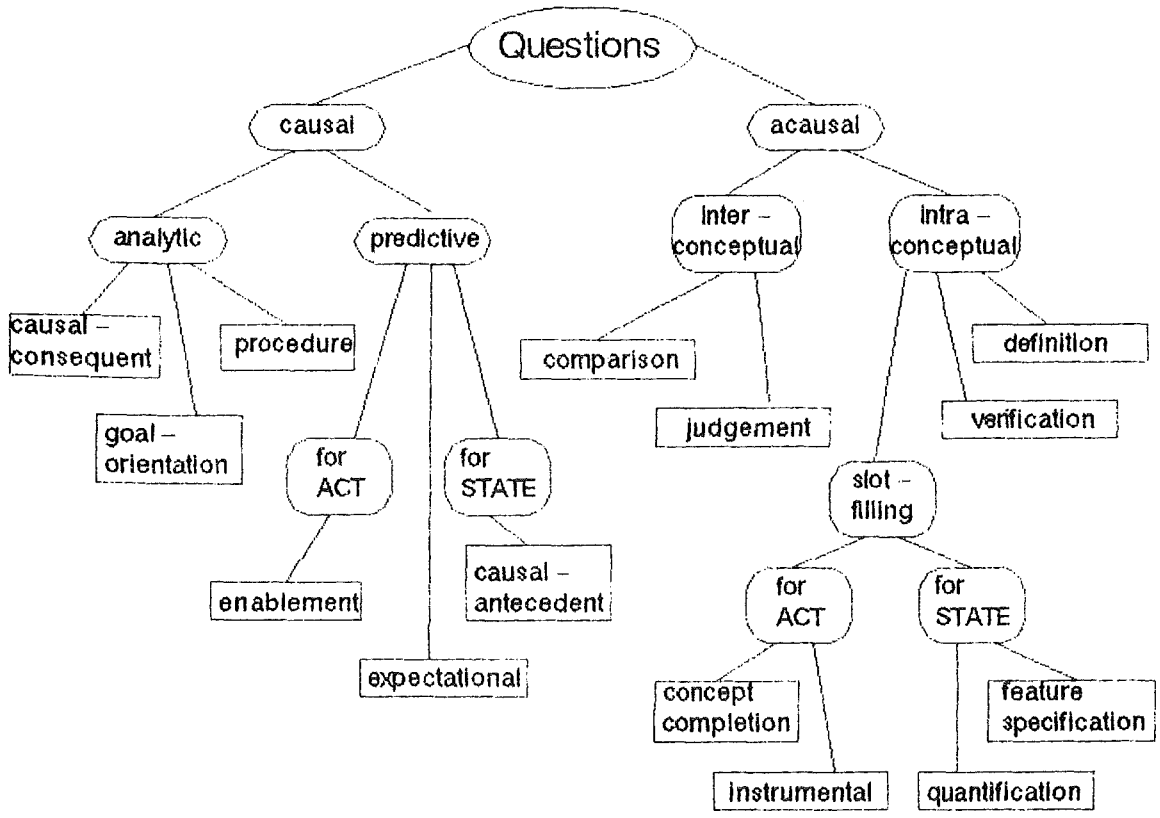


Abb. 3.3: Fragenklassifikation nach [Hughes-86]

Zur Erläuterung der Klassifikation sollen kurz die klassischen Warum- und Wie-Fragen von MYCIN in das Schema eingeordnet werden. Da sich die Warum-Frage auf die Prämisse bezieht, die das Expertensystem mithilfe eines 'backward-goals' gerade herzuleiten versucht, wird diese Frage der Klasse 'goal-oriented' zugeordnet. Bei der Frage "Wie ist eine Schlußfolgerung hergeleitet worden?" geht es um die Fakten, die bisher erfüllt wurden, und Hughes ordnete diesen Fragetyp in die Klasse 'causal-antecedent' oder 'enablement' ein. Hughes weist noch darauf hin, daß nicht jede Frage eindeutig eingeordnet werden kann, da beispielsweise der erfragte Detaillierungsgrad nicht zwingend aus der Frage hervorgeht. (Stellt der Benutzer die Frage: Wie wurde das Faktum hergeleitet?, so kann sich die Antwort auf die letzte Regel beziehen, aber auch auf den vollständigen Herleitungsbaum mit Basiswissen und Benutzereingaben.) Somit kann eine solche Klassifikation nur als Hilfsmittel dienen, um die mögliche Fragenvielfalt zu erfassen.
In der Praxis ergibt sich beispielsweise das Problem, daß der Anwender nicht nur nach dem 'Warum ...' fragt, sondern auch nach dem 'Warum nicht ...'. Dieser Fragetyp, der bei Hughes nicht explizit aufgeführt ist, führt zu erheblichen Problemen bei der Realisierung. Die Ursache liegt darin, daß Informationen (Fakten und Regeln), die zur Herleitung eines Ergebnisses verwendet wurden, problemlos protokolliert werden können, im Gegensatz zu Informationen, die nicht beim Problemlösungsprozeß eingesetzt wurden. Durch die Protokollierung können die benutzten Fakten nachträglich dem Benutzer als Erklärung oder Rechtfertigung angezeigt werden. Bei der 'Warum nicht'-Frage interessieren den Benutzer aber nur spezielle Informationen, wie beispielsweise 'Warum hat die Regel .. nicht gefeuert?'. Ein Versuch, dem Benutzer diese Möglichkeit der 'Warum nicht'-Fragen zu ermöglichen, wird in [Ladwig, Mellis-87] dargestellt. Es wird beschrieben, wie im Rahmen des Expertensystem-Werkzeuges TWAICE der Firma Nixdorf Computer AG, die Beantwortung bestimmter "negativer Fragen" realisiert wurde. Zu beachten ist dabei aber, daß sich der Ansatz von Ladwig und Mellis nur auf Regelsysteme, die keine Variablen beinhalten, übertragen läßt.

- Detailierungsgrad der erwarteten Information

Im vorigen Abschnitt wurde schon darauf hingewiesen, daß der Detaillierungsgrad einer erwarteten Information durch die Fragestellung nicht eindeutig bestimmt ist. Deshalb sollte der Detaillierungsgrad von vorneherein den Benutzerklassen zugeordnet werden, oder aber der Benutzer kann dynamisch bestimmen, wie genau die zu liefernden Informationen sein sollen. Im einfachsten Fall wird jeder Benutzerklasse ein fester Detaillierungsgrad zugeordnet; zum Beispiel werden bei einer Erklärung Fakten für den Wissensingenieur in der Systemdarstellung und für den Anwender in natürlicher Sprache ausgegeben.

Ein Verfahren zur Generierung verständlicher Erklärungen für den Benutzer wird in [Wallis, Shortliffe-82] beschrieben:

Jedem Konzept der Wissensbasis, das sind Regeln und Fakten, wird ein Bedeutungswert und ein Komplexitätswert zugeordnet, wobei die Werte natürliche Zahlen zwischen 1 (= niedrig) und 10 (= hoch) sind. Auch der Benutzer gibt bei der Arbeit mit dem System für seinen Wissensstand ('skill-level') einen Wert zwischen 1 und 10 an. Bei der Anforderung einer Erklärung durch den Benutzer werden dann nur diejenigen Konzepte ausgegeben, deren Komplexitätswerte in einem Intervall liegen, dessen Untergrenze der aktuelle Wissensstand des Benutzers ist. Die Obergrenze wird durch eine Variable bestimmt, die standardmäßig die Intervallbreite auf den Wert 2 setzt, aber durch mehrmaliges Nachfragen des Benutzers kurzzeitig erhöht werden kann. Auf der Abbildung 3.4 wird als Beispiel eine Ableitungskette von Fakt A nach Fakt F dargestellt. Der Begriff 'expertise' entspricht dem Wissensstand des Benutzers, während 'detail' die Breite des anzuzeigenden Intervalls angibt. Der Benutzer erhält in dem Beipiel also nur die Ableitungskette A -> C -> D zur Rechtfertigung der Aussage F angezeigt, da die Konzepte B und E als zu komplex eingestuft werden. Durch eine Kombination der Komplexitätswerte und der Bedeutungswerte kann ein mächtiges Werkzeug zur Erzeugung benutzeradäquater Erklärungen realisiert werden.

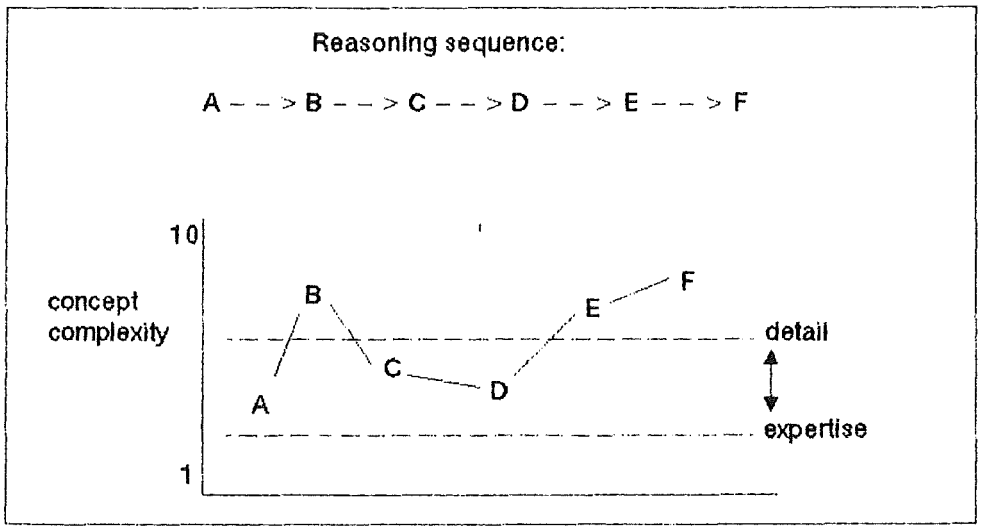


Abb. 3.4: Berücksichtigung von Komplexitätswerten in einer Ableitungskette aus [Wallis, Shortliffe-82]

Das Verfahren von Wallis und Shortliffe bezieht sich ausschließlich auf das 'Was wird angezeigt'. Die Problemstellung 'Wie wird angezeigt' wird in Kapitel 3.3.2 bei der Ausgestaltung der Dialogschnittstelle genauer diskutiert, obwohl auch dies in entferntem Sinne einer Detaillierung entspricht.

- Zeitpunkt der Fragen

Bei der Betrachtung der Zeitpunkte, zu denen der Benutzer Fragen an das System stellt, ergeben sich zwei Möglichkeiten:

1) Die meisten Expertensysteme, die mit rückwärtsverketteten Regeln arbeiten, also zielorientiert ('goal-driven'), erfragen während des Ablaufs mehrmals Informationen vom Benutzer. Dies trifft für die meisten Diagnostik-Expertensysteme zu (siehe auch MYCIN), da hier am Anfang noch nicht klar ist, welche Informationen erforderlich sind. Soll beispielsweise eine Hypothese bestätigt werden, so sind unter Umständen neue Informationen notwendig (zum Beispiel über Allergien etc.). Würden alle diese Informationen bei Beginn einer Sitzung erfragt, so würde die Effizienz und Handlichkeit des Systems stark absinken, da ein Großteil der Informationen

im jeweiligen Spezialfall überflüssig ist. Durch diese Interaktion hat der Benutzer die Möglichkeit, immer dann Fragen an das System zu stellen, wenn das System neue Informationen benötigt. Er kann also während des Ablaufs Zwischeninformationen abfragen.

2) Andere Expertensysteme, wie zum Beispiel XUMA, die eher als Interpretations-Expertensysteme bezeichnet werden, leiten jedoch mithilfe von vorwärtsverketteten Regeln und durch vorher vom Benutzer eingegebene Informationen (in XUMA die Analysenergebnisse) ihre Ergebnisse her, ohne während des Ableitungsprozesses anzuhalten. Der Benutzer hat dadurch keine Möglichkeit, den Ablauf des Systems zu beeinflussen, oder zusätzliche Zwischeninformationen zu erhalten. Er kann damit nur nach dem Ablauf eines Schlußfolgerungsprozesses einzelne Informationen vom System erfragen.

3.4.2 Dialogschnittstelle

Wie schon mehrmals angesprochen wurde, spielt die Akzeptanz eines Systems durch den Anwender eine wesentliche Rolle. Ein wichtiges Kriterium ist daher die Bedienungsoberfläche, d.h. die Dialogschnittstelle des Systems. Die Dialogkomponente hat somit die Aufgabe, die Eingaben des Benutzers aufzunehmen und die Ausgaben des DV-Systems verständlich darzustellen .
Die allgemeinen Anforderungen an Dialogschnittstellen von klassischen DV-Systemen können direkt auf wissensbasierte Systeme übertragen werden. Puppe fordert zum Beispiel von einer Dialogschnittstelle eine hohe Bedienungsgeschwindigkeit, eine geringe Einarbeitungszeit und eine hohe Fehlertoleranz [Puppe-88]. Im Bereich der wissensbasierten Systeme bieten sich, aufgrund der meist leistungsstarken Rechner, zusätzlich zu den herkömmlichen Dialog-Formen neue Techniken, wie 'pop-up-menus' oder maus-sensitive Fenster an.
Bei der Entwicklung der Dialogschnittstelle sollte bei der Eingabe- und Ausgabeschnittstelle für den Benutzer ein einheitliches Konzept realisiert werden, da nur eine konsistente Kombination sinnvoll ist.

Bei der Eingabeschnittstelle gibt es neben vielen Mischformen die folgenden Möglichkeiten:

Kommandosprache
Menü- und Formulartechnik
Natürlichsprachliche Eingabe

Bei der Ausgabeschnittstelle sind möglich:

Menüs, Tabellen etc.
Natürlichsprachliche Ausgabe
Grafische Darstellung

Im folgenden werden die verschiedenen Möglichkeiten zur Ein- und Ausgabe jeweils gemeinsam besprochen, soweit dies sinnvoll erscheint.

Die Vor- und Nachteile von Kommandosprachen und von Menütechniken sind schon häufig diskutiert worden [Schmitt-83, Kap. 3], daher soll hier nicht weiter darauf eingegangen werden. Der Aspekt der natürlichsprachlichen Schnittstelle bedarf jedoch einer genaueren Betrachtung, wobei zwischen der Ein- und Ausgabe unterschieden werden muß. Puppe weist beipielsweise daraufhin, daß bei Eingaben in natürlicher Sprache die Bedienungsgeschwindigkeit durch das notwendige Eintippen gering ist [Puppe-88]. Außerdem fällt es in diesem Fall dem Benutzer schwer, sich von dem System ein Bild zu machen, das heißt, er muß unter Umständen durch Probieren herausfinden, was das System tatsächlich versteht. Im Gegensatz dazu ist die natürlichsprachliche Ausgabe relativ einfach zu erreichen und sehr effektiv. Savory führt die natürlichsprachliche Ausgabe sogar als wesentliches Hilfsmittel zur Erklärung von Ergebnissen in Expertensystemen an [Savory-86]. Die Komplexität hängt dabei auch von der zugrundegelegten Sprache ab, wobei Savory darauf hinweist, daß es Systeme mit deutschsprachiger Ausgabe gibt, beispielsweise die Expertensystem-Shell TWAICE der Firma Nixdorf AG. Wie groß der Aufwand für die natürlichsprachliche Ausgabe ist, hängt im wesentlichen auch davon ab, wie unterschiedlich die auszugebenden Informationen sind, und ob sie sich dynamisch ändern. Müssen zu bestimmten Konzepten nur vorbereitete Texte ausgegeben werden, so ist kein zusätzlicher Aufwand nötig. Geht es aber beispielsweise darum, aus der formalen Definition einer Regel in Verbindung mit dem aktuellen Systemzustand einen Ausgabe-Text zu

generieren, so müssen auch spezielle linguistische Eigenheiten der Sprache (Artikel, Endungen etc.) berücksichtigt werden.
Bei den Möglichkeiten zur Ausgabe von Informationen durch den Rechner spielt neben den oben genannten Formen noch die grafische Ausgabe eine wichtige Rolle. Im Bereich der wissensbasierten Systeme wird beispielsweise die grafische Darstellung von Wissensbasen, die oft netzartige Formen haben, oder von Ableitungsketten, die baumartig strukturiert sind, häufig eingesetzt [Rahmstorf-87]. Mit den oben angesprochenen maus-sensitiven Fenstern ist es bei einigen Systemen möglich, Grafiken zu erzeugen, deren einzelne Objekte maus-sensitiv sind.

Eine optimale Dialogschnittstelle ergibt sich durch eine wirkungvolle Synthese aus den verschiedenen Ein- und Ausgabemöglichkeiten. Die einzelnen Verfahren können unter Umständen durch Maus-Einsatz und grafische Symbole sehr wirkungsvoll verbessert werden.

3.3.3 Wissensrepräsentation

Die Grundlage jeder Erklärung bildet das Wissen, das im Expertensystem vorhanden ist. Dieses Wissen kann in deklarativem und prozeduralem Wissen sowohl explizit als auch implizit enthalten sein. Sämtliches Wissen, das der Wissensingenieur bei der Konstruktion der Wissensbasis implizit miteingebracht hat, kann nicht zur Rechtfertigung oder Erklärung ausgegeben werden. Es ist also grundsätzlich bei Erklärungskomponenten zu beachten, daß sie nur das Wissen erklären können, das in dem Expertensystem explizit dargestellt wird.

Bei der Abbildung des Expertenwissens in die Wissensbasis des Expertensystems sind verschiedene Aspekte zu betrachten, beispielsweise die Arten von Wissen und wie sie im Expertensystem dargestellt werden können. Aus Gründen der Übersichtlichkeit werden diese Aspekte im folgenden getrennt erörtert, wobei in der Praxis die einzelnen Punkte in einem geschlossenen Konzept zusammengefaßt werden sollten.

- Arten von Wissen
Der erste Schritt bei der Wissensakquisition sollte die Abbildung des Expertenwissens in die verschiedenen Arten von Wissen sein. Durch eine

Unterteilung in die Wissensarten erreicht man eine erste Strukturierung, die bei der späteren Abbildung in die Wissensrepräsentationsformen des Expertensystem-Werkzeuges hilfreich ist.
Eine mögliche Einteilung bietet das KADS-Modell (Knowledge Aquisition Design Structure) von [Wielinga, Breuker-86] (Abb. 3.5). Das Wissen wird hier in vier hierarchisch angeordnete Ebenen eingeteilt. Der 'domain level' bildet die unterste Ebene und entspricht dem statischen Wissen innerhalb des Expertensystems. Auf dieser Ebene können Konzepte wie Frames, Relationen oder andere statische, d.h. nicht dynamisch veränderbare, Strukturen dargestellt werden. Die zweite Ebene, der 'inference level', beschreibt die Inferenzen, die direkt auf dem 'domain level' basieren. Dabei beschreiben die 'meta classes' die Rollen im Problemlösungsprozeß der Konzepte auf unterster Ebene, während die 'knowledge sources' Inferenzen beschreiben, die auf den Relationen der untersten Ebene basieren (Generalisation, Verfeinerung u.a.). Auf dem 'task level' wird dann versucht, die einzelnen 'knowledge sources' so zu kombinieren, daß ein gestelltes Ziel erreicht oder eine Aufgabe erledigt werden kann. Die oberste Ebene, der 'strategic level' verwaltet die 'tasks' der darunterliegenden Ebene, indem deren Anwendung geplant, aktiviert und überwacht wird. Da viele Expertensysteme nach einer einzigen festen Strategie arbeiten, besteht diese Ebene meist nur aus dem Aufruf einer 'task'.

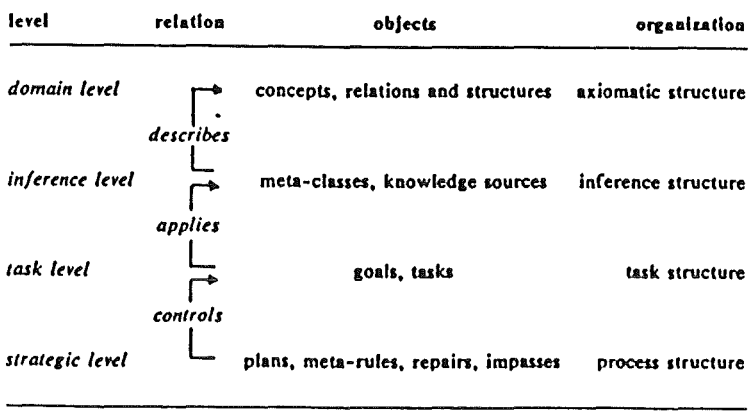


Abb. 3.5: Das KADS-Modell von [Wielinga, Breuker-86]

Eine Abbildung von Wissen in diese vier Kategorien ist jedoch nicht eindeutig. Je nach Anwendungsgebiet und Experte können über die Einteilung des

Wissens verschiedene Auffassungen bestehen. Ein Beipiel für diese Problematik wird in [Voß et al.-87] beschrieben, wo versucht wurde, eine vorhandene Wissensbasis mithilfe des oben gezeigten KADS-Modells neu zu strukturieren.

Einen anderen Ansatz zur Unterteilung des Wissens wählten Swartout und Smoliar bei der Entwicklung ihres EES-I-Systems (Abb. 3.6). Wie schon in Kapitel 3.1 beschrieben, wurde im EES-I-System (inzwischen existiert eine erweiterte Version EES-II) aus einer expliziten Darstellung des Wissens in der Wissensbasis mithilfe eines Programmgenerators das Expertensystem erzeugt [Swartout, Smoliar-87]. Hierzu wurde die Wissensbasis in problembeschreibendes (deklaratives), problemlösendes und terminologisches Wissen aufgeteilt. Das deklarative Wissen sollte dabei dem Fachwissen entsprechen, das in den klassischen Fachbüchern beschrieben wird. Es sollte die Objekte des Fachbereiches und deren Beziehungen untereinander beinhalten. Bei dem terminologischen Wissen handelte es sich um Begriffe und Konzepte des Fachbereichs, die von Experten bei der Kommunikation eingesetzt werden. In dem EES-I-System wurde dazu ein Lexikon zur Erklärung einzelner Begriffe gehalten und Definitionen in prädikatenlogischer Form dargestellt, um Abhängigkeiten von Begriffen zu beschreiben. Der problemlösende Bereich enthielt das Wissen zur Verarbeitung des problembeschreibenden Bereiches. Im wesentlichen waren das Pläne, die mithilfe festgelegter Methoden bestimmte Ziele erreichen konnten.
Beim Vergleich mit dem KADS-Modell fällt auf, daß die oberen drei Ebenen des KADS-Modells bei Swartout und Smoliar zwar in einer Kategorie (problemlösendes Wissen) zusammengefaßt wurden, daß aber sonst einige Gemeinsamkeiten vorhanden sind. Dabei entspricht in etwa der 'strategic level' den Plänen, der 'task level' den zu erreichenden Zielen und der 'inference level' den vorhandenen Methoden.

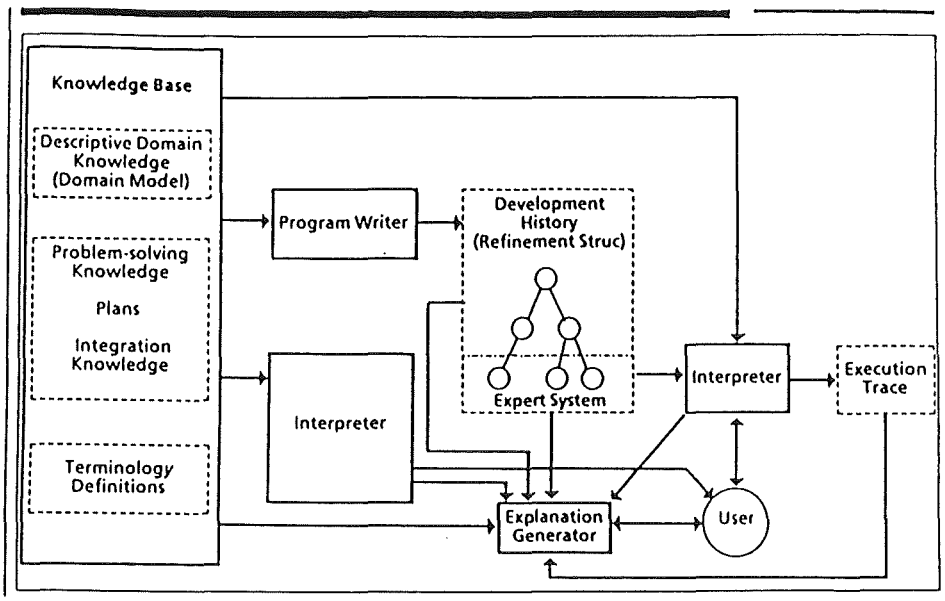


Abb. 3.6: Architektur des EES-I-Systems von [Swartout, Smoliar-87]

Selbst wenn es beim Entwurf des Expertensystems nicht möglich ist, eine der oben gezeigten Zerlegungen zu formulieren, so sollte zumindest versucht werden, eine explizite Darstellung der Problemlösungsmethoden zu erreichen. Dabei sollten fachspezifische Strategien, wie 'Hypothesize and Test' in der Diagnostik, und allgemeinere Methoden, wie 'Top Down', auch als solche erkennbar sein. Es wird dann unter Umständen möglich, Expertensystem-Werkzeuge zu implementieren, die solche Strategien zur Verfügung stellen und durch Austausch der statischen Wissensbasis in verschiedenen Bereichen eingesetzt werden können.

- Wissensrepräsentation auf logischer Ebene

Nach einer Unterteilung des Wissens in die oben angedeuteten Kategorien besteht als nächstes die Aufgabe, dieses Wissen auf die Konzepte des Expertensystem-Werkzeuges oder der Programmiersprache abzubilden. In diesem Abschnitt sollen dabei die Probleme einer Erklärungskomponente betrachtet werden, die sich bei der Verwendung eines hybriden Werkzeuges (ART, etc.) ergeben. Die hybriden Expertensystem-Werkzeuge bieten im allgemeinen als Möglichkeiten zur Wissensrepräsentation Frames, Regeln und Prozeduren in der zugrundeliegenden Programmiersprache an. Ein fertiges

System wird dann aus einer Synthese dieser Repräsentationsformen bestehen, wobei das jeweilige Wissen in einer problemadäquaten Weise dargestellt werden sollte.
Zur Erklärung von Objektstrukturen, also Frames, die über Relationen miteinander verbunden sind, werden meist vom Werkzeug entsprechende Routinen angeboten. Damit ist es möglich, Slot-Werte und Slot-Attribute auszulesen und über die definierten Relationen andere Objekte zu erreichen. Mithilfe dieser Zugriffsfunktionen kann dem Benutzer ein Hilfsmittel zur Verfügung gestellt werden, um die Objekte der Wissensbasis zu betrachten. Allerdings muß der Benutzer dazu die grundlegende Struktur des Systems kennen und wissen, in welcher Art und Weise die Objekte miteinander in Beziehung stehen.
Bei der Erklärung einzelner Regeln kann dem Benutzer, vergleichbar mit den Möglichkeiten bei Frames, der Regelinhalt angezeigt werden. Da dieser jedoch meist in einer speziellen syntaktischen Form implementiert wurde, muß vorher eine Aufbereitung erfolgen. Es muß zusätzliches Wissen eingefügt werden, um auch komplexe Regeln benutzerfreundlich erklären zu können (siehe nächster Abschnitt). Neben der Darstellung einzelner Regeln muß vor allem auch der Ableitungsprozeß erklärt werden. Dazu sollte der Benutzer die Möglichkeit haben, sich die Ableitungsketten, die zu einer Aussage geführt haben, schrittweise (lokal) oder im Zusammenhang (global) aufzeigen zu lassen. Dies erfolgt durch ein "Rückwärtslaufen" im Ableitungsbaum. Je feiner dann die im vorigen Abschnitt beschriebene explizite Abbildung der Lösungsverfahren erfolgt, desto besser wird das Verständnis des Benutzers für eine Ableitungskette. Unter Umständen sollte sich der Benutzer auch vorwärts durch Ableitungsketten bewegen können, damit er die Auswirkungen einer Aussage erkennen kann.
Bei der dritten Wissensrepräsentationsform, den Prozeduren, ergeben sich hinsichtlich einer Erklärung große Einschränkungen. Wie in Kapitel 3.2 schon beschrieben wurde, beschränkt sich die Erklärung von Programm-Modulen in der Regel auf die Modul- oder auch Funktionsebene, je nach der verwendeten Programmiersprache. Damit beschränkt sich die Erklärung einer Funktion auf eine statische Beschreibung der Abhängigkeiten zwischen den Eingabe- und den Ausgabeparametern, und der Seiteneffekte. Besonders die Seiteneffekte in der Wissensbasis sollten vermieden werden, da sie später von der Erklärungskomponente nicht mehr rekonstruiert werden können. Aus diesem Grund sollte im Sinne einer guten Erklärungs-

fähigkeit der Einsatz von programmiersprachlichen Modulen eingeschränkt werden

- *Zusätzliches Wissen zur Erklärung*

Werden Frames verwendet, um statische Objektstrukturen in der Wissensbasis abzubilden, dann reicht es in den meisten Fällen aus, die oben beschriebenen Zugriffsmöglichkeiten zur Verfügung zu stellen. Unter Umständen kann jedem Frame noch ein kurzer erklärender Text zugeordnet werden, der beispielsweise in einem Slot 'Kommentar' eingetragen wird.

Auch bei Funktionen und Modulen, die innerhalb des Expertensystems aufgerufen werden, lassen sich wegen der oben angedeuteten Problematik kaum erklärende Informationen, neben den üblichen Modulbeschreibungen, hinzufügen.

Bei Regeln ist für die Realisierung einer Erklärungskomponente im allgemeinen zusätzliches erklärendes Wissen notwendig, da die Regeln in den Prämissen und Aktionen oftmals sehr komplexe Zusammenhänge beschreiben, und da zudem meist noch implizites Wissen in den Regeln enthalten ist. Zur Erklärung der Zusammenhänge reicht oftmals eine Dokumentation zu jeder Regel aus, in der beschrieben wird, wo die Regel herkommt oder welche Hintergründe die Regel hat. In diesem Zusammenhang kann das angesprochene kausale Wissen eine Rolle spielen, da es zur Erklärung der Hintergründe eingesetzt werden kann. Das implizite Wissen kann dagegen nur dann erklärt werden, wenn es aus dem Regeltext explizit generiert werden kann. Ein Beispiel dafür ist in Abbildung 2.2 zu sehen. Aufgrund der festgelegten Bedeutung der Relation 'current-phase' kann die Prämisse '(current-phase init)' als Kontrollbedingung identifiziert und als solche gesondert behandelt werden.

Selbst wenn alle Informationen zur Regel explizit darin definiert sind, reicht es nicht aus, nur den Regeltext auszugeben, da die meisten regelbasierten Expertensystem-Werkzeuge eine spezielle Syntax vorschreiben (siehe auch Abb. 2.3). Um eine für den Benutzer akzeptable Erklärung zu erhalten, sollte daher der Regeltext vorher in eine natürlichsprachliche Form transformiert werden.

Eine weitere Unterstützung für den Benutzer ergibt sich, wenn der Begriff der Erklärung, wie im oben beschriebenen EES-I-System, etwas weiter gefaßt und sogar terminologisches Wissen explizit in der Wissensbasis

realisiert wird. Dann ist es möglich, dem Benutzer auch Erläuterungen zu Fachbegriffen oder gar ein kleines Lexikon zur Verfügung zu stellen.

- *Berücksichtigung von unsicherem Wissen*

Für die Entwicklung eines Expertensystems ist es ideal, wenn das zu behandelnde Fachgebiet vollständig und konsistent durch kategorische (im Gegensatz zu probabilistischen) Regeln beschrieben werden kann. Die große Anzahl von Einflußgrößen und deren Abhängigkeiten in der Realität macht es in der Praxis jedoch erforderlich, bei der Abbildung eines Problems in ein Modell gewisse Einschränkungen vorzunehmen. Der Systementwickler hat dann die Möglichkeit, sich nur auf einen sehr restriktiven Bereich zu beschränken, oder er muß in der Wissensbasis Unsicherheiten berücksichtigen. Die Berücksichtigung von Unsicherheiten geschieht heute in den meisten Fällen durch sogenannte Gewißheitsfaktoren ('certainty-factors'), die durch verschiedene Algorithmen bei einer Regelverkettung verrechnet werden können. Je nach Anwendungsgebiet können diese Faktoren als Wahrscheinlichkeiten, als Punktbewertungen oder ähnliches dargestellt werden.

Schon [Szolovits, Pauker-78] haben empfohlen, "so viel kategorisches Schließen wie möglich und so wenig probabilistisches Schließen wie nötig" zu verwenden. Es zeigt sich oftmals, daß die Verwendung von Gewißheitsfaktoren, vor allem bei der Rechtfertigung von Ergebnissen, zu Problemen führt. Diese Probleme sind unabhängig davon, welche Verfahren zur Verarbeitung der Gewißheitsfaktoren eingesetzt werden.

Eines der ersten Modelle dafür war das Bayes-Theorem, das für die Diagnostik entwickelt worden war [Charniak, McDermott-85, Kap. 8.2]. Als größte Nachteile im praktischen Einsatz erwiesen sich dabei die erforderliche Unabhängigkeit der einzelnen Diagnosen und das Fehlen von vollständigen Statistiken. Auch die sogenannte Dempster-Shaver-Theorie [Gordon, Shortliffe-85], die das Bayes-Theorem um Diagnosehierarchien erweitert hat, konnte dieses Verfahren nicht wesentlich verbessern. Auch andere Verrechnungsschemata, wie der MYCIN-Ansatz [Shortliffe, Buchanan-75] oder die Theorie der Fuzzy-Sets [Zadeh-75], konnten den kritischsten Aspekt nicht beseitigen. Dieser Kritikpunkt ist die Subjektivität, mit der der Experte die Gewißheitsfaktoren bestimmt. Wenn auch die Verwendung solcher Faktoren in einzelnen Regeln noch

gerechtfertigt erscheint, so scheint der Gewißheitsfaktor von Ergebnissen, die über lange Ableitungsketten ermittelt wurden, kaum noch akzeptabel.

Werden in einem Expertensystem trotzdem Gewißheitsfaktoren eingesetzt, so liegt das Hauptproblem bei der Erklärungskomponente darin zu entscheiden, inwieweit bestimmte Fakten zur Ableitung von Ergebnisssen beigetragen haben. Erste Möglichkeiten dazu wurden unter anderem von [Strat-87] und [Ganascia-85] vorgeschlagen. Strat versuchte in seiner Arbeit, mithilfe der Sensitivitätsanalyse die für das Ergebnis relevanten Informationen herauszufiltern. Der Ansatz von Ganascia bestand darin, mithilfe einer Umkehrfunktion diejenigen Regelprämissen auszugeben, die wesentlichen Anteil am Ergebnis hatten. Damit können zwar die Gewißheitsfaktoren in die Erklärung mit einbezogen werden, dennoch bleibt die Subjektivität erhalten.

Um solche Bewertungsfaktoren zu vermeiden, schlägt Puppe zur Berücksichtigung von unsicherem Wissen das nicht-monotone Schließen und den Ersatz heuristischer Regeln durch kausale Modelle vor [Puppe-88]. Beim nicht-monotonen Schließen können dann beispielsweise solche Konzepte wie die 'Viewpoints' in ART oder auch die 'Worlds' in KEE (Knowledge Engineering Environment von IntelliCorps) eingesetzt werden. Auch Regeln mit Ausnahmen, wie Puppe sie vorschlägt, können einen nicht-monotonen Schlußfolgerungsprozeß nachbilden. Da jedoch beim Einsatz von Regeln mit Ausnahmen die Gefahr besteht, daß die Menge der möglichen Ausnahmen sehr stark anwächst, verweisen Dahr und Pople darauf, daß dieses Verfahren nicht praktikabel ist [Dhar, Pople-87]. Sie versuchten daher, innerhalb eines Planungsproblems eine strukturorientierte Wissensrepräsentation statt einer regelbasierten einzusetzen.

4 Das Expertensystem XUMA

Der ständig wachsende Umfang an umweltrelevanten Informationen verlangt mehr und mehr den Einsatz moderner Informationsverarbeitungstechniken [Trauboth-87]. Neben den konventionellen Einsatzgebieten der DV im Umweltbereich, wie Datenbanken, Simulationsmodelle und Prozeßsteuerung, werden inzwischen auch Verfahren der künstlichen Intelligenz eingesetzt. Es handelt sich dabei fast ausschließlich um Expertensysteme, die in den Bereichen Interpretation, Planung, Vorhersage und Diagnose realisiert werden [Hushon-87]. Einige der einsetzbaren Expertensysteme befassen sich mit den Problemen bei der Bewertung von Böden, Proben oder Abfällen auf ihre Toxizität oder Wiederverwendbarkeit hin. Nach Hushon gibt es zwei Gründe dafür, warum die Umweltprobleme im Vergleich zu anderen Einsatzgebieten erst jetzt mit Expertensystem-Unterstützung gelöst werden. Zum einen werden für die Lösung von Problemen im Umweltbereich oftmals Experten unterschiedlicher Disziplinen benötigt (beispielsweise Chemiker und Geologen), während Wissensbasen von Expertensystemen bisher meist nur von einem Experten erarbeitet wurden. Zum anderen wurden bisher nur wenige heuristische Regeln in Anwendungsgebieten mit diffusem Wissen explizit formuliert, behauptet Hushon. Da aber eindeutige Lösungsalgorithmen nur selten vorliegen, bzw. eine mögliche Modellbildung aus Komplexitätsgründen oftmals nicht in Frage kommt, weichen immer mehr Entwickler auf regelbasierte Systeme aus.
Durch die notwendige Formalisierung des Wissens bei der Wissensakquisition wird der Experte zudem gezwungen, sein Wissen zu ordnen und zu strukturieren. Diese Strukturierung kann oftmals mithelfen, die Problemlösungsmethoden der einzelnen Fachexperten zu normieren, um willkürliche Entscheidungen unterschiedlicher Experten zu vermeiden. Da gerade im Umweltbereich die Entscheidungsverantwortung meist bei den Landes- und Bundesbehörden liegt, gewinnt der Aspekt der Normierung von Entscheidungen hier deutlich an Bedeutung.

Das Expertensystem XUMA, das im folgenden beschrieben wird, hat nicht nur eine unterstützende Aufgabe zu erfüllen, sondern dient auch der Vereinheitlichung von Entscheidungen. Wie in der Einleitung schon beschrieben wurde, soll XUMA bei der Altlastenbewertung durch die

Wasserwirtschaftsämter eingesetzt werden, und dabei als Unterstützung beim Bewertungsvorgang dienen. Im ersten Unterkapitel wird kurz das Vorgehen bei der Altlastenbewertung und die Einordnung von XUMA darin geschildert. Im darauffolgenden Kapitel werden dann der Aufbau und die Funktionen von XUMA beschrieben.

4.1 Einordnung von XUMA in die Altlastenbewertung

Für die Altlasten-Bewertung wurden im Juni 1987 Richtlinien vom Ministerium für Ernährung, Landwirtschaft, Umwelt und Forsten in Baden Württemberg ausgegeben [Umweltministerium-87]. Diese Richtlinien sollen dabei helfen, den Vorgang der Altlasten-Bewertung zu vereinheitlichen und zu objektivieren. Die Richtlinien geben dabei einen äußeren Rahmen für die Bewertung vor, wobei die eigentliche Bewertung nur von Experten vorgenommen wird. Die Richtlinien enthalten Angaben darüber, wie die Erkennung, Erhebung und Beschreibung des gefahrverdächtigen Standorts, sowie vor allem die Abschätzung der von ihm ausgehenden Gefahr für die Schutzgüter Boden, Luft, Oberflächenwasser und Grundwasser durchgeführt werden soll.
Der kritischste und wichtigste Vorgang dabei ist die Bewertung der Gefährlichkeit von Stoffen. Aufgrund von Analysenergebnissen und der Berücksichtigung verschiedener Randbedingungen wird die Stoffgefährlichkeit in Vergleichslage bestimmt. Die Vergleichslage wurde eingeführt, um in einem ersten Schritt von der Umgebung des untersuchten Abfalls zu abstrahieren, damit verschiedenartige Abfälle standortunabhängig verglichen werden können. Die Vergleichslage entspricht dabei einer idealisierten Umgebung des Abfalls und bezieht sich jeweils auf das untersuchte Schutzgut Boden, Wasser etc. (Abb. 4.1).

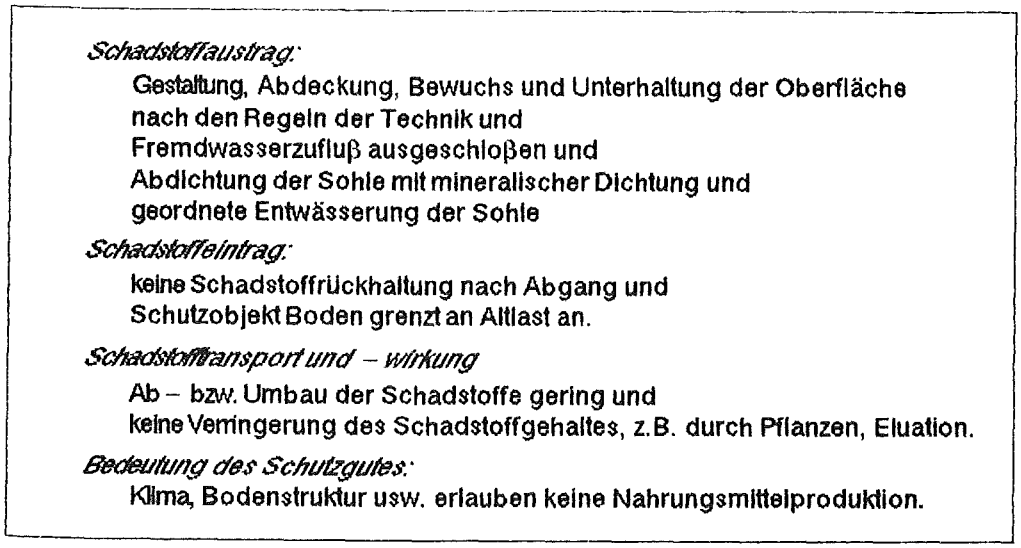


Abb. 4.1: Vergleichslage bei Schutzgut Boden aus [Umweltministerium-87]

Die sich daraus ergebende Stoffgefährlichkeit wird als Grundrisiko bezeichnet, und durch eine Zahl r0 dargestellt. Dieser r0-Wert liegt zwische 0 und 6 und soll durch XUMA ermittelt werden. Zur Berücksichtigung der örtlichen Verhältnisse wird dieser r0-Wert mit Zahlwerten multipliziert und damit vergrößert oder verkleinert. Dabei wird versucht, durch diese Multiplikatoren den Austrag des Schadstoffes aus dem Gefahrgut, den Eintrag ins Schutzgut, das Verhalten im Schutzgut und die Nutzung des Schutzgutes zu berücksichtigen. Die Multiplikatoren werden gewonnen durch den Vergleich der tatsächlichen örtlichen Verhältnisse mit der Vergleichslage. Das so errechnete Resultat wird als maßgebliches Risiko R bezeichnet und bestimmt mit dem Beweisniveau den Handlungsbedarf (Abb. 4.2).

Der Handlungsbedarf bedeutet dabei:

A: "Ausscheiden" aus der Altlastendatei (der Abfall ist unproblematisch);

B: "Belassen" in der Altlastendatei (beobachten);

C: "Fachtechnische Kontrolle" (ständige Kontrolle);

D: "Durchprüfen" von Möglichkeiten zur Gefahrenminderung;

Eij: "Erkundung", um von Beweisniveau i auf Beweisniveau j

zu gelangen.

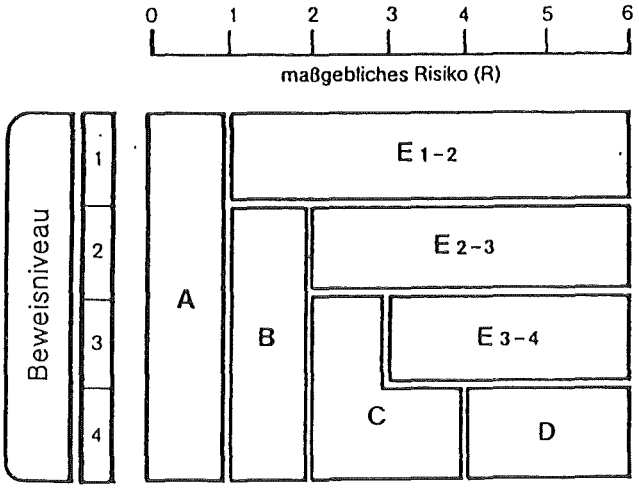


Abb. 4.2: Berechnung des Handlungsbedarfs abhängig vom Beweisniveau und dem maßgeblichen Risiko R aus [Umweltministerium-87]

Das sogenannte Beweisniveau entspricht dem Erkundungsstand, wobei die Kosten zur Erreichung eines höheren Beweisniveaus stark ansteigen. Daher ist es sinnvoll, erst ab einer gewissen Gefährlichkeit weitere Untersuchungen zu fordern. Ergibt sich beispielsweise bei einer ersten Erkundung E01 (= historische Erkundung) ein R-Wert von 1.1, so muß eine weitere Erkundung E12 durchgeführt werden. Ergibt diese Erkundung nur Werte zwischen 0 und 2, so kann der Handlungsbedarf jetzt festgelegt werden. Alle Rechnungen, die auf dem r0-Wert aufbauen, werden in einem speziellen Bewertungsbogen (Abb. 4.3) festgehalten.

G

Beispiel 3
Altablagerung — Hausmull
Gewasserbettverfullung — Wasserfassung

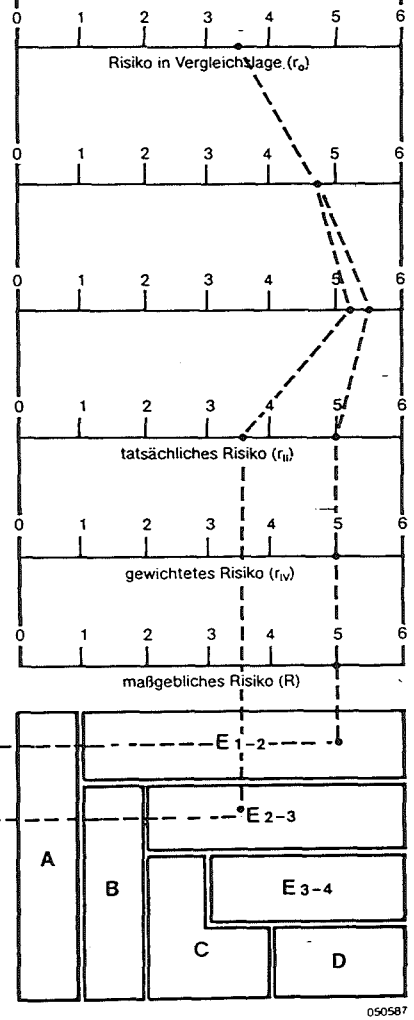
Bewertungsbogen

Raum fur statistische Daten, Prufungsvermer interne Angaben

Regenmaßschlüssel / PLZ / Gemeinde / Kennziffer / Bezeichnung

	1. Bewertung	2. Bewertung	3. Bewertung
	Bearbeiter / Datum: 05.03.1987	Bearbeiter / Datum: 15.05.1987	Bearbeiter / Datum
0. Stoffgefährlichkeit	Hausmull, Sperrmull, Gartenabfalle, Bauschutt, Filterstaub, Galvanikschlamme Fasser mit wassergefahrdenden Stoffen / Akte 1931 – 1969 / Akte bereits weitgehend ausgelaugt / Vermutung Grundflache ca. 10.000 m², Auffullhohe ca 2 m / Akte, Lageplan r_0 = 3,5	– ohne Änderung – r_0 = 3,5	r_0 = ...
I. Austrag	Abfalle liegen innerhalb der Grundwasserwechselzone, sie sind überwiegend eingestaut / Geologe keine Sohl- und Flankenabdichtung / Vermutung Abdeckung der Oberfläche mit humushaltigem Erdmaterial, landbauliche Nutzung, ebene Oberfläche / Begehung N = 650 mm m_I = 1,3 Δr_I = + 1,1 r_I = 4,6	– ohne Änderung – m_I = 1,3 Δr_I = +1,1 r_I = 4,6	m_I = ... Δr_I = ... r_I = ...
II. Eintrag	eine wasserungesättigte Zone ist den größten Teil des Jahres nicht vorhanden wegen hohem Grundwasserstand / Geologe m_{II} = 1,2 Δr_{II} = + 0,9 r_{II} = 5,5	– geänderte Einschätzung siehe unten – m_{II} = 1,1 Δr_{II} = + 0,5 r_{II} = 5,1	m_{II} = ... Δr_{II} = ... r_{II} = ...
III. Transport/ Wirkung	Fließgeschwindigkeit des Grundwassers ca. 0,5 m/d / Geologe geringe Abbau- und Sorptionseffekte sind zu erwarten / Vermutung m_{III} = 0,9 Δr_{III} = – 0,5 r_{III} = 5,0	Im Grundwasser unterstromig wurden hohe Zinkwerte und erhöhte organische Belastung festgestellt / chem. Analyse **Schlußfolgerung:** Abbau- u. Sorptionseffekte treten auf Schadstoffeintrag geringer als zunächst geschätzt m_{III} = 0,7 Δr_{III} = – 1,5 r_{III} = 3,6	m_{III} = ... Δr_{III} = ... r_{III} = ...
IV. Bedeutung	Im Zustrombereich zu einer Wasserfassung großes Wasserdargebot, geringe Schadstoffemission / Vermutung Fließzeit von Gefahrenherd zur Entnahmestelle rechnerisch ca. 3 Jahre / Vergleichsrechnung m_{IV} = 1,0 Δr_{IV} = ± 0 r_{IV} = 5,0	– ohne Änderung – m_{IV} = 1,0 Δr_{IV} = ± 0 r_{IV} = 3,6	m_{IV} = ... Δr_{IV} = ... r_{IV} = ...
maßgebl. Risiko	Regelfall Altlast R = 5,0	– ohne Änderung – R = 6	= .....
Bewels-niveau	Begehung durchgeführt, historische Erkundung abgeschlossen BN = 1 Handlungsbedarf: E 5,0 (1 – 2)	Durch chem.-physikalische Analyse des Grundwassers wurde eine (leichte) Kontamination des Grundwassers festgestellt BN = 2 Handlungsbedarf: E 3,6 (2 – 3)	BN = ... Handlungsbedarf
Maßnahmen	Untersuchung des Grundwassers an einer neu zu errichtenden Grundwassermeßstelle (geeignete Meßstellen sind noch nicht vorhanden)	Weitere Untersuchungen mit dem Ziel, Schadstoffahne einzugrenzen. Dazu: Einrichtung weiterer Grundwassermeßstellen.	

Bewertungsprofil



050587

Abb. 4.3 (vorige Seite): Beispielhafter Bewertungsbogen eines Fachexperten aus [Umweltministerium-87]

4.2 Aufbau von XUMA

Die Bewertung des Umweltrisikos von Abfallstoffen mithilfe des Expertensystems XUMA besteht im wesentlichen aus zwei Phasen. Die erste Phase bildet das Erstellen eines Analysenplans, in dem ausgehend von der betrachteten Branche oder von Stoffen, ein Analyseplan durch XUMA erstellt wird. Dieser gibt an, welche Analysenparameter untersucht werden sollen. Der Benutzer kann dann die entsprechenden Analysen von Labors durchführen lassen und dem Expertensystem die Ergebnisse der Analysen mitteilen. Ausgehend von den Analysenergebnissen bestimmt XUMA in der zweiten Phase mithilfe von Regeln die Umweltgefährlichkeit und schlägt Maßnahmen zur Bereinigung des untersuchten Gebietes vor.
Der aktuelle Prototyp beinhaltet derzeit für die Bewertung von Abfällen von Kohleveredelungsbetrieben etwa 40 Branchenkonzepte, 400 Stoffe und ca. 310 Analysenparameter.

1) Erstellung eines Analysenplans
Wie oben geschildert, wird in der ersten Phase mithilfe von XUMA ein Analysenplan erstellt, in dem enthalten ist, welche Analysenparameter in welchen Schutzgütern gemessen werden sollen. Die Analysenplanerstellung kann der Anwender auf verschiedene Weisen beginnen.

a) Ist beispielsweise der Industriebetrieb bekannt, der als letztes auf dem zu untersuchenden Gelände gearbeitet hat, so ist es sinnvoll über die Branche (z.B. Gaswerk) einzusteigen. Je genauer eine Probenahmestelle den ehemaligen Anlagenteilen oder Produktionsverfahren zugeordnet werden kann, um so exakter lassen sich die typischerweise vorhandenen Verunreinigungen bestimmen.

b) Wenn ein Einstieg über die Branche nicht möglich ist, jedoch mögliche Stoffe bekannt sind (z.B. Teer), dann kann der Anwender auf dem Gelände vorhandene Stoffe einem Analysenplan zuordnen. Er kann aus der Stoffliste von XUMA die entsprechenden Stoffe und Analysenparameter auswählen.

c) Liegen dem Anwender keinerlei Kenntnisse über Branchen, Anlagen oder Stoffe vor, so kann er von einem zweistufigen Standardplan ausgehen.

Nach Erstellung des Analysenplans durch XUMA kann der Benutzer beliebige Parameter hinzufügen oder streichen, um den Analysenplan seinen individuellen Erfordernissen anzupassen. Der Anwender kann dann die angegebenen Analysen durchführen lassen und die Ergebnisse dem Expertensystem XUMA mitteilen.

2) Bewertung der Stoffgefährlichkeit

Nachdem die Analysenergebnisse dem System eingegeben worden sind, kann mit der Bewertung begonnen werden.

Als erstes wird versucht, die einzelnen Parameterwerte, die quantitativ erfaßt wurden, anhand von Grenzwerttabellen (Klokeliste, Trinkwasserverordnung, etc.) auf qualitative Werte abzubilden. Es existieren heute etwa 100 verschiedene Tabellen mit jeweils bis zu 120 Parametern, die dabei in Frage kommen können. Die Relevanz der einzelnen Tabellen hängt dabei von der Art des Untersuchungsobjekts (Boden, Grundwasser etc.), der Größenordnug des Meßwerts und anderen Randbedingungen ab. Bei den qualitativen Werten handelt es sich um sogenannte Qualitätsklassen, die von Klasse I (natürliche Werte) bis zur Klasse VI (sehr hohes Gefährdungspotential) reichen. Die angesprochenen Grenzwerttabellen müssen dabei nach Prioritäten geordnet werden, da es möglich ist, daß ein Parameter über mehrere Tabellen in unterschiedliche Qualitätsklassen eingeordnet wird. Existiert jedoch keine Vergleichstabelle für einen Parameter, so versucht das System anhand von gespeicherten Altfällen, Ersatzparametern oder Benutzerangaben diesen Parameterwert einer Qualitätsklasse zuzuordnen. Neben den quantitativ gemessenen Parametern liegen auch viele qualitative Parameterwerte vor, wie etwa "Geruch: faulig" oder "Trübung: klar", die nach individuellen Verfahren verarbeitet werden müssen.

Die Bearbeitung der Bewertung erfolgt in XUMA ausschließlich mit Regeln, die bei der Wissensakquisition vom Experten definiert wurden (Abb. 4.4). Dabei werden qualitative Ergebnisse im Sinne einer Expertise ausgegeben und auch quantitative Ergebnisse, etwa r0, ermittelt. Die Ergebnisse von

einzelnen zusammengehörigen Analysen werden dann weiter zu Aussagen für eine Probe, und für einen ganzen Fall weiterverarbeitet, wobei auf der Fallebene der oben erwähnte r0-Wert abgeleitet wird.

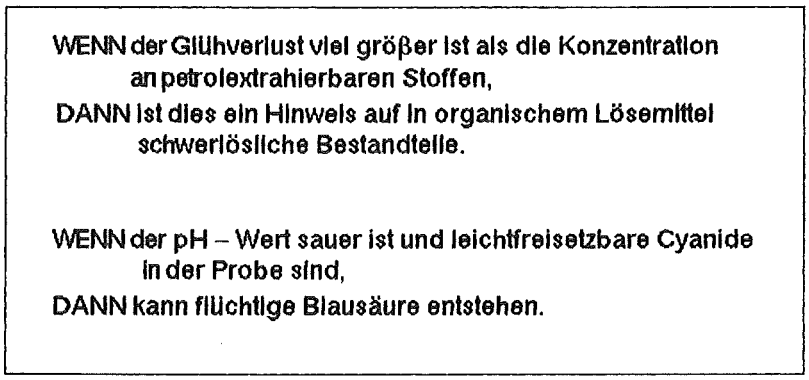


Abb. 4.4: Regeln für die Analysebewertung mit XUMA

5 Realisierung einer Erklärungskomponente für XUMA

Wie in dem vorigen Kapitel beschrieben, ist die wichtigste Aufgabe von XUMA die Bewertung der Stoffgefährlichkeit. Die zusätzlichen Funktionen, die XUMA anbietet (Analysenplanerstellung etc.), dienen im Prinzip nur zur Vorbereitung der Bewertung und zur Unterstützung des Sachbearbeiters bei der Bearbeitung eines Falles. Aus diesem Grund ist es die Aufgabe der Erklärungskomponente in XUMA, die ermittelten Bewertungsaussagen zu erklären und zu rechtfertigen. Welche Anforderungen daraus an die Erklärungskomponente von XUMA zu stellen sind, welche Konzepte zu deren Realisierung eingesetzt und wie diese Konzepte implementiert wurden, wird im folgenden erläutert.

5.1 Spezifikation der Anforderungen

Die Bedeutung einer Erklärungskomponente in einem Expertensystem beruht im wesentlichen auf den drei Aspekten, die in Kapitel 3.2 detailliert beschrieben wurden. Im folgenden soll kurz die Bedeutung dieser Aspekte für die Erklärungskomponente in XUMA diskutiert werden.

Akzeptanz durch den Anwender
Die Erklärungskomponente in XUMA dient der Rechtfertigung und Erklärung von Bewertungsaussagen. Diese Aussagen bilden die Basis für das weitere Vorgehen bei der Erkundung, Sicherung und Sanierung von Altlast-Standorten. Zum einen sind die aufgrund der Bewertungsergebnisse notwendigen Maßnahmen (beispielsweise die Beseitigung des Abfalls in eine Untertagedeponie) zum Teil mit sehr hohen Kosten verbunden. Zum anderen ist die Bewertung durch die hohe Anzahl von Daten (bis zu 1000 Meßwerte pro Fall) selbst für den Fachexperten nicht unmittelbar nachvollziehbar. Beide Aspekte zeigen, daß es nicht ausreicht, dem Anwender von XUMA nur die Bewertungsergebnisse auszugeben. Der Anwender, der im allgemeinen der zuständige Sachbearbeiter der LfU ist, muß in der Lage sein, die Schlußfolgerungsprozesse von XUMA zu überprüfen, und er erwartet daher vom Expertensystem eine Möglichkeit zur Rechtfertigung der Bewertungsergebnisse. Gelingt es nicht, dem Anwender die Abläufe innerhalb von XUMA transparent darzustellen, wird er, aufgrund der möglicherweise hohen Folgekosten, eher seinen eigenen

Erfahrungen vertrauen und trotz des u.U. geringeren Fachwissens als in XUMA das Expertensystem nicht weiter einsetzen.

Test und Validation der Wissensbasis
Bei den Aspekten Test und Validation der Wissensbasis müssen die zwei Sichtweisen des Systementwicklers und des Fachexperten unterschieden werden. Dem Systementwickler stehen in ART verschiedene Hilfsmittel zum Testen der Wissensbasis zur Verfügung, wie etwa das schrittweise Abarbeiten der Regeln. Diese Hilfsmittel genügen dem Systementwickler in den meisten Fällen, um die Wissensbasis zu testen, wobei ihn die Erklärungskomponente dabei unterstützen kann. Die Durchführung von Tests durch den Fachexperten, der sich mit den internen Abläufen des Expertensystems nicht auskennt, wird erst dann notwendig, wenn der Experte sein Wissen mithilfe einer Wissensakquisitionskomponente selbst eingeben kann. Diese Komponente ist bei XUMA in der Vorbereitung. Das Konzept der Erklärungskomponente muß somit berücksichtigen, daß der Benutzer beim Testen und Validieren der Wissensbasis unterstützt wird.

Nachvollziehbarkeit des zugrundeliegenden Wissens
Dieser dritte Punkt spielt in XUMA keine Rolle, da nicht daran gedacht wird, das Expertensystem zur Schulung einzelner Mitarbeiter einzusetzen. Wie bereits in Kapitel 3.1 geschildert, werden an eine Erklärungskomponente, die zur Schulung eingesetzt wird, besondere Anforderungen (beispielsweise bezüglich der verwendeten Terminologie) gestellt. Diese Anforderungen konnten bei der Realisierung der Erklärungskomponente von XUMA vernachlässigt werden.

Es kann festgestellt werden, daß die Erklärungskomponente in XUMA primär die Aufgabe hat, dem Anwender die Möglichkeit zu geben, sich die Bewertungsaussagen in benutzeradäquater Weise rechtfertigen zu lassen. Dabei sollte die Erklärungskomponente als eigenständiges Modul realisiert werden und in das Gesamtsystem integrierbar sein. Die einzelnen Anforderungen, die sich aus dieser Aufgabenstellung explizit ergeben haben, lauten:

- Erzeugung der Rechtfertigungen für einzelne Aussagen;

- Ausfiltern überflüssiger Informationen, wie Steuer- und Kontrollinformationen, abhängig von der Rolle des Benutzers gegenüber dem System;
- Textuelle Aufbereitung der Fakten und Regeln, die zur Rechtfertigung benötigt werden;
- Rechtfertigung einer Aussage durch den letzten Ableitungsschritt und durch die gesamte Schlußfolgerungskette;
- Integrierbarkeit der Erklärungskomponente in XUMA durch modularen Aufbau und die Definition einer Schnittstelle.

5.2 Abbildung der Anforderungen auf einzelne Konzepte

In diesem Kapitel sollen die einzelnen Konzepte zur Realisierung der Erklärungskomponente in XUMA aus den in Kapitel 3.3 beschriebenen Sichtweisen (Benutzermodell, Dialogschnittstelle und Wissensrepräsentation) vorgestellt werden.

5.2.1 Benutzermodell bei XUMA

Eine grobe Analyse zur Bestimmung der Benutzerklassen von XUMA führte zu den drei Klassen Anwender, Fachexperte und Systementwickler. Eine weitere Differenzierung innerhalb dieser Klassen ist für XUMA nicht notwendig. Diese Klassifizierung spiegelt auch das entsprechende Fachwissen der einzelnen Benutzergruppen, sowohl im fach- als auch im DV-spezifischen Bereich, wieder. Die Systementwickler, die zwangsläufig viel DV- und relativ wenig Fachwissen mitbringen, sind bezüglich der Erklärungskomponente an einer flexiblen Dialogschnittstelle mit Möglichkeiten zum Testen einer Wissensbasis interessiert. Anwender und Experte sind DV-Laien oder verfügen über DV-technisches Grundwissen, wobei sich dieses auf die Bedienung von DV-Systemen beschränkt. Tiefergehende Erfahrungen, wie etwa Programmierung etc., sind nicht vorhanden. Im fachspezifischen Bereich kann bei beiden Benutzerklassen von derselben Terminologie ausgegangen werden, wenn auch mit unterschiedlichem Hintergrundwissen. Damit kann die Gruppe der Anwender und die Gruppe der Experten bezüglich der Aufbereitung der Ausgabe gleich behandelt werden; es muß nur bezüglich des Detaillierungsgrads der Informationen unterschieden werden.

In XUMA werden diese drei Benutzerklassen durch die Zuordnung eines Relevanzwertes (ganze Zahl zwischen 0 und 5) zu jeder Benutzerklasse dargestellt. Zusätzlich wird jeder Regel und jeder Relation ein Relevanzwert bei der Definition der Wissensbasis statisch zugeordnet. Bei der Erklärung einer Aussage erhält der Benutzer dann nur diejenigen Fakten und Regeln angezeigt, deren Relevanzwerte seiner Benutzerklasse entsprechen oder darüberliegen. Wählt ein Benutzer die Klasse des Fachexperten (Relevanzwert = 3), so erhält er nur diejengen Aussagen und Regeln angezeigt, deren Relevanzwerte größer oder gleich 3 sind. Dieses Konzept entspricht in seinen Grundzügen dem in Kapitel 3.3.1 ausführlicher beschriebenen Verfahren von [Wallis, Shortliffe-82].
Da der Anwender sowohl an der Rechtfertigung einer Aussage interessiert ist, die den letzten Schritt genau beschreibt, als auch an einer überblicksartigen Darstellung des gesamten Schlußfolgerungsprozesses, kann der Benutzer bei XUMA eine lokale oder eine globale Erklärung auswählen.

Die lokale Erklärung zeigt ihm die Regel auf, die die angewählte Aussage hergeleitet hat und die dabei gültigen Prämissen. Dies entspricht einem Rückwärtsschritt im Ableitungsbaum. Bei einer globalen Erklärung wird dem Benutzer der Ableitungsbaum ausgegeben, damit er sich ein Bild von dem gesamten Schlußfolgerungsprozeß machen kann (siehe dazu auch Kapitel 5.2.2). Neben den Rechtfertigungen von Aussagen kann der Benutzer auch erfragen, welche Aussagen aus einer bestimmten Aussage abgeleitet wurden, was einem Vorwärtsschritt im Ableitungsbaum entspricht. Auch dies ist wieder lokal und global möglich.

5.2.2 Dialogschnittstelle der Erklärungskomponente von XUMA

Die Dialogschnittstelle der Erklärungskomponente wurde so aufgebaut, daß sie konsistent zum Gesamtsystem XUMA ist. Da XUMA für DV-Laien konzipiert wurde, bedeutet das, daß die Erklärungkomponente die hohen Anforderungen bezüglich Benutzerfreundlichkeit und Bedienkomfort von XUMA erfüllen muß. Hierzu gehören Hilfstexte zu jedem Fenster, eine fehlertolerante Eingabeschnittstelle und eine Bedienung mit möglichst wenig Tastatureingaben. Nach der Bewertung eines Falls, einer Probe oder einer Analyse werden dem Benutzer die Bewertungsaussagen maus-sensitiv ausgegeben (Abb. 5.1). Durch Anklicken einer Aussage wird die Erklärungs-

komponente aufgerufen, mit der sich der Benutzer Rechtfertigungen ausgeben lassen kann. Bedient wird die Erklärungskomponente mithilfe maus-sensitiver Menüs, also ohne Tastatureingaben. Die einzelnen Menüs werden in den beiden größeren Fenster ausgegeben, wobei das untere Fenster (Ausgabefenster) zur Ausgabe der Erklärungen und das obere Fenster (Eingabefenster) zur Ausgabe der Auswahlmenüs dient.

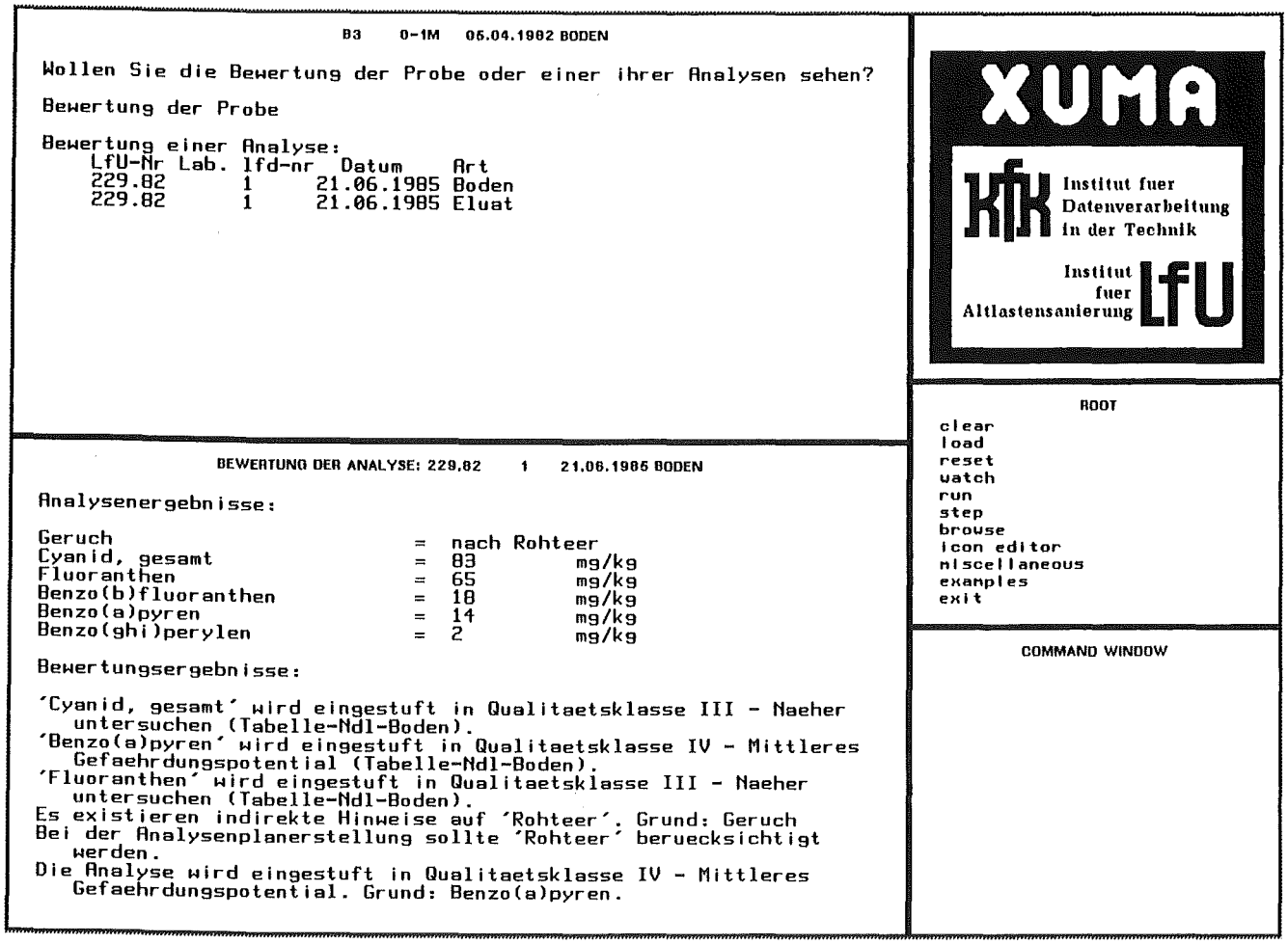


Abb. 5.1: Bildschirmaufbau bei XUMA

Das Anklicken einer Aussage bewirkt, daß dem Benutzer ein Auswahlmenü zur Verfügung gestellt wird, aus dem er eine Erklärungsform auswählen kann (Abb. 5.2). Er kann zur Rechtfertigung einer Aussage eine lokale oder eine globale Rechtfertigung anfordern oder einen anderen Menüpunkt durch Anklicken auswählen. Der schwarze Balken in der Abbildung bedeutet, daß

sich die Maus über der maus-sensitiven Kommandozeile "Ende der Erklärung" befindet. Betätigt der Benutzer jetzt den linken Mausknopf, so wird dieses Kommando aufgerufen und die Erklärungskomponente verlassen.

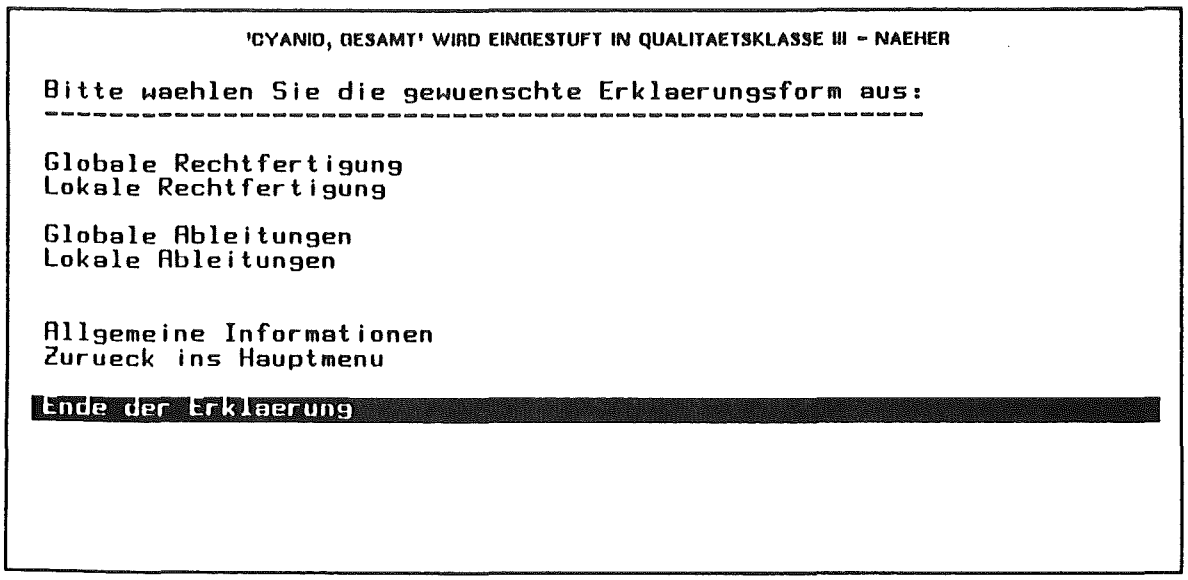


Abb. 5.2: Auswahlmenü zur Rechtfertigung einer Aussage

Wählt der Benutzer die globale Rechtfertigung, so wird ihm der gesamte Ableitungsbaum (wobei nicht relevante Informationen weggelassen wurden) in einer pseudografischen Form dargestellt (Abb. 5.3). Bei dieser Darstellungsform werden die Regeln und Prämissen der Ableitungstiefe entsprechend eingerückt. Diese Ausgabeform wurde gewählt, um dem Benutzer in einer kompakten und übersichtlichen Form die Informationen zur Verfügung zu stellen, die für ihn von Interesse sind. Bei einer grafischen Darstellung des Ableitungsbaumes mit Knoten und Kanten wäre es nicht möglich gewesen, die einzelnen Bewertungsaussagen in einer übersichtlichen Form in die Grafik zu integrieren.

Im Unterschied zur globalen Erklärung, bei der nur die Regeln und Prämissen angezeigt werden, sind bei der lokalen Erklärung noch weitere Kommentare eingefügt, um die Lesbarkeit zu erhöhen (Abb. 5.4). Neben einleitenden Sätzen wird dem Anwender zur Erläuterung noch der Regel-

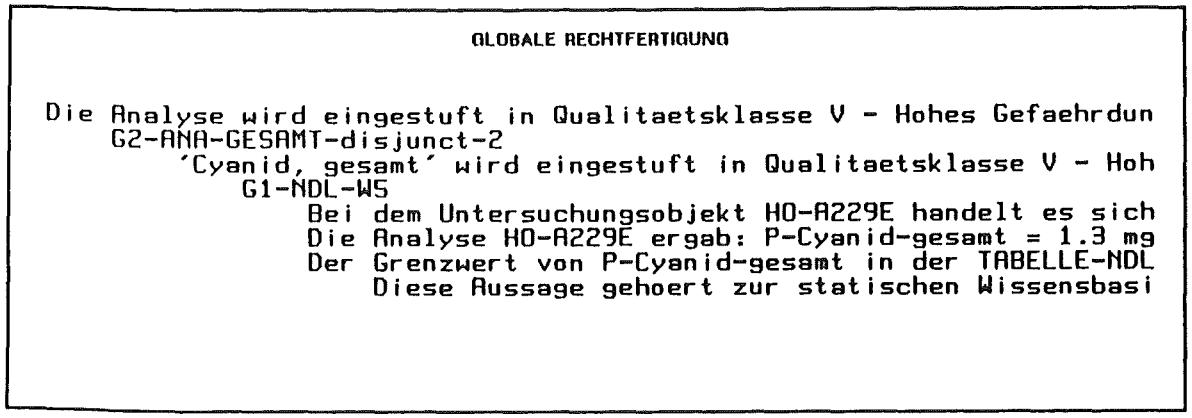


Abb. 5.3: Eine globale Rechtfertigung einer Aussage

kommentar angegeben. Dieser Regelkommentar wurde vom Systementwickler bei der Definition der Regel eingegeben und soll dem Anwender die Abhängigkeiten zwischen den Prämissen und den Aktionen der betrachteten Regel darlegen.

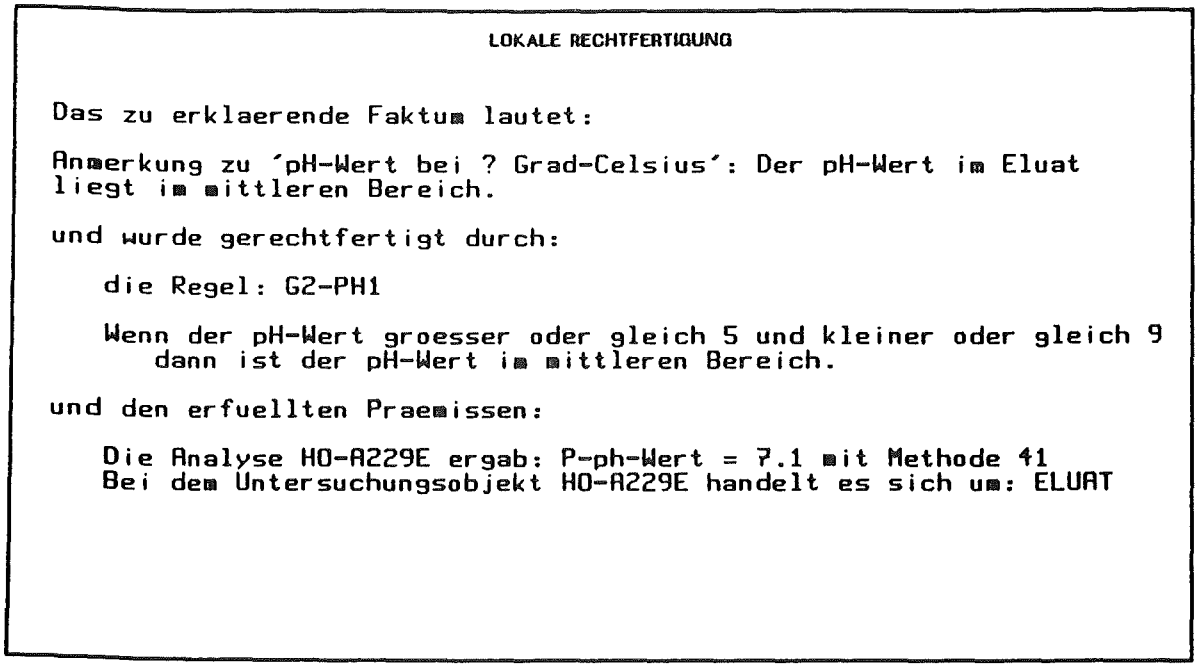


Abb. 5.4: Eine lokale Rechtfertigung einer Aussage

Neben den Prämissen, die genauso wie die Bewertungsaussagen erklärt werden, können auch Regeln erklärt werden. Durch Anklicken einer Regel wird dem Anwender ein Auswahlmenü zur Erklärung dieser Regel angezeigt (Abb. 5.5). Zum besseren Verständnis der Regel kann er sich den Regelkommentar, den Relevanzwert der Regel oder den Programmcode der Regel anzeigen lassen.

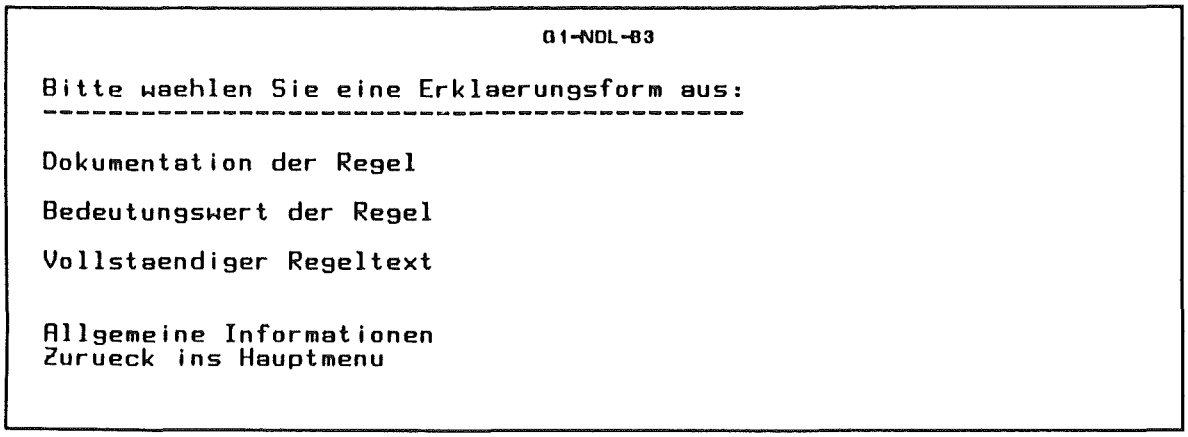


Abb. 5.5: Auswahlmenü zur Erklärung einer Regel

Da ART die Faktinformationen der Wissensbasis in der erwähnten Relationenschreibweise darstellt, müssen diese erst in eine benutzergerechte Form umgesetzt werden. Dies geschieht durch die Generierung natürlichsprachlicher Aussagen mithilfe selbstdefinierter Umsetzungsfunktionen. Dadurch wird vermieden, daß der Anwender Kenntnisse über die Syntax von ART und den internen Aufbau des Expertensystems haben muß, um die Erklärungen verstehen zu können.

5.2.3 Wissensrepräsentation bei XUMA

In Kapitel 3.3.3 wird als Basis für die Ermittlung einer geeigneten Wissensrepräsentationsform eine Abbildung des Expertenwissens in verschiedene Wissensarten gefordert.
Eine Einteilung in diesem Sinne, wie bei dem KADS-Modell oder dem EES-I-System, existiert in XUMA zwar nicht explizit, aber die Wissensbasis

wurde dem Anwendungsgebiet entsprechend strukturiert. Das statissche Wissen in XUMA, wie das Wissen über Branchen, Stoffe etc., wird in semantischen Netzen aus Schemas dargestellt (derzeit ca. 1200 Schemas), die über verschiedene Relationen miteinander verbunden sind. Dieses Wissen wird in erster Linie zur Analysenplanerstellung benötigt. Das sonstige deklarative Wissen, wie Zwischenergebnisse und Bewertungsaussagen, wird auf ART-Fakten abgebildet. Das gesamte heuristische Wissen (beispielsweise die Bewertung der Stoffgefährlichkeit) wird in XUMA mithilfe von ART-Regeln realisiert (bis jetzt ca. 800 Regeln), wobei die gesamte Regelmenge in mehrere Regelpakete ('Rule-sets') aufgeteilt wurde. Mithilfe einer Phasensteuerung (über die Relation 'current-phase') wird erreicht, daß jeweils nur die Regeln eines Regelpaketes 'gleichzeitig' ausgeführt werden. Um eine Trennung zwischen Fachwissen und Ablaufsteuerung zu erreichen, werden in XUMA sogenannte Metaregeln (zur Ablaufsteuerung) und fachspezifische Regeln unterschieden. Da es in ART aber nicht möglich ist, einzelne Regelpakete isoliert zu aktivieren, konnte in einigen fachspezifischen Regeln nicht auf zusätzliches Kontrollwissen verzichtet werden.
Wie bereits geschildert, ist das Wissen in XUMA zur Bewertung der Stoffgefährlichkeit in ART-Regeln kodiert. Diese Regeln werden, bis auf wenige Ausnahmen (beispielsweise bei Unschärfeoperatoren), vorwärtsverkettet abgearbeitet und liefern die Bewertungsaussagen in Form von ART-Fakten. Die Ableitungsbäume sind in diesem Anwendungsfall zwar sehr breit, da unter Umständen viele Eingangsdaten existieren, jedoch eher flach, da die Ableitungsketten meist nach Anwendung von 3 bis 10 Regeln in einer Bewertungsaussage enden. Da bei der Abfallbewertung die Analysenergebnisse bei Beginn einer Sitzung vollständig vorliegen und interpretiert werden müssen, kommen in XUMA keine expliziten Strategien (wie 'Hypothesize and Test' in der Diagnostik) zur Anwendung. Aus diesem Grund und da zudem keine Einschränkungen zur Vermeidung einer kombinatorischen Explosion notwendig sind, wie etwa bei rückwärtsverkettenden Diagnostik-Expertensystemen, kann in XUMA auf eine explizite Darstellung von Strategien verzichtet werden.

Zur Ablaufsteuerung werden in XUMA neben den Metaregeln, die in erster Linie die einzelnen Regelpakete steuern, auch andere ART-Regeln und Lisp-Funktionen eingesetzt. Damit konnte jeweils problemspezifisch die optimale

Wissensrepräsentation für das prozedurale Wissen erreicht werden. Beispielsweise ist in XUMA die Verwaltung der einfacheren Menüs mit ART-Regeln implementiert, während komplexere Menüs mithilfe einer mächtigen Lisp-Funktion realisiert werden.

Um die Erklärungsfähigkeit von XUMA den Anforderungen der Anwender anzupassen, mußte in der statischen Wissensbasis zusätzliches Wissen eingebracht werden. Dazu gehörte unter anderem ein natürlichsprachlicher Kommentar in Wenn-Dann-Form zu den Regeln und der Relevanzwert von Regeln und Relationen. Dieses Wissen war notwendig, um die Erklärungen benutzerklassenabhängig und in einer benutzeradäquaten Form generieren zu können. Eine Unterstützung der Rechtfertigung von Aussagen durch kausales Wissen war bei XUMA nicht notwendig. Im Bereich der Altlastenbewertung spielen sehr komplexe physikalisch/chemische Prozesse eine Rolle, bei denen entsprechendes kausales Wissen nur beschränkt vorhanden und daher nicht zum Einsatz in Expertensystemen geeignet ist.
Auch auf den Einsatz von Gewißheitsfaktoren konnte in XUMA bisher verzichtet werden, da die von dem Fachexperten definierten Regeln immer gelten und sich die Unsicherheiten von Aussagen nicht quantifizieren lassen, sondern auf Aussagen der Form "gilt" beziehungsweise "gilt vielleicht" beschränken.

5.3 Implementierung der Erklärungskomponente

Im folgenden wird erläutert, wie die im vorigen Kapitel beschriebenen Konzepte mithilfe von ART und Lisp realisiert wurden. Dabei soll auf den eigentlichen Programmcode nur ansatzweise eingegangen werden.
Die Erklärungskomponente muß in erster Linie die dynamischen Schlußfolgerungsprozesse nachvollziehen und rechtfertigen. Das dazu notwendige deklarative und terminologische Wissen beschränkt sich auf die Relevanzwerte, die Regelkommentare und die sprachlichen Umsetzungsfunktionen. Dieses Wissen gehört zur statischen Wissensbasis und ist nicht an die Erklärungskomponente gebunden. Das notwendige prozedurale Wissen für die Erklärungskomponente wurde in Lisp-Funktionen und in ART-Regeln kodiert. Da das Expertensystem-Werkzeug ART einen leistungsfähigen Mechanismus zur Fenster- und Menüverwaltung anbietet, erfolgt die gesamte Menüverwaltung der Erklärungskomponente mithilfe von ART-

Regeln. Wenn in ART eine maus-sensitive Menüzeile angeklickt wird, so produziert das System ein Faktum ('utterance-fact'), das alle notwendigen Informationen (Name des Fensters, Mauskoordinaten, ...) enthält, und das dann mithilfe von ART-Regeln weiterverarbeitet werden kann.
Die eigentlichen Erklärungen von Aussagen werden jedoch mit Lisp-Funktionen generiert, da Lisp unter anderem sehr stark das rekursive Durchlaufen des Ableitungsbaumes unterstützt. Somit ergaben sich bei der Realisierung der Erklärungskomponente drei Teilaufgaben, deren Lösung im folgenden beschrieben wird.

5.3.1 Erzeugung der Rechtfertigungen von Fakten

Jedem Fakt der Wissensbasis in ART ist zur Identifikation eine Nummer zugeordnet, die sogenannte Faktnummer. Mithilfe dieser Faktnummer kann über Schnittstellenfunktionen, die ART zur Verfügung stellt, auf alle notwendigen Informationen über ein Fakt zugegriffen werden. ART bietet beispielsweise eine Funktion an, die bei Übergabe einer Faktnummer eine Liste der Regeln und Prämissen (als Faktnummern) ausgibt, aus denen das Fakt unmittelbar hergeleitet wurde. Bei der lokalen Erklärung reicht ein einmaliger Funktionsaufruf aus, um die notwendige Ausgabe zu erzeugen. Bei einer globalen Erklärung muß jedoch mithilfe eines rekursiven Funktionsaufrufs eine Klammerstruktur erzeugt werden, die dem Ableitungsbaum entspricht (Abb. 5.6). Die Rekursion wird dabei für jedes Faktum solange fortgesetzt bis das gerade untersuchte Fakt zur statischen Wissensbasis gehört oder schon gelöscht wurde. Gehört ein Fakt zur statischen Wissensbasis, so wurde es entweder bei der Definition der Wissensbasis oder vom Benutzer eingegeben, aber nicht aus Regeln abgeleitet. Solch ein Fakt kann nicht weiter gerechtfertigt werden. Wurde in ART ein Fakt gelöscht, wie es schon beim Modifizieren einer Relation geschieht, wird zwar die Faktnummer noch geführt, jedoch keine weiteren Informationen (beispielsweise zu welcher Relation das Fakt gehörte). Zu diesem Faktum existieren somit weder erklärende Informationen noch Rechtfertigungen.

```
                                    COMMAND WINDOW
(drucke-ableitungs-struktur 4991)

(4991
 ((GI-NDL-B4
   ((4980 ((RETRACTED))) (4966 ((RETRACTED)))
    (4703
     ((FALL-ALTE-DATEN-HOLEN
       ((4622 ((RETRACTED)))
        (4624
         ((FALL-AKTUELL-IST ((4622 ((RETRACTED))) (4623 ((RETRACTED))))))))))
    )
    (4708
     ((FALL-ALTE-DATEN-HOLEN
       ((4622 ((RETRACTED)))
        (4624
         ((FALL-AKTUELL-IST ((4622 ((RETRACTED))) (4623 ((RETRACTED))))))))))
    )
    (2705 ((JUSTIFIED-BY-DEFINITION)))))))


NIL
 ART3.0 Lisp>
```

Abb. 5.6: Rekursive Rechtfertigungsstruktur eines Faktums

5.3.2 Ausfiltern der relevanten Informationen

Um bei den Erklärungen die gewählte Benutzerklasse berücksichtigen zu können, müssen die für die jeweilige Benutzerklasse relevanten Informationen aus dem gesamten Ableitungsbaum extrahiert werden. Dazu muß vorher in der statischen Wissensbasis für jede Regel und jede Relation ein Relevanzwert definiert werden. Dieser Relevanzwert bezieht sich auf die Relevanz der Information aus Anwendersicht und wird durch eine ganze Zahl im Intervall [0...5] repräsentiert. Ein niedriger Relevanzwert (0 oder 1) wird den Regeln und Relationen zugeordnet, die nur für den Systementwickler von Interesse sind, nicht aber für den Anwender. Der Relevanzwert wird jeder Regel durch ein spezielles Codewort (':evidence') innerhalb der Kommentarzeile der Regel zugeordnet (Abb. 5.7), während die Relevanzwerte der Relationen in einer speziellen Tabelle (Abb. 5.8) verwaltet werden. Der schnelle Zugriff auf diese Tabelle wird dabei über die Lisp-Funktion 'assoc' gewährleistet. Ist zu einem Konzept kein Relevanz-

wert vorhanden, so wird standardmäßig der höchste Wert (d.h. 5) angenommen.

```
                                        COMMAND WINDOW
(drucke-regel 'g4-stat2)

(DEFRULE G4-STAT2
          ":comment Wenn ein Parameter fuer die Ergebnis-Statistik
                    vorgemerkt ist,
                    dann wird der Parameter in die Ergebnis-Statistik
                    aufgenommen.
        :evidence 3"
         (CURRENT-PHASE STOFFGEF (FALL $?))
         (AKTUELL-IST FALL ?FALL)
         (STAT-FACT ? ?PARAMETER ? ?)
         (NOT (WERTUNG ?FALL (FALL-STATISTIK ?PARAMETER ? ?)))
    =>
         (ASSERT (SCHEMA ?FALL
                  ((WERTUNG
                    (FALL-STATISTIK ?PARAMETER
                     ((QK-I 0) (QK-II 0) (QK-III 0) (QK-IV 0) (QK-V 0)
                      (QK-VI 0))
                     0))
                   NIL)))
         )

NIL
 ART3.0 Lisp>
```

Abb. 5.7: Bewertungsregel in ART mit speziell formatierter Kommentarzeile

```
                                        COMMAND WINDOW
(drucke-relation-tabelle)


(CURRENT-PHASE 3 "aktuelle Phase")
(AKTUELL-IST 3 "welche Analyse wird gerade betrachtet")
(RETRACTED 0 "Das Fakt ist schon geloescht worden")
(WERTUNG 5 "Bewertungs-Fakten")
(XUMA-MENU 0 "Menu-Fakten")
(INSTANCE-OF 0 "Ist Instanz einer Klasse")
(STOFFLISTE 3 "beinhaltet interne und externe Namen von Stoffen")
(UNTERSUCHUNGSOBJEKT 3 "Es handelt sich um Boden/Eluat/Luft/Grundwasser")
(GREATER-THAN 0 "Groesser-als Relation zum Vergleich von Qualitaetsklassen")


NIL
 ART3.0 Lisp>
```

Abb. 5.8: Tabelle zur Verwaltung der Relevanzwerte von ART-Relationen

Bei der Generierung des Ableitungsbaumes wird jeder Zweig einzeln betrachtet. Entspricht der Relevanzwert einer Regel der aktuellen Benutzerklasse (Regel-Relevanzwert größer als Benutzerklasse), so wird dieser Zweig weiter verfolgt. Ist dies nicht der Fall, so wird dieser Zweig dem Benutzer nicht angezeigt. Nach der Betrachtung der Regel erfolgt die Betrachtung der zugehörigen Prämissen analog. Dabei werden nur diejenigen Fakten, die benutzerrelevant sind, ausgegeben und weiterbetrachtet. Eine durch Wahl einer höheren Benutzerklasse entsprechend reduzierte Ableitungsstruktur zu Abbildung 5.5 ist in Abbildung 5.8 dargestellt.

```
                                    COMMAND WINDOW
(drucke-ableitungs-struktur 4991)

(4991
 ((G1-NDL-B4 ((4703 NIL) (4708 NIL) (2705 ((JUSTIFIED-BY-DEFINITION)))))))


NIL
 ART3.0 Lisp>
```

Abb. 5.8: Verkürzte Ableitungstruktur

Durch dieses Verfahren können beliebig genau diejenigen Informationen für den Anwender herausgefiltert werden, die für ihn interessant und auch verständlich sind.

5.3.3 Sprachliche Aufbereitung und Ausgabe

Die Ausgabe einer rekursiven Ableitungsstruktur reicht alleine nicht aus, um dem Anwender die gewünschten Informationen benutzeradäquat zur Verfügung zu stellen. Beispielsweise müssen den Faktnummern die entsprechenden Fakt-Inhalte zugeordnet werden, und die Struktur muß dem Anwender in einer verständlichen Form präsentiert werden. Bei der Aufarbeitung der Informationen der Ableitungsstruktur fallen zwei Hauptaufgaben an:

a) Umsetzung der Klammerstruktur in eine pseudografische Darstellung

b) Ausgabe der Faktinformation in natürlichsprachlicher Form

Die erforderliche Umsetzung der Klammerstruktur erfolgte durch die Abbildung des Ableitungsbaumes auf eine eingerückte Struktur. Dabei werden alle Informationen soweit nach hinten eingerückt, wie es ihrer Tiefe im Ableitungsbaum entspricht. Dazu wird die erzeugte Ableitungsstruktur rekursiv abgearbeitet, und abhängig von der Baumtiefe werden entsprechend viele Leerzeichen vor die Information gesetzt. Die eingerückten Regeln und Aussagen werden dann in einer Liste sortiert, die in ART als Faktum eingetragen wird. Die Menüverwaltung übernimmt diese Liste und generiert daraus ein maus-sensitives Menü, das in dem Ausgabefenster ausgegeben wird. Die eingerückte Struktur ermöglicht dem Anwender einen schnellen Überblick über den gesamten Schlußfolgerungsprozeß, der zu einer Aussage geführt hat.

Da der Anwender die Faktinformationen, wie sie in der Wissensbasis enthalten sind (in der Form von Relationstupeln), nicht deuten kann, müssen die Fakten in einer natürlichsprachlichen Form ausgegeben werden. Die Definition einer Grammatik und eines Sprachgenerators war im Falle von XUMA nicht sinnvoll, da eine Erklärung für einen Anwender nur auf etwa zehn verschiedene Relationen zugreift.
In ART gehört jedes Fakt zu einer vorher definierten Relation, und somit war es möglich, für jede relevante Relation eine sprachliche Umsetzung zu definieren. Dazu muß jeder Relation in der Wissensbasis eine Lisp-Funktion zugeordnet werden, die aus den Attributen des Relationstupels mithilfe von Füllwörtern einen deutschen Satz generiert. Bei einer Erklärung wird nachgesehen, wie die Umsetzungsfunktion für die betrachtete Relation heißt und diese dann aufgerufen. Durch dieses Vorgehen werden die Fakten in der Wissensbasis so aufbereitet, daß der Anwender sie ohne genauere DV-Kenntnisse verstehen kann.

6 Zusammenfassung und Ergebnisse

In der vorliegenden Arbeit wurden verschiedene Konzepte zur Realisierung von Erklärungskomponenten herausgearbeitet. Dabei standen drei Teilaspekte im Mittelpunkt: das Benutzermodell, die Dialogschnittstelle und die Wissensrepräsentation. Das Benutzermodell dient dazu, die spezifischen Anforderungen der Anwender an das Expertensystem zu beschreiben und zu strukturieren. Die erarbeiteten Anforderungen bezüglich der Benutzerschnittstelle können dann innerhalb der Dialogschnittstelle auf die vorhandenen Ein- und Ausgabemöglichkeiten abgebildet werden. Die Grundlage der Erklärungsfähigkeit eines Expertensystems ist jedoch die explizite Darstellung des eingesetzten Wissens. Da nur das explizit vorhandene Wissen zur Erklärung verwendet werden kann, sollte bei der Definition einer Wissensbasis eine Strukturierung, etwa in Form des KADS-Modells, zu Grunde gelegt werden.

Am Beispiel des Expertensystems XUMA zur Bewertung der Stoffgefährlichkeit von Abfällen wurde gezeigt, wie sich die beschriebenen Konzepte in die Praxis umsetzen lassen. Dabei wurde in XUMA versucht, die unterschiedlichen Anforderungen verschiedener Benutzerklassen (Anwender, Fachexperte und Wissensingenieur) zu berücksichtigen. Zu diesem Zweck werden dem Benutzer abhängig von seiner Benutzerklasse nur diejenigen Informationen (Regeln und Aussagen) bei einer Erklärung ausgegeben, die für ihn relevant sind.

Weitere wichtige Ergebnisse dieser Arbeit sind:

- Es führt zu Problemen, wenn eine Erklärungskomponente nachträglich auf eine vorhandene Wissensbasis aufgesetzt werden soll. Die Realisierung der Erklärungskomponente sollte besser schon bei der Konzeption des Expertensystems berücksichtigt werden.
- Um eine gute Erklärungsfähigkeit zu erreichen, muß zusätzliches Wissen in die Wissensbasis eingebracht werden, zum Beispiel Wenn-Dann-Kommentare zu den Regeln. Außerdem genügt es nicht, dem Anwender die Informationen der Wissensbasis in der Darstellung zu präsentieren, wie sie intern vom Expertensystem verwendet wird, sondern sie müssen in eine benutzergerechte Form gebracht werden.

- Eine konsequente Strukturierung der Wissensbasis in verschiedene Wissensarten, zum Beispiel in deklaratives, terminologisches und prozedurales Wissen, trägt viel dazu bei, die Akzeptanz des Anwenders zu erhöhen, da der Anwender dann mithilfe der Erklärungskomponente die Schlußfolgerungen des Expertensystems besser nachvollziehen kann.
- Der eingesetzte Lisp-Rechner vom Typ Explorer II der Firma Texas Instruments hat sich als sehr leistungsfähiges Entwicklungssystem erwiesen. Besonders durch den inkrementellen Lisp-Compiler und durch die hilfreiche Unterstützung von verschiedenen Debug-Tools waren kurze Programmier-Test-Debug-Zyklen möglich.
- Das hybride Expertensystem-Werkzeug ART zeigte sich als stabiles Hilfsmittel mit vielen Möglichkeiten zur Wissensdarstellung, wobei jedoch die Abgeschloßenheit des Systems von Nachteil war, da auf bestimmte Funktionen, die ART im Dialogmodus zum Testen der Wissensbasis anbietet, nicht vom Programm aus zugegriffen werden kann.

Literaturverzeichnis:

[Appelrath-85]
Hans Jürgen Appelrath, "Von Datenbanken zu Expertensystemen", Informatik Fachberichte, 102, Springer Verlag (1985);

[Buchanan, Shortliffe-84]
Bruce.G.Buchanan, Edward H.Shortliffe, "Rule-Based Expert Systems; The MYCIN Experiments of the Stanford Heuristic Programming Project", Addison-Wesley Publishing Company (1984);

[Chandrasekaran et al.-86]
B.Chandrasekaran, John Josephson and Anne Keuneke, "Functional Representations as a Basis for Generating Explanations", International Conference on Systems, Man and Cybernetics, Proceedings of the IEEE, p.726-31 (1986);

[Charniak, McDermott-85]
E.Charniak, D.McDermott, "Statistics in Abduction", Artificial Intelligence, Kapitel 8.2, Addison Wesley (1985);

[Clancey-83]
William J.Clancey, "The Epistemology of a Rule-Based Expert System -a Framework for Explanation", Artificial Intelligence, 20, p.215-251 (1983);

[Davis-80]
Randall Davis, "Meta Rules: Reasoning about Control", Artificial Intelligence, 15, p.179-222 (1980);

[Davis-84]
Randall Davis, "Diagnostic Reasoning Based on Structure and Behavior", Artificial Intelligence, 24, p.347-410 (1984);

[Dhar, Pople-87]
Vasant Dhar and Harry E.Pople, "Rule-Based versus Structure- Based Models for Explaining and Generating Expert Behavior", Communications of the ACM, 30 (6), p.542-555 (1987);

[Ganascia-85]
Jean-Gabriel Ganascia, "Explanation and Justification in Expert Systems", Computers and Artificial Intelligence, 4 (1), p.3-13 (1985);

[Gordon, Shortliffe-85]
J.Gordon, E.Shortliffe, "A Method for Managing Evidential Reasoning in a Hierarchical Hypothesis Space", AI-Journal, 26, p.323-357 (1985);

[Harman-65]
G.Harman, "The Inference to the Best Explanation", Philosophical Review. Ithaca, N.J. 124, p.88-95 (1965);

[Harmon, King-86]
Paul Harmon, David King, "Expertensysteme in der Praxis", R.Oldenbourg Verlag München Wien (1986);

[Hasling et.al.-84]
Diane W.Hasling, William J.Clancey and Glenn Rennels, "Strategic Explanations for a Diagnostic Consultation System",
International Journal of Man-Machine Studies, 20 (1), p.3- 19 (1984);

[Hawkins-83]
David Hawkins, "An Analysis of Expert Thinking",
International Journal of Man-Machine Studies, 18, p.1-47 (1983);

[Helman, Bennett-86]
David H.Helman, Jeffrey L.Bennett, "Theories of Explanation: Expert Systems and Simulations",
Symposium on Human Factors in Management Information Systems, Texas A&M University, (1986);

[Hughes-86]
Sheila Hughes, "Question Classification in Rule-Based Systems",
The Sixth Annual Conference of British Computer Society Specialist, Proceedings of Expert Systems, p.123-131 (1986);

[Hushon-87]
Judith M.Hushon, "Expert Systems for Environmental Problems",
Environment Science Technology, 21 (9), p.838-841 (1987);

[Josephson et al.-87]
John R.Josephson, B.Chandrasekaran, Jack W.Smith, "A Mechanism for Forming Composite Explanatory Hypotheses",
IEEE Transactions on Systems, Man and Cybernetics, 17 (3), p.445-454 (1987);

[Karras et al.-87]
D.Karras, L.Kredel, U.Pape, "Entwicklungsumgebungen für Expertensysteme",
Walter de Gruyter Verlag (1987);

[Ladwig, Mellis-87]
Britta Ladwig, Werner Mellis, "Negative Erklärungen in einem Emycin-artigen Expertensystem-Shell",
in "Expertensysteme 87, Konzepte und Werkzeuge", B.G.Teubner Verlag Stuttgart, p.150-168 (1987);

[Molokova-86]
O.S.Molokova, "The Formation of an Individual Explanation in Expert Systems",
Soviet Journal of Computer and Systems Sciences Volume 24 (1), p.34-44 (1986);

[Phillips et al.-86]
Elizabeth R.Phillips, David R.Eike & Stephen A.Fleger, "Human Factors Issuses in Designing Explanation Facilities for Expert Troubleshooting Systems",
Proceedings of the Human Factors Society, 1, p.502-506 (1986);

[Pleszkun, Thazhuthaveetil-87]
Andrew R.Pleszkun, Matthew J.Thazhuthaveetil, "The Architecture of Lisp Machines",
IEEE Computer, 20 (3), p.35 (1987);

[Puppe-88]
Frank Puppe, "Einführung in Expertensysteme",
Springer Verlag (1988);

[Rahmstorf-87]
G.Rahmstorf, "Grafische Funktionen in der künstlichen Intelligenz", Proceedings 2.int.GI-Kongreß, "Wissensbasierte Systeme", München, Informatik-Fachberichte, 155, p.207-216 (1987);

[Rome, Uthmann-87]
Erich Rome, Thomas Uthmann, "KI-Workstations: Überblick, Marktsituation, Entwicklungstrends", GMD-Studien, Nr.118, (1987);

[Savory-86]
Stuart E.Savory, "What is an Explanation?", Proceedings of the International Conference 'Knowledge Based Systems' in London, p.231-238, (1986);

[Shortliffe, Buchanan-75]
E.Shortliffe, E.Buchanan, "A Model of Inexact Reasoning in Medicine", Mathematical Bioscience, 23, p.351-379 (1975);

[Schmitt-83]
A.Schmitt, "Dialogsysteme", Bibliografisches Institut, Reihe Informatik, 40, (1983);

[Schnupp, Huu-87]
P.Schnupp, C.T.Nguyen Huu, "Expertensystem-Praktikum", Springer Verlag Berlin Heidelberg (1987);

[Sinha et al.-84]
A.Sinha, L.Caplan, B.Desai, "Making Decision Support Systems self Explanatory", Proceedings of the IEEE Computer Society's Eighth International Computer Software & Applications Conference, p.358-362 (1984);

[Szolovits, Pauker-78]
P.Szolovits, Pauker, "Categorical and Probabilistic Reasoning in Medical Diagnosis, AI-Journal, 11, p.115-144 (1978);

[Strat-87]
Thomas M.Strat, "The Generation of Explanations within Evidential Reasoning Systems", Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI) 1987, p.1097-1104 (1987);

[Swartout, Smoliar-87]
William R.Swartout, Stephen W.Smoliar, "On Making Expert Systems More Like Experts", Expert Systems, 4 (3), p.196-207 (1987);

[Swartout, Smoliar-88]
William R.Swartout, Stephen W.Smoliar, "Explaining the Link Between Causal Reasoning and Expert Behavior", Symposium on Computer Applications in Medical Care, (1987);

[Trauboth-87]
H.Trauboth, "Was kann die Informationstechnik für den Umweltschutz tun?", Automatisierungstechnik - at, 35 (11), p.431-442 (1987);

[Umweltministerium-87]
Ministerium für Ernährung, Landwirtschaft, Umwelt und Forsten Baden-Württemberg, "Altlasten-Handbuch, Teil I, Altlasten-Bewertung", (1987);

[Voß et al.-87]
A.Voß, B.Müller, J.Walther, "Über ein Experiment zur Umsetzung einer BABYLON-Wissensbasis in das KADS-Modell: Erste Schritte auf dem Weg zur Modellierung wissensbasierter Systeme", Proceedings 2.int.GI-Kongreß, "Wissensbasierte Systeme", München, Informatik-Fachberichte, 155, p.358-368 (1987);

[Wallis, Shortliffe-82]
J.W.Wallis, E.H.Shortliffe, "Explanatory Power for Medical Expert Systems", Meth.Inform.Med., 21 (3), p.127-36 (1982);

[Weidemann et al.-87]
R.Weidemann, W.Geiger, W.Eitel, "Entwurf eines Expertensystems zur Bewertung von Abfallstoffen", in A.Jaeschke, B.Page (Hrsg.): Informatikanwendungen im Umweltbereich, Springer-Verlag Berlin Heidelberg, Informatik-Fachberichte 170, p.116 (1988);

[Weiner-80]
J.L.Weiner, "BLAH, a System Which Explains its Reasoning", Artificial Intelligence 15, pp 19-48 (1980);

[Wielinga, Breuker-86]
B.J.Wielinga, J.A.Breuker, "Models of Expertise", Proc. European Conf. on Artificial Intelligence (ECAI), p.497-509 (1986);

[Zadeh-75]
L.Zadeh, "Fuzzy Logic and Approximate Reasoning", Synthese, 30, p.407-428 (1975);